Tasca M6 T01

Exercici 1

Crea almenys dos models de regressió diferents per intentar predir el millor possible el preu de les vivendes (MEDV) de l'arxiu adjunt.

In [2]:

```
# Tratamiento de datos
# ==============================================================================
import pandas as pd
import numpy as np

# Gráficos
# ==============================================================================
import matplotlib.pyplot as plt
from matplotlib import style
import matplotlib.ticker as ticker
import seaborn as sns
from pprint import pprint

# Preprocesado y análisis
# ==============================================================================
#import statsmodels.api as sm
#import pingouin as pg
from scipy import stats
import random as rd
from imblearn.over_sampling import SMOTE

# Preprocesado y modelado
# ==============================================================================
from sklearn.datasets import load_boston
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
from sklearn.metrics import accuracy_score
from sklearn import metrics
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.model_selection import RepeatedKFold
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import ParameterGrid
from sklearn.tree import DecisionTreeRegressor
from sklearn.tree import plot_tree
from sklearn.tree import export_text
from sklearn.inspection import permutation_importance
import multiprocessing
from sklearn import neighbors, datasets, preprocessing
from sklearn.preprocessing import Normalizer
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import PolynomialFeatures
from sklearn import datasets
from sklearn.model_selection import KFold
import statsmodels.api as sm
from sklearn.model_selection import cross_val_predict


# Test Estadísticos
# ==============================================================================
```

```
from scipy.stats import pearsonr
from statistics import mode
from scipy.stats import shapiro
from scipy.stats import normaltest
from scipy.stats import anderson
from scipy.stats import pearsonr
from scipy.stats import spearmanr
from scipy.stats import kendalltau
from scipy.stats import chi2_contingency
from statsmodels.tsa.stattools import adfuller
from statsmodels.tsa.stattools import kpss
from scipy.stats import ttest_ind
from scipy.stats import ttest_rel
from scipy.stats import f_oneway
from scipy.stats import mannwhitneyu
from scipy.stats import wilcoxon
from scipy.stats import kruskal
from scipy.stats import friedmanchisquare

# Ajuste de distribuciones
# ==============================================================================
from scipy import stats
import inspect
from statsmodels.distributions.empirical_distribution import ECDF


# Configuración matplotlib
# ==============================================================================
plt.style.use('ggplot')
from statsmodels.graphics.gofplots import qqplot
from matplotlib import pyplot

# Configuración warnings
# ==============================================================================
import warnings
warnings.filterwarnings('ignore')
```

A) Data Frame

In [3]:

```
data= pd.read_csv(r"C:\Users\hecto\OneDrive\Documentos\IT Data Science\Sprint5_DS\da
data_original =data
data_original.head(5)
```

Out[3]:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222.0	18.7	396.90	5.33	36.2

Relevant Information: Concerns housing values in suburbs of Boston.

- Number of Instances: 506
- Number of Attributes: 13 continuous attributes (including "class" attribute "MEDV"), 1 binary-valued attribute.

- Attribute Information:

    1. CRIM : per capita crime rate by town
    2. ZN : proportion of residential land zoned for lots over "25,000 sq.ft".
    3. INDUS : proportion of non-retail business acres per town
    4. CHAS : Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
    5. NOX : nitric oxides concentration (parts per 10 million)
    6. RM : average number of rooms per dwelling
    7. AGE : proportion of owner-occupied units built prior to 1940
    8. DIS: weighted distances to five Boston employment centres
    9. RAD . index of accessibility to radial highways
    10. TAX : full-value property-tax rate per 10.000 dólares
    11. PTRATIO: pupil-teacher ratio by town
    12. B : "1000(Bk - 0.63)^2" where Bk is the proportion of blacks by town
    13. LSTAT : % lower status of the population
    14. MEDV : Median value of owner-occupied homes in "$1000's"

In [4]:
```
data.columns =["CRIM","ZN","INDUS","CHAS","NOX","RM","AGE","DIS","RAD","TAX","PTRATI
data.head()
```

Out[4]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	M
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296.0	15.3	396.90	4.98	
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	17.8	396.90	9.14	
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242.0	17.8	392.83	4.03	
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222.0	18.7	394.63	2.94	
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222.0	18.7	396.90	5.33	

In [5]:
```
data.describe().transpose()
```

Out[5]:

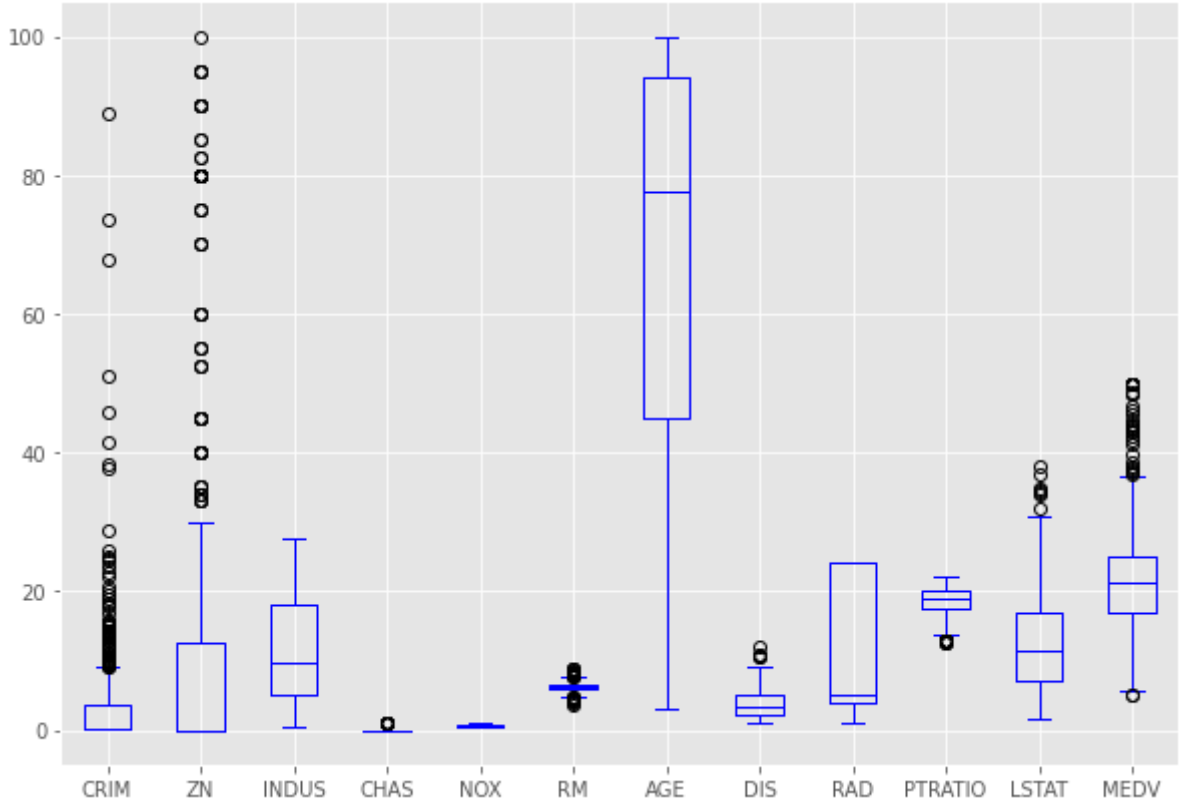
	count	mean	std	min	25%	50%	75%	max
CRIM	506.0	3.613524	8.601545	0.00632	0.082045	0.25651	3.677083	88.9762
ZN	506.0	11.363636	23.322453	0.00000	0.000000	0.00000	12.500000	100.0000
INDUS	506.0	11.136779	6.860353	0.46000	5.190000	9.69000	18.100000	27.7400
CHAS	506.0	0.069170	0.253994	0.00000	0.000000	0.00000	0.000000	1.0000
NOX	506.0	0.554695	0.115878	0.38500	0.449000	0.53800	0.624000	0.8710
RM	506.0	6.284634	0.702617	3.56100	5.885500	6.20850	6.623500	8.7800
AGE	506.0	68.574901	28.148861	2.90000	45.025000	77.50000	94.075000	100.0000
DIS	506.0	3.795043	2.105710	1.12960	2.100175	3.20745	5.188425	12.1265
RAD	506.0	9.549407	8.707259	1.00000	4.000000	5.00000	24.000000	24.0000
TAX	506.0	408.237154	168.537116	187.00000	279.000000	330.00000	666.000000	711.0000
PTRATIO	506.0	18.455534	2.164946	12.60000	17.400000	19.05000	20.200000	22.0000

	count	mean	std	min	25%	50%	75%	max
B	506.0	356.674032	91.294864	0.32000	375.377500	391.44000	396.225000	396.9000
LSTAT	506.0	12.653063	7.141062	1.73000	6.950000	11.36000	16.955000	37.9700
MEDV	506.0	22.532806	9.197104	5.00000	17.025000	21.20000	25.000000	50.0000

- Para tener una mejor aproximación a la dispersión de cada una de las variables explicativas hemos realizado la gráfica boxplot de cada una de ellas separadas en dos grupos diferenciados las que su rango de valores es inferior a 100, y las que superan este rango, que son la variable TAX y B.

In [6]:
```
data_sin= data.drop(columns=["TAX","B"])
data_sin.boxplot( grid = True, figsize=(10, 7), color="blue")
```

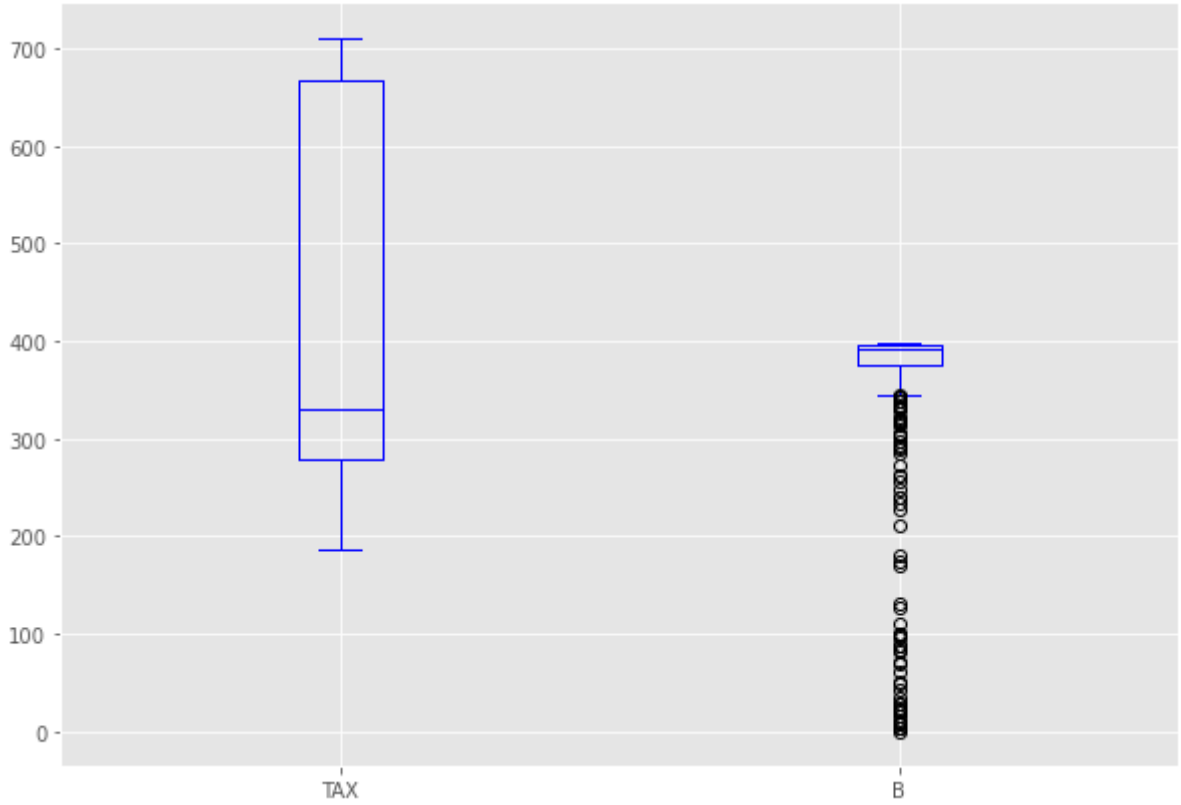
Out[6]: <AxesSubplot:>



- Las variables explicativas que presentan mas outliers son CRIM, ZN y LSTAT
- RM también presenta outliers tanto en los valores inferiores como superiores, pero la escala del gráfico no permtite apreciar si son o no numerosos.
- La variable objetivo MEDV también presenta outliers en los valores más elevados, por tanto cualquier modelo predictivo sobre esta variable presentará errores más significativos en el ajuste de los valores máximos.

In [7]:
```
data_tax_B= data[["TAX","B"]]
data_tax_B.boxplot(grid = True, figsize=(10, 7), color="blue")
```

Out[7]: <AxesSubplot:>



- La variable TAX no presenta outliers a pesar de ser la que tiene el rango más amplio, mientras que la variable B que es el porcentaje de población negra, al estar ajustada a una función exponencial presenta outliers en los valores inferiores.

B) Media y Desviación Estándar de cada una de las variables

In [8]:
```
df= data.describe().transpose()
data_medias=df[["mean","std"]]
data_medias
```

Out[8]:

	mean	std
CRIM	3.613524	8.601545
ZN	11.363636	23.322453
INDUS	11.136779	6.860353
CHAS	0.069170	0.253994
NOX	0.554695	0.115878
RM	6.284634	0.702617
AGE	68.574901	28.148861
DIS	3.795043	2.105710
RAD	9.549407	8.707259
TAX	408.237154	168.537116
PTRATIO	18.455534	2.164946
B	356.674032	91.294864
LSTAT	12.653063	7.141062

	mean	std
MEDV	22.532806	9.197104

C) Creación de Columnas Dummies

Dado que la variable CHAS solo tiene dos valores, 1,0, no tiene sentido separar en Dummies esa información, ya que la propia columna la contiene. Sin embargo, si vamo a crear columnas Dummies de RAD, para ver si esta separación de la información, añade más precisión al modelo.

In [9]:
```
# creamos dummy RAD
df=data
df=pd.get_dummies(df, columns=["RAD"],drop_first = False)
df.head()
```

Out[9]:

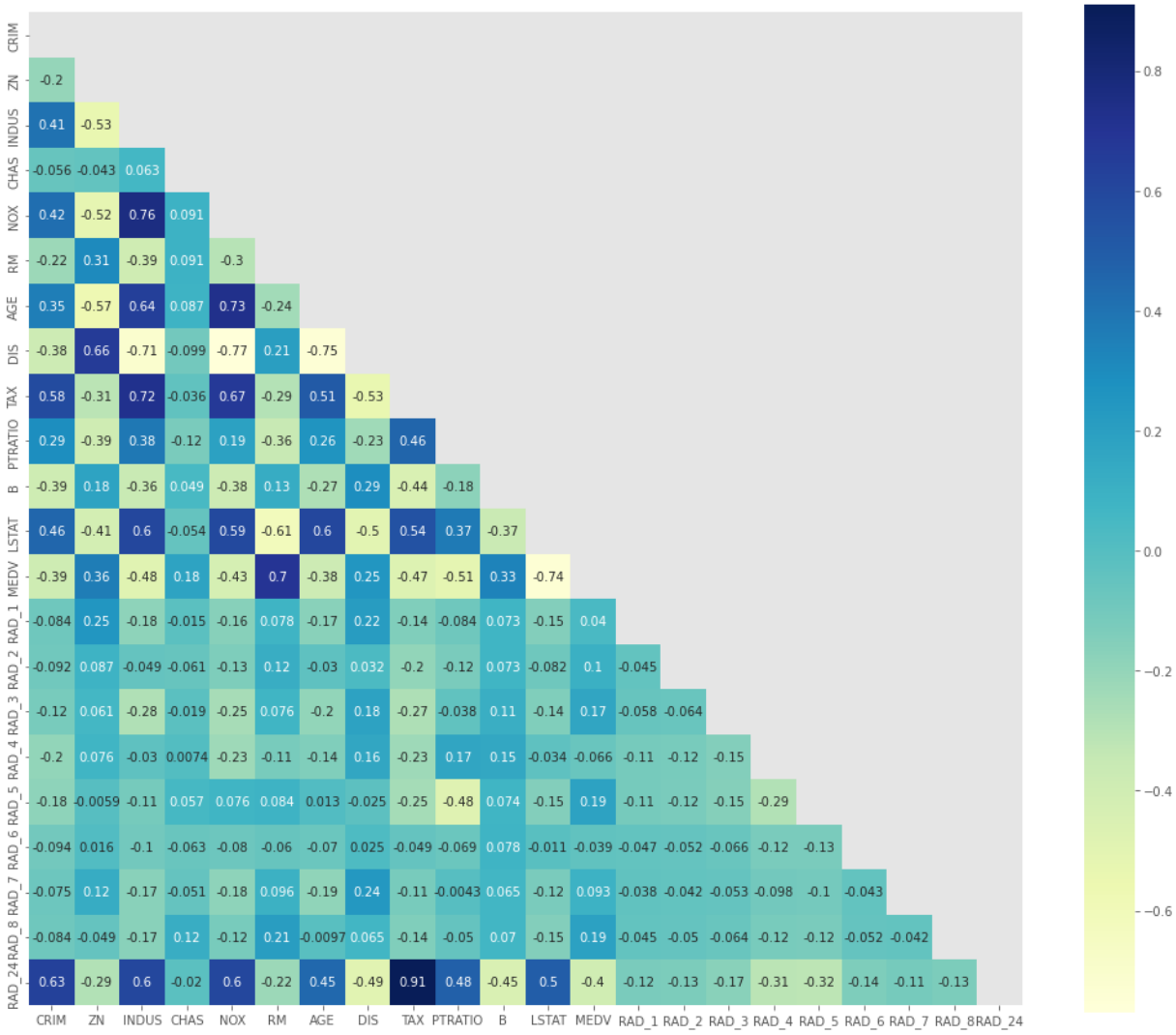
	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	TAX	PTRATIO	...	MEDV	RAD_1	RA
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	296.0	15.3	...	24.0	1	
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	242.0	17.8	...	21.6	0	
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	242.0	17.8	...	34.7	0	
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	222.0	18.7	...	33.4	0	
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	222.0	18.7	...	36.2	0	

5 rows × 22 columns

D) Correlación entre las variables del Data Frame

In [10]:
```
plt.figure(figsize=(18,15))
upp_mat = np.triu(df.corr())
sns.heatmap(df.corr(),cmap="YlGnBu", square = True,annot=True, mask = upp_mat)
```

Out[10]:
```
<AxesSubplot:>
```



- Entre la variable objetivo MEDV y el resto de variables, la correlación (inversa) más elevada se da con la variable LSTAT que es el porcentaje de población con menos ingresos.En los Los barrios con más trabajadores de clase baja (mayor valor de 'LSTAT') los inmuebles valdrán men
- La siguiente variable con mayor correlación es RM que es el promedio de habitaciones por vivienda. Las casas con más habitaciones (valor más alto de 'RM') valdrán más. Por lo general, las casas con más habitaciones son más grandes y pueden acomodar a más personas, por lo que es razonable que cuesten más dinero. Son variables directamente proporcionales.
- Los vecindarios con más proporción de estudiantes por maestro (mayor valor de 'PTRATIO') el valor de las casas está inversamente correlacionado, Si el porcentaje de estudiantes por maestro es mayor, es probable que en el vecindario haya menos escuelas, o mayor densidad de población, por lo que influye negativamente en el valor de los inmuebles.
- Una vez aplicados los dummies sobre la variable RAD, los rangos de correlación de cada una de las variables RAD_* generadas vs MEDV, oscilan entre [-0,4 y 0.19] mientras que la correlación inical era de -0.38.

E) División de los datos en train y test

In [11]:
```
# División de los datos en train y test
# ==============================================================================
#X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_st
```

```
X_train, X_test, y_train, y_test = train_test_split(df.drop(columns = "MEDV"),df['ME
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

```
(339, 21) (167, 21) (339,) (167,)
```

In [12]:
```
X_train.head()
```

Out[12]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	TAX	PTRATIO	...	LSTAT	RAD_1
378	23.64820	0.0	18.10	0	0.671	6.380	96.2	1.3861	666.0	20.2	...	23.69	0
127	0.25915	0.0	21.89	0	0.624	5.693	96.0	1.7883	437.0	21.2	...	17.19	0
82	0.03659	25.0	4.86	0	0.426	6.302	32.2	5.4007	281.0	19.0	...	6.72	0
463	5.82115	0.0	18.10	0	0.713	6.513	89.9	2.8016	666.0	20.2	...	10.29	0
207	0.25199	0.0	10.59	0	0.489	5.783	72.7	4.3549	277.0	18.6	...	18.06	0

5 rows × 21 columns

In [13]:
```
X_test.head()
```

Out[13]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	TAX	PTRATIO	...	LSTAT	RAD_1	I
307	0.04932	33.0	2.18	0	0.472	6.849	70.3	3.1827	222.0	18.4	...	7.53	0	
343	0.02543	55.0	3.78	0	0.484	6.696	56.4	5.7321	370.0	17.6	...	7.18	0	
47	0.22927	0.0	6.91	0	0.448	6.030	85.5	5.6894	233.0	17.9	...	18.80	0	
67	0.05789	12.5	6.07	0	0.409	5.878	21.4	6.4980	345.0	18.9	...	8.10	0	
362	3.67822	0.0	18.10	0	0.770	5.362	96.2	2.1036	666.0	20.2	...	10.19	0	

5 rows × 21 columns

E.1) Descripción estadística de X_train -X-test

In [14]:
```
X_train.describe().transpose()
```

Out[14]:

	count	mean	std	min	25%	50%	75%	max
CRIM	339.0	3.793986	8.669966	0.00632	0.08577	0.25387	3.81234	73.5341
ZN	339.0	10.963127	23.058729	0.00000	0.00000	0.00000	12.50000	100.0000
INDUS	339.0	11.377493	6.874160	0.46000	5.64000	9.90000	18.10000	27.7400
CHAS	339.0	0.079646	0.271145	0.00000	0.00000	0.00000	0.00000	1.0000
NOX	339.0	0.556238	0.117996	0.39200	0.44800	0.53800	0.62750	0.8710
RM	339.0	6.247490	0.669953	3.86300	5.87300	6.16700	6.58250	8.3980
AGE	339.0	69.173451	28.246657	6.00000	45.75000	79.70000	94.30000	100.0000
DIS	339.0	3.799550	2.114897	1.12960	2.09445	3.26280	5.03375	12.1265
TAX	339.0	408.890855	169.654094	187.00000	279.00000	330.00000	666.00000	711.0000

	count	mean	std	min	25%	50%	75%	max
PTRATIO	339.0	18.418584	2.176714	12.60000	17.00000	19.00000	20.20000	22.0000
B	339.0	356.552537	92.095470	0.32000	375.99000	391.45000	396.06000	396.9000
LSTAT	339.0	13.087522	7.340139	1.73000	7.41500	12.03000	17.13500	37.9700
RAD_1	339.0	0.038348	0.192319	0.00000	0.00000	0.00000	0.00000	1.0000
RAD_2	339.0	0.056047	0.230353	0.00000	0.00000	0.00000	0.00000	1.0000
RAD_3	339.0	0.070796	0.256864	0.00000	0.00000	0.00000	0.00000	1.0000
RAD_4	339.0	0.209440	0.407510	0.00000	0.00000	0.00000	0.00000	1.0000
RAD_5	339.0	0.230088	0.421511	0.00000	0.00000	0.00000	0.00000	1.0000
RAD_6	339.0	0.053097	0.224559	0.00000	0.00000	0.00000	0.00000	1.0000
RAD_7	339.0	0.032448	0.177450	0.00000	0.00000	0.00000	0.00000	1.0000
RAD_8	339.0	0.044248	0.205949	0.00000	0.00000	0.00000	0.00000	1.0000
RAD_24	339.0	0.265487	0.442245	0.00000	0.00000	0.00000	1.00000	1.0000

In [15]:
```
X_test.describe().transpose()
```

Out[15]:

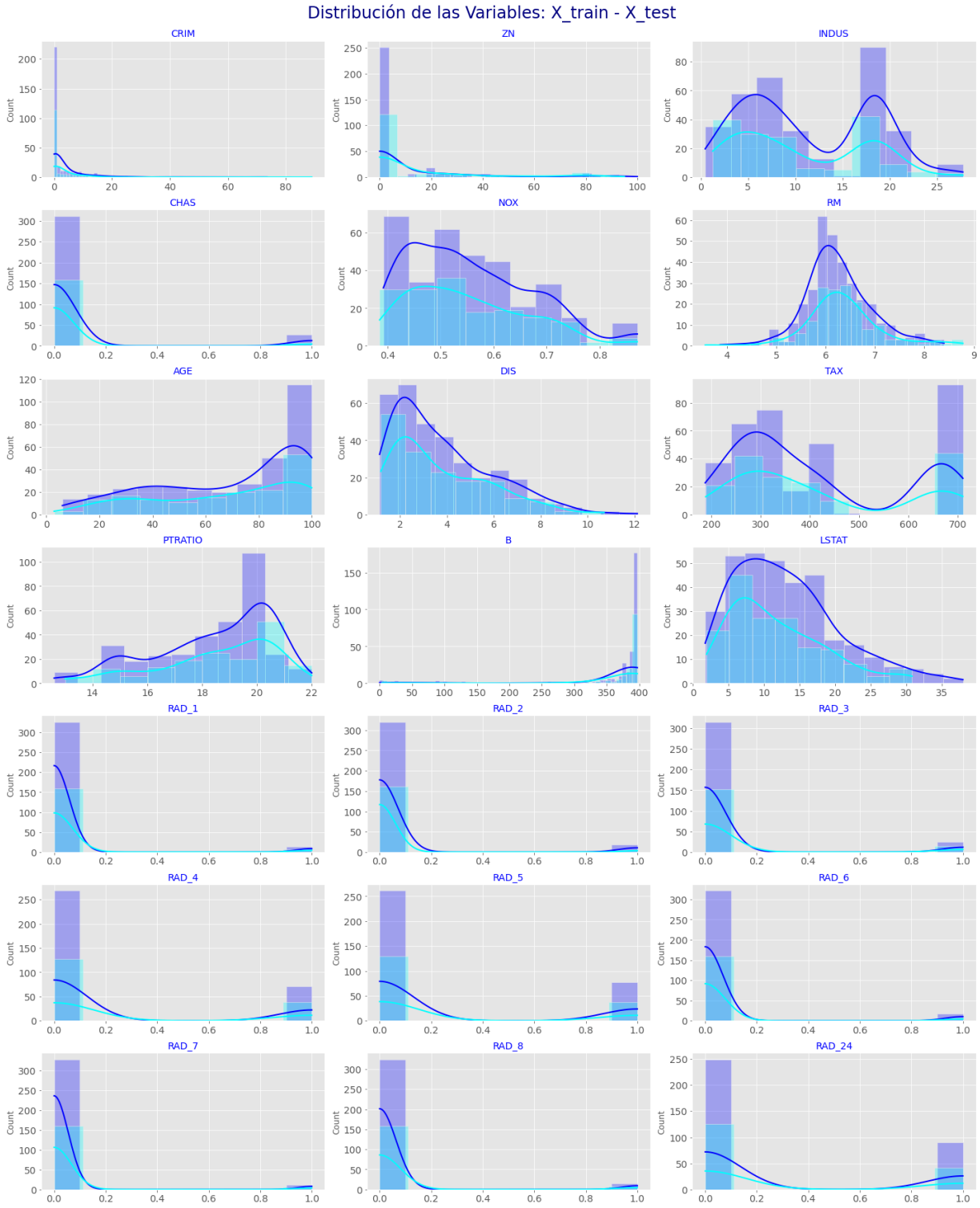
	count	mean	std	min	25%	50%	75%	max
CRIM	167.0	3.247195	8.474939	0.01096	0.076945	0.26169	3.122525	88.9762
ZN	167.0	12.176647	23.898258	0.00000	0.000000	0.00000	20.000000	95.0000
INDUS	167.0	10.648144	6.826662	1.25000	4.675000	8.56000	18.100000	27.7400
CHAS	167.0	0.047904	0.214206	0.00000	0.000000	0.00000	0.000000	1.0000
NOX	167.0	0.551563	0.111736	0.38500	0.458000	0.53200	0.624000	0.8710
RM	167.0	6.360036	0.761180	3.56100	5.979500	6.31000	6.630500	8.7800
AGE	167.0	67.359880	27.994376	2.90000	41.700000	73.30000	93.150000	100.0000
DIS	167.0	3.785894	2.093240	1.17810	2.101800	3.09230	5.307650	10.7103
TAX	167.0	406.910180	166.745525	188.00000	281.000000	337.00000	666.000000	711.0000
PTRATIO	167.0	18.530539	2.145399	13.00000	17.400000	19.10000	20.200000	22.0000
B	167.0	356.920659	89.921726	3.65000	373.715000	390.68000	396.660000	396.9000
LSTAT	167.0	11.771138	6.653123	1.92000	6.605000	10.24000	15.695000	30.8100
RAD_1	167.0	0.041916	0.201000	0.00000	0.000000	0.00000	0.000000	1.0000
RAD_2	167.0	0.029940	0.170935	0.00000	0.000000	0.00000	0.000000	1.0000
RAD_3	167.0	0.083832	0.277970	0.00000	0.000000	0.00000	0.000000	1.0000
RAD_4	167.0	0.233533	0.424351	0.00000	0.000000	0.00000	0.000000	1.0000
RAD_5	167.0	0.221557	0.416543	0.00000	0.000000	0.00000	0.000000	1.0000
RAD_6	167.0	0.047904	0.214206	0.00000	0.000000	0.00000	0.000000	1.0000
RAD_7	167.0	0.035928	0.186671	0.00000	0.000000	0.00000	0.000000	1.0000
RAD_8	167.0	0.053892	0.226484	0.00000	0.000000	0.00000	0.000000	1.0000

	count	mean	std	min	25%	50%	75%	max
RAD_24	167.0	0.251497	0.435178	0.00000	0.000000	0.00000	0.500000	1.0000

E.1) Descripción gráfica de X_train -X-test

In [16]:

```
# Distribución gráfica de cada variable de X_train
# ==============================================================================

fig, axes = plt.subplots(ncols=3, nrows=7, figsize=(20, 25))
axes = axes.flat
columnas_numeric = X_train.select_dtypes(include=['float64', 'int','uint8']).columns

for i, colum in enumerate(columnas_numeric):
    sns.histplot(data = X_train, x= colum, stat= "count",label="X_train", color="blu
    sns.histplot(data = X_test, x= colum, stat= "count", label= "X_test", color="cya
    axes[i].set_title(colum, fontsize = 14, fontweight = "ultralight", color="blue")
    axes[i].tick_params(labelsize = 14)
    axes[i].set_xlabel("")

fig.tight_layout()
plt.subplots_adjust(top = 0.95)
plt.suptitle('Distribución de las Variables: X_train - X_test', fontsize = 24, color
plt.savefig("Gafico1_Dist_Variables_X_train_test.png")
```



- X_train -color navy, X_test - color cyan (* labels definidos pero no aparecen en la imagen)
- RM sigue claramente una distribución normal así como MEDV la variable objetivo.
- Las variables INDUS, RAS Y TAX siguen una distribución que se aproximan más a una distribución polinómica.
- LSTAT, AGE, DIS, PTRATIO y NOX se aproximan más a una distribución Gaussiana con sesgo
- El resto de variables CRIM, ZN, CHAS podrían ajustarse a exponenciales decrecientes o crecientes en el caso de B.

E.2) Student's t-test

- Un t-test de Student es una herramienta para evaluar las medias de uno o dos grupos mediante pruebas de hipótesis.

- Una prueba t puede usarse para determinar si un único grupo difiere de un valor conocido (una prueba t de una muestra) o bien, si dos grupos difieren entre sí (prueba t de muestras independientes), y finalmente si hay una diferencia significativa en medidas pareadas (una prueba t de muestras dependientes)
- En nuestro caso vamos a comparar si la variables LSTAT tienen la misma o diferente distribución para Los subgrupos X_train y X_test, para un nivel de significación= 5%

In [17]:
```
print("====================")
print("  Student's t-test  ")
print("====================")
#from scipy.stats import ttest_ind
data1 = X_train["LSTAT"]
data2 = X_test["LSTAT"]
stat, p = ttest_ind(data1, data2)
print('stat=%.3f, p=%.3f' % (stat, p))
if p > 0.05:
    print('Probably the same distribution')
else:
    print('Probably different distributions')
```

```
====================
  Student's t-test
====================
stat=1.955, p=0.051
Probably the same distribution
```

- El resultado del test indica que las funciones de distribución de X_train y X_test de la variable LSTAT siguen la misma funciín de distribución

E.3) Mann-Whitney U Test

- La U de Mann-Whitney (también llamada de Mann-Whitney-Wilcoxon, prueba de suma de rangos Wilcoxon, o prueba de Wilcoxon-Mann-Whitney) es una prueba no paramétrica aplicada a dos muestras independientes.
- Es la versión no paramétrica de la habitual prueba t de Student y se usa para comprobar la heterogeneidad de dos muestras ordinales.
- Vamos a comparar las distribuciondes de la variable RM de las muestras X_train y X_test, para un p-value del 0,05%.

In [18]:
```
print("======================")
print("  Mann-Whitney U Test  ")
print("======================")
#from scipy.stats import mannwhitneyu

data1 = X_train["RM"]
data2 = X_test["RM"]
stat, p = mannwhitneyu(data1, data2)
print('stat=%.3f, p=%.3f' % (stat, p))
if p > 0.05:
    print('Probably the same distribution')
else:
    print('Probably different distributions')
```

```
======================
  Mann-Whitney U Test
======================
```

```
stat=25399.000, p=0.060
Probably the same distribution
```

- El resultado del test indica que las funciones de distribución de X_train y X_test de la variable RM siguen la misma funciín de distribución

F) Escalado de la información y Elección de los Modelos

- El escalado se realiza para normalizar los datos de forma que no se dé prioridad a una característica concreta. El papel del escalado es muy importante en los algoritmos basados en la distancia y que requieren la distancia euclidiana.
- Sin empargo escalado, no es necesario para los modelos basados en árboles aleatorios y dado que los dos modelos que vamos a comparar son Decision Tree Regresor y Randon Forest Regressor el escalado no aporta más precisión al resultado de las métricas.
- Diferencias entre Árboles de decisión y modelos lineales:
    1. Si la relación entre la variable dependiente y la(s) independiente se aproxima a un modelo lineal, la regresión lineal dará mejores resultados que un modelo de árbol de decisión.
    2. Si la relación es compleja y altamente no lineal, entonces el árbol de decisión tendrá mejores resultados de que un método clásico de regresión.
    3. Si se quiere construir un modelo que sea fácil de explicar, entonces un modelo de árbol de decisión será mejor que un modelo lineal.
- En nuestro caso la variable objetivo MEDV y las variables explicativas tiene una relación compleja que se detalla en el estudio de Harrison, D. and Rubinfeld, D.L. - Hedonic prices and the demand for clean air, J. Environ. Economics & Management, vol.5, 81-102, 1978.-

G) Ajuste del modelo 1:

G.1) Modelo 1: Árboles de regresión

La clase DecisionTreeRegressor del módulo sklearn.tree permite entrenar árboles de decisión para problemas de regresión. Se ha ajustado un árbol de regresión empleando como variable respuesta MEDV y como predictores el resto de variables disponibles.

G.1.1) Creación y entrenamiento del Modelo 1

In [19]:
```
# Creación del modelo
# ------------------------------------------------------------------------------
modelo1 = DecisionTreeRegressor(max_depth= 4,criterion="poisson", random_state = 123

# Entrenamiento del modelo
# ------------------------------------------------------------------------------
modelo1.fit(X_train, y_train)
```

Out[19]:
```
DecisionTreeRegressor
DecisionTreeRegressor(criterion='poisson', max_depth=4, random_state=123)
```

In [20]:
```
# Error del modelo con train
#-------------------------------------------------------------------------------
y_pred = modelo1.predict(X = X_train)
```

In [21]:
```
# Evaluación del modelo con train
#-------------------------------------------------------------------------------
mse_modelo1_train= metrics.mean_squared_error(y_train, y_pred)
rmse_modelo1_train= np.sqrt(metrics.mean_squared_error(y_train, y_pred))
rdt2_modelo1_train= metrics.r2_score(y_train, y_pred)
rdt2_modelo1_adj_train= 1 - (1-metrics.r2_score(y_train, y_pred))*(len(y_train)-1)/(

print('MAE:',metrics.mean_absolute_error(y_train, y_pred))
print('MSE:',mse_modelo1_train)
print('RMSE:',rmse_modelo1_train,)
print('R^2:',rdt2_modelo1_train)
print('Adjusted R^2:',rdt2_modelo1_adj_train)
```

```
MAE: 2.2757502980583113
MSE: 9.262828213941626
RMSE: 3.0434894798473717
R^2: 0.8876539843050414
Adjusted R^2: 0.8802115037700441
```

In [22]:
```
model_rdos_train =np.empty(shape=(0,4), dtype=float)
model_rdos_train = np.append(model_rdos_train ,[[mse_modelo1_train,rmse_modelo1_trai
```

G.1.2) Error de test del modelo 1

In [23]:
```
# Error de test del modelo
#-------------------------------------------------------------------------------
y_test_pred = modelo1.predict(X = X_test)
```

In [24]:
```
# Evaluación del Modelo
# ==============================================================================
print('MAE:',metrics.mean_absolute_error(y_test, y_test_pred))
mse_modelo1=metrics.mean_squared_error(y_test, y_test_pred)
print('MSE:',mse_modelo1)
rmse_modelo1=np.sqrt(mse_modelo1)
print('RMSE:',rmse_modelo1)
rdt2_modelo1= metrics.r2_score(y_test, y_test_pred)
print('R^2:', rdt2_modelo1)
rdt2_modelo1_adj = 1 - (1-rdt2_modelo1)*(len(y_test)-1)/(len(y_test)-X_test.shape[1]
print('Adjusted R^2:',rdt2_modelo1_adj)
```

```
MAE: 3.0352289044349186
MSE: 17.007864250526865
RMSE: 4.124059195807798
R^2: 0.8068533996424487
Adjusted R^2: 0.7788804437285965
```

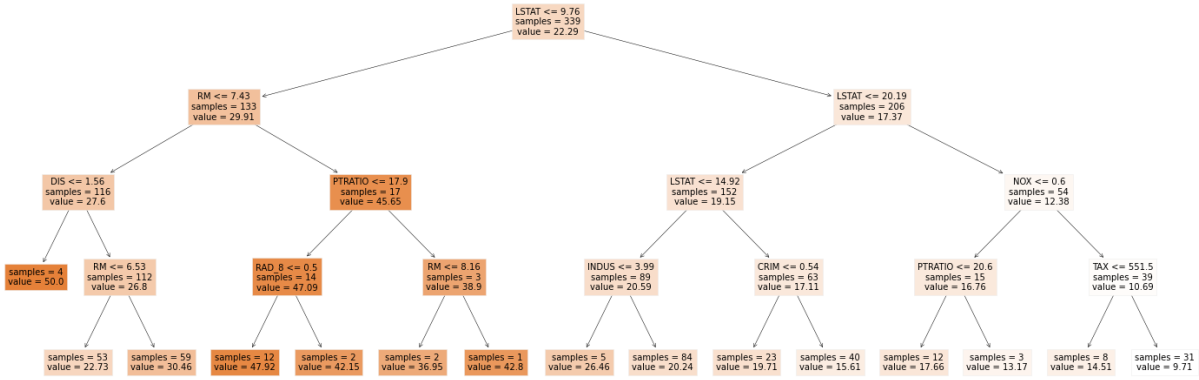
In [25]:
```
model_rdos=np.empty(shape=(0,4), dtype=float)
model_rdos= np.append(model_rdos,[[mse_modelo1,rmse_modelo1,rdt2_modelo1,rdt2_modelo
```

G.1.3) Gráficos de modelo 1

In [26]:
```
# Estructura del árbol creado
# ------------------------------------------------------------------------------
fig, ax = plt.subplots(figsize=(25, 9))

print(f"Profundidad del árbol: {modelo1.get_depth()}")
print(f"Número de nodos terminales: {modelo1.get_n_leaves()}")

plot = plot_tree(
```

```
            decision_tree = modelo1,
            feature_names = df.drop(columns = "MEDV").columns,
            class_names   = 'MEDV',
            filled        = True,
            impurity      = False,
            fontsize      = 10,
            precision     = 2,
            ax            = ax
        )
```

```
Profundidad del árbol: 4
Número de nodos terminales: 15
```



G.1.4) Parámetros de los nodos del modelo 1

In [27]:

```
texto_modelo = export_text(
                    decision_tree = modelo1,
                    feature_names = list(df.drop(columns = "MEDV").columns)
                )
print(texto_modelo)
```

```
|--- LSTAT <= 9.76
|   |--- RM <= 7.43
|   |   |--- DIS <= 1.56
|   |   |   |--- value: [50.00]
|   |   |--- DIS >  1.56
|   |   |   |--- RM <= 6.53
|   |   |   |   |--- value: [22.73]
|   |   |   |--- RM >  6.53
|   |   |   |   |--- value: [30.46]
|   |--- RM >  7.43
|   |   |--- PTRATIO <= 17.90
|   |   |   |--- RAD_8 <= 0.50
|   |   |   |   |--- value: [47.92]
|   |   |   |--- RAD_8 >  0.50
|   |   |   |   |--- value: [42.15]
|   |   |--- PTRATIO >  17.90
|   |   |   |--- RM <= 8.16
|   |   |   |   |--- value: [36.95]
|   |   |   |--- RM >  8.16
|   |   |   |   |--- value: [42.80]
|--- LSTAT >  9.76
|   |--- LSTAT <= 20.19
|   |   |--- LSTAT <= 14.92
|   |   |   |--- INDUS <= 3.99
|   |   |   |   |--- value: [26.46]
|   |   |   |--- INDUS >  3.99
|   |   |   |   |--- value: [20.24]
|   |   |--- LSTAT >  14.92
|   |   |   |--- CRIM <= 0.54
```

```
|   |   |   |   |--- value: [19.71]
|   |   |   |--- CRIM >  0.54
|   |   |   |   |--- value: [15.61]
|   |--- LSTAT >  20.19
|   |   |--- NOX <= 0.60
|   |   |   |--- PTRATIO <= 20.60
|   |   |   |   |--- value: [17.66]
|   |   |   |--- PTRATIO >  20.60
|   |   |   |   |--- value: [13.17]
|   |   |--- NOX >  0.60
|   |   |   |--- TAX <= 551.50
|   |   |   |   |--- value: [14.51]
|   |   |   |--- TAX >  551.50
|   |   |   |   |--- value: [9.71]
```

- El modelo predice un valor promedio de MEDV de 50.000 dólares para viviendas que están en una zona con un LSTAT <= 9.76, un RM<= 7.43 y un DIS <= 1.56
- El rmse de test implica que las predicciones del modelo se alejan en promedio 4.124 dólares del valor real

G.1.5) Importancia de los predictores

In [28]:
```
importancia_predictores = pd.DataFrame(
                            {'predictor': df.drop(columns = "MEDV").columns,
                             'importancia': modelo1.feature_importances_}
                            )
print("Importancia de los predictores en el modelo")
print("-------------------------------------------")
importancia_predictores.sort_values('importancia', ascending=False)
```

```
Importancia de los predictores en el modelo
-------------------------------------------
```

Out[28]:

	predictor	importancia
11	LSTAT	0.670051
5	RM	0.199536
7	DIS	0.059288
4	NOX	0.029380
0	CRIM	0.013650
8	TAX	0.012320
2	INDUS	0.007919
9	PTRATIO	0.006645
19	RAD_8	0.001211
14	RAD_3	0.000000
18	RAD_7	0.000000
17	RAD_6	0.000000
16	RAD_5	0.000000
15	RAD_4	0.000000
10	B	0.000000

	predictor	importancia
13	RAD_2	0.000000
12	RAD_1	0.000000
1	ZN	0.000000
6	AGE	0.000000
3	CHAS	0.000000
20	RAD_24	0.000000

- El predictor LSTAT, que es un índice en porcentaje de las personas en situación de pobreza, ha resultado se en este caso ser el predictor más importante del modelo, seguido de RM que es el número medio de habitaciones en las viviendas de la zona.

H) Ajuste del modelo 2

Se ajusta un modelo empleando como variable respuesta MEDV y como predictores todas las otras variables disponibles. La clase RandomForestRegressor del módulo sklearn.ensemble permite entrenar modelos random forest para problemas de regresión

H.1.1) Creación y entrenamiento del Modelo 2

In [29]:
```
# Creación del modelo
# ==============================================================================
modelo2 = RandomForestRegressor(random_state=123)

# Entrenamiento del modelo
# ==============================================================================
modelo2.fit(X_train, y_train)
```

Out[29]:
```
RandomForestRegressor
RandomForestRegressor(random_state=123)
```

In [30]:
```
# Model prediction on train data
y_pred = modelo2.predict(X_train)
```

In [31]:
```
# Evaluación del modelo con train
#-------------------------------------------------------------------------------
mse_modelo2_train= metrics.mean_squared_error(y_train, y_pred)
rmse_modelo2_train= np.sqrt(metrics.mean_squared_error(y_train, y_pred))
rdt2_modelo2_train= metrics.r2_score(y_train, y_pred)
rdt2_modelo2_adj_train= 1 - (1-metrics.r2_score(y_train, y_pred))*(len(y_train)-1)/(

print('MAE:',metrics.mean_absolute_error(y_train, y_pred))
print('MSE:',mse_modelo2_train)
print('RMSE:',rmse_modelo2_train,)
print('R^2:',rdt2_modelo2_train)
print('Adjusted R^2:',rdt2_modelo2_adj_train)
```

```
MAE: 0.8532241887905595
MSE: 1.5780336312684342
RMSE: 1.2561980859993516
```

```
R^2: 0.9808605118209123
Adjusted R^2: 0.9795925962002157
```

In [32]:
```
model_rdos_train = np.append(model_rdos_train ,[[mse_modelo2_train,rmse_modelo2_trai
```

H.1.2) Error de test del modelo 2

In [33]:
```
# Predicciones con Test data
# ==============================================================================
y_test_pred = modelo2.predict(X_test)
```

In [34]:
```
# Evaluación del Modelo
# ==============================================================================
print('MAE:',metrics.mean_absolute_error(y_test, y_test_pred))
mse_modelo2=metrics.mean_squared_error(y_test, y_test_pred)
print('MSE:',mse_modelo2)
rmse_modelo2=np.sqrt(mse_modelo2)
print('RMSE:',rmse_modelo2)
rdt2_modelo2= metrics.r2_score(y_test, y_test_pred)
print('R^2:',rdt2_modelo2 )
rdt2_modelo2_adj = 1 - (1-rdt2_modelo2)*(len(y_test)-1)/(len(y_test)-X_test.shape[1]
print('Adjusted R^2:',rdt2_modelo2_adj)
```

```
MAE: 2.2382275449101794
MSE: 9.695962766467066
RMSE: 3.1138340942425087
R^2: 0.8898896288240014
Adjusted R^2: 0.8739426095502361
```

In [35]:
```
model_rdos= np.append(model_rdos,[[mse_modelo2,rmse_modelo2,rdt2_modelo2,rdt2_modelo
```

- En este caso el modelo Randon Forest Regressor tiene un rmse de test implica que las predicciones del modelo se alejan en promedio 3.114 dólares del valor real, y es inferior al del modelo 1.

H.1.3) Importancia de los predictores

In [36]:
```
importancia_predictores = pd.DataFrame(
                            {'predictor': df.drop(columns = "MEDV").columns,
                             'importancia': modelo2.feature_importances_}
                            )
print("Importancia de los predictores en el modelo")
print("-------------------------------------------")
importancia_predictores.sort_values('importancia', ascending=False)
```

```
Importancia de los predictores en el modelo
-------------------------------------------
```

Out[36]:

	predictor	importancia
11	LSTAT	0.595458
5	RM	0.197957
7	DIS	0.066939
0	CRIM	0.038028
4	NOX	0.023147

	predictor	importancia
6	AGE	0.020468
9	PTRATIO	0.014740
10	B	0.014207
8	TAX	0.011835
2	INDUS	0.009619
16	RAD_5	0.001119
1	ZN	0.001065
3	CHAS	0.000991
12	RAD_1	0.000980
19	RAD_8	0.000762
20	RAD_24	0.000693
13	RAD_2	0.000512
15	RAD_4	0.000471
14	RAD_3	0.000394
17	RAD_6	0.000358
18	RAD_7	0.000255

- En cuanto a la importancia de los predictores se mantienen en los primeros lugares los mismos que en el modelo anterior, pero en este caso Randon Forest Regressor sí pondera y utiliza todas las variables, por lo que asigna valor a todos los predictores.

Exercici 2

Compara'ls en base al MSE i al R2.

A) Definición y significado de MSE

- La métrica más comúnmente utilizada para las tareas de regresión es el error cuadrático medio (MSE) y representa a la raíz cuadrada de la distancia al cuadrado del promedio entre el valor real y el valor pronosticado.
- Indica el ajuste absoluto del modelo a los datos, cuán cerca están los puntos de datos observados de los valores predichos por el modelo. El error cuadratico medio o MSE es una medida absoluta de ajuste.
- RMSE (Root Mean Squared Error) o raíz cuadrada del error cuadrático medio (MSE), se puede interpretar como la desviación estándar de la varianza inexplicada, y tiene la propiedad de estar en las mismas unidades que la variable de respuesta.
- Cuanto más bajos más bajos son los valores de MSE y RMSE mejor es el ajuste del modelo.
- MSE y RMSE son buenas medidas de la precisión con que el modelo realiza una predicción, y suelen ser el criterio más importante para ajustar si el propósito principal del modelo es la predicción.

B) Definición y significado de R2

- El coeficiente de determinación es la proporción de la varianza total de la variable explicada por la regresión. El coeficiente de determinación, también llamado R cuadrado, refleja la bondad del ajuste de un modelo a la variable que pretender explicar.
- El resultado del coeficiente de determinación oscila entre 0 y 1. Cuanto más cerca de 1 se sitúe su valor, mayor será el ajuste del modelo a la variable que estamos intentando explicar. De forma inversa, cuanto más cerca de cero, menos ajustado estará el modelo y, por tanto, menos fiable será
- El problema del coeficiente de determinación, y razón por el cual surge el coeficiente de determinación ajustado, radica en que no penaliza la inclusión de variables explicativas no significativas.
- El coeficiente de determinación ajustado (R cuadrado ajustado) es la medida que define el porcentaje explicado por la varianza de la regresión en relación con la varianza de la variable explicada, por tanto penaliza la inclusión de nuevas variables.

C) Sumario de Metricas

C.1) Métricas sobre Train

In [37]:
```
row_indices = ["Modelo 1: Decision Tree Regressor", "Modelo 2: Random Forest Regress
column_names = ["MSE","RMSE", "R^2","R^2 Ajustado"]
resultados = pd.DataFrame(model_rdos_train, index=row_indices,columns=column_names)
resultados
```

Out[37]:

	MSE	RMSE	R^2	R^2 Ajustado
Modelo 1: Decision Tree Regressor	9.262828	3.043489	0.887654	0.880212
Modelo 2: Random Forest Regressor	1.578034	1.256198	0.980861	0.979593

- El modelo 1 Decision Tree Regressor, tiene unos valores superiores de RMSE e inferiores de R^2 al modelo 2, cuando aplicamos la predccion sobre train.
- El RMSE es de 3.043 dólares y se corresponde con la desviadcion en promedio de las preciciones en relación al valor real de la variable objetivo MEDV (valor de las viviendas en Boston), mientras que el ajuste del modelo R^2 es el 88,76%
- El modelo 2 Random Forest Regressor, aporta mejores métricas tanto en RMSE como en R^2, con una desviación media del valor de las preccciones de 1.256 dólares y un nivel de ajuste del 98,08%

C.2) Métricas sobre Test

In [38]:
```
row_indices = ["Modelo 1: Decision Tree Regressor", "Modelo 2: Random Forest Regress
column_names = ["MSE","RMSE", "R^2","R^2 Ajustado"]
resultados = pd.DataFrame(model_rdos, index=row_indices,columns=column_names)
resultados
```

Out[38]:

	MSE	RMSE	R^2	R^2 Ajustado
Modelo 1: Decision Tree Regressor	17.007864	4.124059	0.806853	0.778880
Modelo 2: Random Forest Regressor	9.695963	3.113834	0.889890	0.873943

- El modelo 1 Decision Tree Regressor, tiene unos valores superiores de RMSE e inferiores de

R^2 al modelo 2, cuando aplicamos la predccion sobre test.

- El RMSE es de 4.124 dólares y es la desviadcion en promedio de las preciciones en relación al valor real de la variable objetivo MEDV, mientras que el ajuste del modelo R^2 es del 80,68%
- El modelo 2 Random Forest Regressor, aporta mejores métricas tanto en RMSE como en R^2, con una desviación media del valor de las preccciones de 3.113 dólares y un novel de ajuste del 88,98%

Exercici 3

Entrena'ls utilitzant els diferents paràmetres que admeten per intentar millorar-ne la predicció.

3.1. Modificación parámetros del modelo DecisionTreeRegressor - modelo 3

In [39]:
```
DecisionTreeRegressor().get_params().keys()
```

Out[39]:
```
dict_keys(['ccp_alpha', 'criterion', 'max_depth', 'max_features', 'max_leaf_nodes',
'min_impurity_decrease', 'min_samples_leaf', 'min_samples_split', 'min_weight_fracti
on_leaf', 'random_state', 'splitter'])
```

In [40]:
```
# Grid de hiperparámetros evaluados
# ==============================================================================
param_grid = {'criterion': ["squared_error", "friedman_mse", "absolute_error", "pois
              'max_features': ["auto", "sqrt", "log2", None],
              'splitter': ["best", "random"]
             }

# cross validator
# ==============================================================================
cv= RepeatedKFold(n_splits=10, n_repeats=3, random_state=123)

# Búsqueda por grid search con validación cruzada
# ==============================================================================
grid = GridSearchCV(
        estimator  = DecisionTreeRegressor(max_depth= 4,random_state = 123),
        param_grid = param_grid,
        scoring    = 'neg_median_absolute_error',
        n_jobs     = multiprocessing.cpu_count() - 1,
        cv         = cv,
        refit      = True,
        verbose    = 0,
        return_train_score = True
       )

result= grid.fit(X = X_train, y = y_train)

print('Best Hyperparameters: ', result.best_params_)
```

```
Best Hyperparameters:  {'criterion': 'absolute_error', 'max_features': 'auto', 'spli
tter': 'best'}
```

In [41]:
```
# # Repetición del modelo con Best Hyperparameters
# ------------------------------------------------------------------------------
modelo3 = DecisionTreeRegressor(max_depth= 4,random_state = 123, criterion="absolute

# Entrenamiento del modelo
```

```
# ------------------------------------------------------------------------
modelo3.fit(X_train, y_train)
```

Out[41]:

```
DecisionTreeRegressor(criterion='absolute_error', max_depth=4,
                      max_features='auto', random_state=123)
```

In [42]:

```
# Error de test del modelo
#-------------------------------------------------------------------------------
y_predicciones = modelo3.predict(X = X_test)

print('MAE:', metrics.mean_absolute_error(y_test, y_predicciones))

mse_modelo3=metrics.mean_squared_error(y_test, y_predicciones)
print('MSE:',mse_modelo3)

rmse_modelo3=np.sqrt(mse_modelo3)
print('RMSE:',rmse_modelo3)

#R2 for the decision tree regression model
rdt2_modelo3= metrics.r2_score(y_test, y_predicciones)
print('R^2:', rdt2_modelo3)

rdt2_modelo3_adj = 1 - (1-rdt2_modelo3)*(len(y_test)-1)/(len(y_test)-X_test.shape[1]
print('Adjusted R^2:',rdt2_modelo3_adj)
```

```
MAE: 2.9970059880239512
MSE: 19.59494011976048
RMSE: 4.426617232126636
R^2: 0.7774737608089574
Adjusted R^2: 0.7452458227192202
```

In [43]:

```
model_rdos= np.append(model_rdos,[[mse_modelo3,rmse_modelo3,rdt2_modelo3,rdt2_modelo
```

- Con esta combinación de parámetros sobre el modelo de Decision Tree Regressor las métricas de RMSE y R^2 empeoran respecto al modelo 1

3.1.2. Modificación parámetros del modelo DecisionTreeRegressor - modelo 4

In [47]:

```
grid = GridSearchCV(DecisionTreeRegressor(), param_grid, scoring = 'neg_mean_squared
result = grid.fit(X_train, y_train)

print('Best Hyperparameters: ', result.best_params_)
```

```
Best Hyperparameters:  {'criterion': 'squared_error', 'max_features': 'auto', 'split
ter': 'best'}
```

In [48]:

```
# Creación del modelo
# ------------------------------------------------------------------------
modelo4 = DecisionTreeRegressor(max_depth= 4,random_state = 123, criterion="squared_

# Entrenamiento del modelo
# ------------------------------------------------------------------------
modelo4.fit(X_train, y_train)
```

Out[48]:

DecisionTreeRegressor

DecisionTreeRegressor(max_depth=4, random_state=123)

In [49]:
```
# Error de test del modelo
#------------------------------------------------------------------------------
y_predicciones = modelo4.predict(X = X_test)

print('MAE:', metrics.mean_absolute_error(y_test, y_predicciones))


mse_modelo4=metrics.mean_squared_error(y_test, y_predicciones)
print('MSE:',mse_modelo4)

rmse_modelo4=np.sqrt(mse_modelo4)
print('RMSE:',rmse_modelo4)

#R2 for the decision tree regression model
rdt2_modelo4= metrics.r2_score(y_test, y_predicciones )
print('R^2:', rdt2_modelo4)

#R2 Ajusted for the decision tree regression model
rdt2_modelo4_adj = 1 - (1-rdt2_modelo4)*(len(y_test)-1)/(len(y_test)-X_test.shape[1]
print('Adjusted R^2:',rdt2_modelo4_adj)
```

```
MAE: 2.849673566949767
MSE: 15.074854351517601
RMSE: 3.8826349753122043
R^2: 0.8288052617311626
Adjusted R^2: 0.8040115410163655
```

In [50]:
```
model_rdos= np.append(model_rdos,[[mse_modelo4,rmse_modelo4,rdt2_modelo4,rdt2_modelo
```

3.2. Modificación parámetros del modelo RandomForestRegressor - modelo 5

In [51]:
```
RandomForestRegressor().get_params().keys()
```

Out[51]:
```
dict_keys(['bootstrap', 'ccp_alpha', 'criterion', 'max_depth', 'max_features', 'max_
leaf_nodes', 'max_samples', 'min_impurity_decrease', 'min_samples_leaf', 'min_sample
s_split', 'min_weight_fraction_leaf', 'n_estimators', 'n_jobs', 'oob_score', 'random
_state', 'verbose', 'warm_start'])
```

In [52]:
```
modelo_rf = RandomForestRegressor(random_state = 123)
print('Parameters currently in use:\n')
pprint(modelo_rf.get_params())
```

```
Parameters currently in use:

{'bootstrap': True,
 'ccp_alpha': 0.0,
 'criterion': 'squared_error',
 'max_depth': None,
 'max_features': 1.0,
 'max_leaf_nodes': None,
 'max_samples': None,
 'min_impurity_decrease': 0.0,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'min_weight_fraction_leaf': 0.0,
```

```
'n_estimators': 100,
'n_jobs': None,
'oob_score': False,
'random_state': 123,
'verbose': 0,
'warm_start': False}
```

In [53]:

```
# Grid de hiperparámetros evaluados
# ==============================================================================
param_grid = {'n_estimators': [150],
              'max_features': [5, 7, 9],
              'max_depth'   : [None, 3, 10, 20]
             }
# cross validator
# ==============================================================================
cv= RepeatedKFold(n_splits=10, n_repeats=3, random_state=123)

# Búsqueda por grid search con validación cruzada
# ==============================================================================
grid = GridSearchCV(
        estimator  = RandomForestRegressor(random_state = 123),
        param_grid = param_grid,
        scoring    = 'neg_mean_squared_error',
        n_jobs     = multiprocessing.cpu_count() - 1,
        cv         = cv,
        refit      = True,
        verbose    = 0,
        return_train_score = True
       )

result= grid.fit(X = X_train, y = y_train)

print('Best Hyperparameters: ', result.best_params_)
```

```
Best Hyperparameters:  {'max_depth': 20, 'max_features': 9, 'n_estimators': 150}
```

In [54]:

```
# Repetición del modelo con Best Hyperparameters
# ==============================================================================
modelo5 = RandomForestRegressor(max_depth= 20, max_features= 9, n_estimators= 150,ra

# Entrenamiento del modelo
# ==============================================================================
modelo5.fit(X_train, y_train)
```

Out[54]:

```
RandomForestRegressor

RandomForestRegressor(max_depth=20, max_features=9, n_estimators=150,
                      random_state=123)
```

In [55]:

```
# Predicciones
# ==============================================================================
y_predicciones = modelo5.predict(X_test)

# Evaluación de las predicciones
# ==============================================================================

print('MAE:', metrics.mean_absolute_error(y_test, y_predicciones))

mse_modelo5=metrics.mean_squared_error(y_test, y_predicciones)
print('MSE:',mse_modelo5)
```

```
rmse_modelo5=np.sqrt(mse_modelo5)
print('RMSE:',rmse_modelo5)

#R2 for the Randon Forest Regression model
rdt2_modelo5= metrics.r2_score(y_test, y_predicciones )
print('R^2:', rdt2_modelo5)

#R2 Ajusted for the Randon Forest Regression model
rdt2_modelo5_adj = 1 - (1-rdt2_modelo5)*(len(y_test)-1)/(len(y_test)-X_test.shape[1]
print('Adjusted R^2:',rdt2_modelo5_adj)
```

```
MAE: 2.248965529546966
MSE: 9.143771699685777
RMSE: 3.0238670109126455
R^2: 0.8961604824553332
Adjusted R^2: 0.8811216557764504
```

In [56]:
```
model_rdos= np.append(model_rdos,[[mse_modelo5,rmse_modelo5,rdt2_modelo5,rdt2_modelo
```

C) Sumary de resultados sobre test

In [57]:
```
row_indices = ["M_1: Decision Tree Regressor", "M_2: Random Forest Regressor","M_3:
column_names = ["MSE", "RMSE", "R^2","R^2 Ajustado"]
resultados = pd.DataFrame(model_rdos, index=row_indices,columns=column_names)
resultados.sort_values(by="MSE")
```

Out[57]:

	MSE	RMSE	R^2	R^2 Ajustado
M_5: RandomForestRegressor	9.143772	3.023867	0.896160	0.881122
M_2: Random Forest Regressor	9.695963	3.113834	0.889890	0.873943
M_4: DecisionTreeRegressor v2	15.074854	3.882635	0.828805	0.804012
M_1: Decision Tree Regressor	17.007864	4.124059	0.806853	0.778880
M_3: DecisionTreeRegressor v1	19.594940	4.426617	0.777474	0.745246

- El modelo que tiene mejores métricas de MSE, RMSE y R^2 es el modelo 5 Random Forest REgressor con los parámetros optimizados, con un RMSE de 3.023 dólares y un R^" del 89,61%
- En todos los casos evaluados el Modelo Randon Fores proporciona mejores métricas que el Decision Tree Reressor.

Exercici 4

Compara el seu rendiment emprant l'aproximació traint/test o emprant totes les dades (validació interna).

No existe un método de validación que supere al resto en todos los escenarios, la elección debe basarse en varios factores.

- Si el tamaño de la muestra es pequeño, se recomienda emplear repeated k-Fold-Cross-Validation, ya que consigue un buen equilibrio bias-varianza y, dado que no son muchas observaciones, el coste computacional no es excesivo.

- Si el objetivo principal es comparar modelos mas que obtener una estimación precisa de las métricas, se recomienda bootstrapping ya que tiene menos varianza.
- Si el tamaño muestral es muy grande, la diferencia entre métodos se reduce y toma más importancia la eficiencia computacional. En estos casos, 10-Fold-Cross-Validation simple es suficiente.

4.1. Validación cruzada del modelo DecisionTreeRegressor - modelo 4

In [58]:
```
X_todos= df.drop(columns = "MEDV")
y_todos=df['MEDV']
```

a) Métricas de la Validación Cruzada

In [59]:
```
# Validación cruzada
# ==============================================================================
cv = KFold(n_splits=10, random_state=123, shuffle=True)

modelo4_r2_scores = cross_val_score(modelo4, X_todos, y_todos, cv = cv, scoring="r2"
modelo4_r2_mean = np.mean(modelo4_r2_scores)
modelo4_mse_scores = cross_val_score(modelo4, X_todos, y_todos, cv = cv,scoring = "n
modelo4_mse_media= abs(np.mean(modelo4_mse_scores))
modelo4_rmse=np.sqrt(modelo4_mse_media)

print("1. MSE de cada n-splits: ", modelo4_mse_scores)
print("2. MSE media: ",modelo4_mse_media)
print("3. RMSE: ",modelo4_rmse)
print("4. R^2 de cada n_splits", modelo4_r2_scores)
print("5. R^2 media :" ,modelo4_r2_mean)
```

```
1. MSE de cada n-splits:  [-5.07675102 -2.11582928 -3.39685357 -3.52465817 -2.538455
58 -2.76811448
 -2.53979365 -2.44844316 -2.58481892 -2.98084804]
2. MSE media:  2.997456587102096
3. RMSE:  1.7313164318235115
4. R^2 de cada n_splits [0.26972536 0.84200914 0.67575511 0.7783927  0.87789799 0.85
749317
 0.87129532 0.81322367 0.8801674  0.79580763]
5. R^2 media : 0.7661767488306064
```

In [60]:
```
model_rdos_cruzados=np.empty(shape=(0,3), dtype=float)
model_rdos_cruzados= np.append(model_rdos_cruzados,[[modelo4_mse_media,modelo4_rmse,
```

b) Gráficos - Diagnóstico errores (residuos) de las predicciones de validación cruzada

In [61]:
```
# Gráficos :Diagnóstico errores (residuos) de las predicciones de validación cruzada
# ==============================================================================

cv_predicciones = cross_val_predict(estimator = modelo4, X = X_todos, y = y_todos, c

fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(14, 9))

axes[0, 0].scatter(y_todos, cv_predicciones, edgecolors=(1, 1, 1), alpha = 0.4)
axes[0, 0].plot([y_todos.min(), y_todos.max()], [y_todos.min(), y_todos.max()],'k--'
axes[0, 0].set_title('Valor predicho vs valor real', fontsize = 12, fontweight = "no
axes[0, 0].set_xlabel('Real')
axes[0, 0].set_ylabel('Predicción')
axes[0, 0].tick_params(labelsize = 9)
```

```
axes[0, 1].scatter(cv_predicciones, y_todos - cv_predicciones, edgecolors=(1, 1, 1),
axes[0, 1].axhline(y = 0, linestyle = '--', color = 'black', lw=2)
axes[0, 1].set_title('Residuos del modelo4', fontsize = 12, fontweight = "normal")
axes[0, 1].set_xlabel('Predicciones')
axes[0, 1].set_ylabel('Residuo')
axes[0, 1].tick_params(labelsize = 9)

sns.histplot(data  = y_todos - cv_predicciones, stat  = "density", kde = True, line_

axes[1, 0].set_title('Distribución residuos del modelo4', fontsize = 12, fontweight
axes[1, 0].set_xlabel("Residuo")
axes[1, 0].tick_params(labelsize = 9)

sm.qqplot( y_todos - cv_predicciones, fit = True, line = 'q', ax = axes[1, 1], color
axes[1, 1].set_title('Q-Q residuos del modelo4', fontsize = 12, fontweight = "normal
axes[1, 1].tick_params(labelsize = 9)

fig.tight_layout()
plt.subplots_adjust(top=0.85)
fig.suptitle('Diagnóstico residuos DecisionTreeRegressor', fontsize = 14, fontweight
```

Out[61]: Text(0.5, 0.98, 'Diagnóstico residuos DecisionTreeRegressor')

- El análisis gráfico de los errores (residuos) de las precciones sobre la validación cruzada reflejan que la función de distribución se aproxima a la normal
- Sin embargo, el gráfico de residuos vs. predicciones se observa que los errores son más grades cuanto mayor el el valor de la predicción de la variable objetivo.
- Este mimo efecto se observa en la gráfica qqplot donde las colas de los valores inferiores y superiores se alejan de los valores teóricos.
- Como conclusión el modelo se ajusta bien en los valores centrales de la variable objetivo, pero pierde precisión en los valores más extremos, tienen problemas de dispersión irregular.

- En todos los casos la varianza de los residuos aumenta con los valores ajustados, esto indica que la variabilidad de los errores aumenta al aumentar su media y que es un tema que ya se detectó al analizar los outliers de la variable objetivo MEDV.
- Para mejorar estos resultados se podría utilizar pruebas de igualdad de varianza (complementarias a los análisis gráficos), considera utilizar transformaciones de las variables o modelar la heterogeneidad encontrada con modelos generalizados (GLM) o modelos mixtos (MM).

4.2. Validación cruzada del modelo RandomForestRegressor - modelo 5

a) Métricas de la Validación Cruzada

In [62]:
```
# Validación cruzada
# ============================================================================
cv = KFold(n_splits=10, random_state=123, shuffle=True)

modelo5_r2_scores = cross_val_score(modelo5, X_todos, y_todos, cv = cv,scoring="r2")
modelo5_r2_mean = np.mean(modelo5_r2_scores)
modelo5_mse_scores = cross_val_score(modelo5, X_todos, y_todos, cv = cv,scoring = "n
modelo5_mse_media= abs(np.mean(modelo5_mse_scores))
modelo5_rmse=np.sqrt(modelo5_mse_media)

print("1. MSE de cada n-splits: ", modelo5_mse_scores)
print("2. MSE media: ",modelo5_mse_media)
print("3. RMSE: ",modelo5_rmse)
print("4. R^2 de cada n_splits", modelo5_r2_scores)
print("5. R^2 media :" ,modelo5_r2_mean)
```

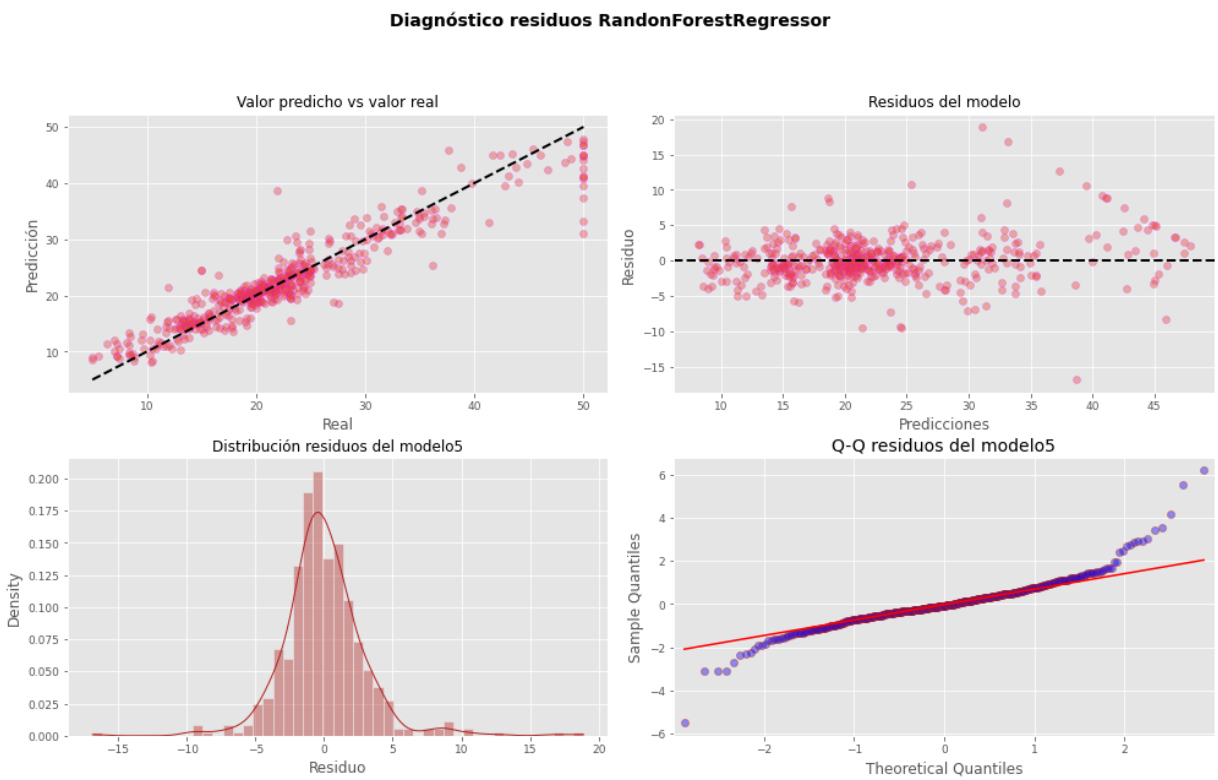
```
1. MSE de cada n-splits:  [-2.8704344  -1.49927015 -2.26853354 -2.48655241 -1.607214
19 -2.31470733
 -2.01709129 -1.62067946 -2.22029178 -1.90924541]
2. MSE media:  2.0814019959305696
3. RMSE:  1.4427064829446665
4. R^2 de cada n_splits [0.79975471 0.90198632 0.8810006  0.88085156 0.94696826 0.89
35899
 0.93050628 0.91295743 0.85526213 0.91034536]
5. R^2 media : 0.8913222554309324
```

In [63]:
```
model_rdos_cruzados= np.append(model_rdos_cruzados,[[modelo5_mse_media,modelo5_rmse,
```

b) Gráficos - Diagnóstico errores (residuos) de las predicciones de validación cruzada

In [64]:
```
# Gráficos: Diagnóstico errores (residuos) de las predicciones de validación cruzada
# ============================================================================
cv_predicciones = cross_val_predict(estimator = modelo5, X = X_todos, y = y_todos, c

fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(14, 9))

axes[0, 0].scatter(y_todos, cv_predicciones, edgecolors=(1, 0, 1), alpha = 0.4)
axes[0, 0].plot([y_todos.min(), y_todos.max()], [y_todos.min(), y_todos.max()],'k--'
axes[0, 0].set_title('Valor predicho vs valor real', fontsize = 12, fontweight = "no
axes[0, 0].set_xlabel('Real')
axes[0, 0].set_ylabel('Predicción')
axes[0, 0].tick_params(labelsize = 9)

axes[0, 1].scatter(cv_predicciones, y_todos - cv_predicciones, edgecolors=(1, 0, 1),
axes[0, 1].axhline(y = 0, linestyle = '--', color = 'black', lw=2)
axes[0, 1].set_title('Residuos del modelo', fontsize = 12, fontweight = "normal")
axes[0, 1].set_xlabel('Predicciones')
```

```
axes[0, 1].set_ylabel('Residuo')
axes[0, 1].tick_params(labelsize = 9)

sns.histplot(data = y_todos - cv_predicciones,stat = "density",kde  = True,line_kws=

axes[1, 0].set_title('Distribución residuos del modelo5', fontsize = 12,fontweight =
axes[1, 0].set_xlabel("Residuo")
axes[1, 0].tick_params(labelsize = 9)


sm.qqplot( y_todos - cv_predicciones, fit = True, line  = 'q', ax  = axes[1, 1], col
axes[1, 1].set_title('Q-Q residuos del modelo5', fontsize = 14, fontweight = "normal
axes[1, 1].tick_params(labelsize = 9)

fig.tight_layout()
plt.subplots_adjust(top=0.85)
fig.suptitle('Diagnóstico residuos RandonForestRegressor', fontsize = 14, fontweight
```



- En el modelo 5 Randon Forest Regressor, el análisis gráfico de los errores (residuos) de las precciones sobre la validación cruzada reflejan que la función de distribución se aproxima a la normal y con mayor precisión que el modelo anterior, al concentrar los errores en una rango de [-5, a 5], mientras que en el Modelo 4 Decision Tree Regressor el rango es de [-10 a 10] aproximadamente.
- El gráfico de residuos vs. predicciones se observa que los errores son más grades cuanto mayor el el valor de la predicción de la variable objetivo.
- Este mismo efecto se observa en la gráfica qqplot donde las colas de los valores inferiores y superiores se alejan de los valores teóricos.
- Como conclusión el modelo se ajusta bien en los valores centrales de la variable objetivo, pero pierde precisión en los valores más extremos, tienen problemas de dispersión irregular.
- En todos los casos la varianza de los residuos aumenta con los valores ajustados, esto indica que la variabilidad de los errores aumenta al aumentar su media como ya se preveía al analizar los outliers de la variable objetivo MEDV.

- Para mejorar estos resultados se podría utilizar pruebas de igualdad de varianza (complementarias a los análisis gráficos), considera utilizar transformaciones de las variables o modelar la heterogeneidad encontrada con modelos generalizados (GLM) o modelos mixtos (MM).

4.3 Sumario Méticas de Validación Cruzada

In [66]:
```
row_indices = ["Modelo 4: Decision Tree Regressor", "Modelo 5: Random Forest Regress
column_names = ["MSE","RMSE", "R^2"]
resultados1 = pd.DataFrame(model_rdos_cruzados, index=row_indices,columns=column_nam
resultados1.sort_values(by="MSE")
```

Out[66]:

	MSE	RMSE	R^2
Modelo 5: Random Forest Regressor	2.081402	1.442706	0.891322
Modelo 4: Decision Tree Regressor	2.997457	1.731316	0.766177

- La validación cruzada aporta una mejora significativa de las métricas en los dos modelos en relación a MSE y RMSE, pero se reduce en ambos casos el R^2 que es del 76,62% para el Modelo 4 Decision Tree Regressor, y del 89,13% para el modelo 5 Randon Fores REgressor.

Exercici 5

No facis servir la variable del nombre d'habitacions (RM) a l'hora de fer prediccions.

In [67]:
```
data.head()
```

Out[67]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	M
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296.0	15.3	396.90	4.98	
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	17.8	396.90	9.14	
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242.0	17.8	392.83	4.03	
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222.0	18.7	394.63	2.94	
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222.0	18.7	396.90	5.33	

In [68]:
```
df_sinRM= data.drop(columns = "RM")
df_sinRM.head()
```

Out[68]:

	CRIM	ZN	INDUS	CHAS	NOX	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
0	0.00632	18.0	2.31	0	0.538	65.2	4.0900	1	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	78.9	4.9671	2	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	61.1	4.9671	2	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	45.8	6.0622	3	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	54.2	6.0622	3	222.0	18.7	396.90	5.33	36.2

In [69]:
```
# División de los datos en train y test
# ==============================================================================
#X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_st
X_train, X_test, y_train, y_test = train_test_split(df_sinRM.drop(columns = "MEDV"),
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

```
(339, 12) (167, 12) (339,) (167,)
```

5.1. Modificación parámetros del modelo DecisionTreeRegressor - modelo 4_sinRM

In [70]:
```
# Creación del modelo
# ------------------------------------------------------------------------------
modelo4_sinRM = DecisionTreeRegressor(max_depth= 4,random_state = 123, criterion="fr

# Entrenamiento del modelo
# ------------------------------------------------------------------------------
modelo4_sinRM.fit(X_train, y_train)
```

Out[70]:
```
DecisionTreeRegressor(criterion='friedman_mse', max_depth=4,
                      max_features='auto', random_state=123)
```

In [71]:
```
# Error de test del modelo
#-------------------------------------------------------------------------------
y_predicciones = modelo4_sinRM.predict(X = X_test)

print('MAE:', metrics.mean_absolute_error(y_test, y_predicciones))

mse_modelo4_sinRM=metrics.mean_squared_error(y_test, y_predicciones)
print('MSE:',mse_modelo4_sinRM)

rmse_modelo4_sinRM=np.sqrt(mse_modelo4_sinRM)
print('RMSE:',rmse_modelo4_sinRM)

#R2 for the decision tree regression model
rdt2_modelo4_sinRM= metrics.r2_score(y_test, y_predicciones )
print('R^2:', rdt2_modelo4_sinRM)

rdt2_modelo4_adj_sinRM = 1 - (1-rdt2_modelo4_sinRM)*(len(y_test)-1)/(len(y_test)-X_t
print('Adjusted R^2:',rdt2_modelo4_adj_sinRM)
```

```
MAE: 3.85620726290491
MSE: 37.289280148307704
RMSE: 6.106494915113555
R^2: 0.5667005002789289
Adjusted R^2: 0.5329369028980663
```

In [72]:
```
model_rdos= np.append(model_rdos,[[mse_modelo4_sinRM,rmse_modelo4_sinRM,rdt2_modelo4
```

5.2. Modelo RandomForestRegressor - modelo 5_sinRM

In [73]:
```
# Repetición del modelo con Best Hyperparameters
# ==============================================================================
modelo5_sinRM = RandomForestRegressor(max_depth= 20, max_features= 9, n_estimators=

# Entrenamiento del modelo
```

```
# ============================================================================
modelo5_sinRM.fit(X_train, y_train)
```

Out[73]:

RandomForestRegressor

RandomForestRegressor(max_depth=20, max_features=9, n_estimators=150, random_state=123)

In [74]:

```
# Predicciones
# ============================================================================
y_predicciones = modelo5_sinRM.predict(X_test)

# Evaluación de las predicciones
# ============================================================================

print('MAE:', metrics.mean_absolute_error(y_test, y_predicciones))

mse_modelo5_sinRM=metrics.mean_squared_error(y_test, y_predicciones)
print('MSE:',mse_modelo5_sinRM)

rmse_modelo5_sinRM=np.sqrt(mse_modelo5_sinRM)
print('RMSE:',rmse_modelo5_sinRM)

#R2 for the decision tree regression model
rdt2_modelo5_sinRM = metrics.r2_score(y_test, y_predicciones )
print('R^2:', rdt2_modelo5_sinRM)

rdt2_modelo5_adj_sinRM = 1 - (1-rdt2_modelo5_sinRM)*(len(y_test)-1)/(len(y_test)-X_t
print('Adjusted R^2:',rdt2_modelo5_adj_sinRM)
```

```
MAE: 2.8993007318695914
MSE: 17.07185370924817
RMSE: 4.131809979808869
R^2: 0.8016259460598826
Adjusted R^2: 0.7861682275710422
```

In [75]:

```
model_rdos= np.append(model_rdos,[[mse_modelo5_sinRM,rmse_modelo5_sinRM,rdt2_modelo5
```

5.2.1) Importancia de los predictores

In [76]:

```
importancia_predictores = pd.DataFrame(
                            {'predictor': df_sinRM.drop(columns = "MEDV").columns,
                             'importancia': modelo5_sinRM.feature_importances_}
                            )
print("Importancia de los predictores en el modelo")
print("-------------------------------------------")
importancia_predictores.sort_values('importancia', ascending=False)
```

```
Importancia de los predictores en el modelo
-------------------------------------------
```

Out[76]:

	predictor	importancia
11	LSTAT	0.593981
9	PTRATIO	0.087651
6	DIS	0.073908
0	CRIM	0.066803

	predictor	importancia
2	INDUS	0.045457
4	NOX	0.038190
8	TAX	0.031569
5	AGE	0.028631
10	B	0.021851
7	RAD	0.006266
1	ZN	0.004079
3	CHAS	0.001613

5.3) Sumary de resultados

In [77]:
```
row_indices = ["M_1: Decision Tree Regressor", "M_2: Random Forest Regressor","M_3:
column_names = ["MSE","RMSE", "R^2","R^2 Ajustado"]
resultados = pd.DataFrame(model_rdos, index=row_indices,columns=column_names)
resultados.sort_values(by="MSE")
```

Out[77]:

	MSE	RMSE	R^2	R^2 Ajustado
M_5: RandomForestRegressor_v1	9.143772	3.023867	0.896160	0.881122
M_2: Random Forest Regressor	9.695963	3.113834	0.889890	0.873943
M_4: DecisionTreeRegressor v2	15.074854	3.882635	0.828805	0.804012
M_1: Decision Tree Regressor	17.007864	4.124059	0.806853	0.778880
M5_sinRM_RFR	17.071854	4.131810	0.801626	0.786168
M_3: DecisionTreeRegressor v1	19.594940	4.426617	0.777474	0.745246
M4_sinRM_DTR	37.289280	6.106495	0.566701	0.532937

- Cuando eliminamos la variable explicativa RM, las metricas de los dos modelos: Decision Tree Regressor y Randon Forest Regressor, empeoran en relación a todos los modelos anteriores, ya que es una de las variables que más importancia tiene a la hora de realizar la ponderación de los predictores en los árboles de decisión.

5.4) Validación cruzada

5.4.1) Modelo 4 Decision Tree Regressor

In [78]:
```
X_todos= df_sinRM.drop(columns = "MEDV")
y_todos=df_sinRM['MEDV']
```

a) Métricas de la Validación Cruzada

In [79]:
```
# Validación cruzada
# ============================================================================
cv = KFold(n_splits=10, random_state=123, shuffle=True)

modelo4s_r2_scores = cross_val_score(modelo4_sinRM, X_todos, y_todos, cv = cv, scori
modelo4s_r2_mean = np.mean(modelo4s_r2_scores)
```

```
modelo4s_mse_scores = cross_val_score(modelo4_sinRM, X_todos, y_todos, cv = cv,scori
modelo4s_mse_media= abs(np.mean(modelo4s_mse_scores))
modelo4s_rmse=np.sqrt(modelo4s_mse_media)

print("1. MSE de cada n-splits: ", modelo4s_mse_scores)
print("2. MSE media: ",modelo4s_mse_media)
print("3. RMSE: ",modelo4s_rmse)
print("4. R^2 de cada n_splits", modelo4s_r2_scores)
print("5. R^2 media :" ,modelo4s_r2_mean)
```

```
1. MSE de cada n-splits:  [-4.30430928 -3.58461024 -2.55328665 -3.68962726 -3.151581
82 -3.22242978
 -3.66296876 -3.55863539 -4.03090432 -3.88345317]
2. MSE media:  3.5641806669321823
3. RMSE:  1.8879037758668162
4. R^2 de cada n_splits [0.67635587 0.29741069 0.84004601 0.75201503 0.79754776 0.81
550644
 0.7106797  0.52337809 0.56810721 0.61637719]
5. R^2 media : 0.6597423986531866
```

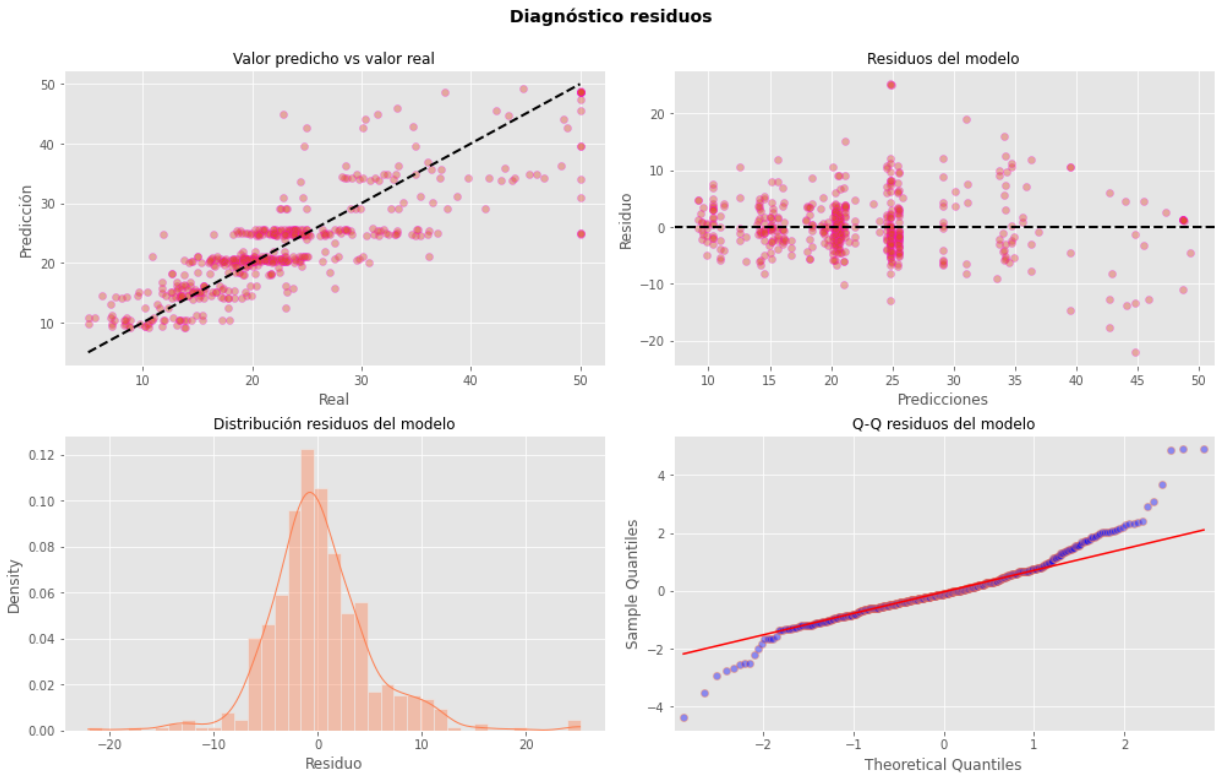
In [80]:

```
model_rdos_cruzados= np.append(model_rdos_cruzados,[[modelo4s_mse_media,modelo4s_rms
```

b) Gráficos - Diagnóstico errores (residuos) de las predicciones de validación cruzada

In [81]:

```
# Gráficos:Diagnóstico errores (residuos) de las predicciones de validación cruzada
# ==============================================================================
cv_predicciones = cross_val_predict(estimator = modelo4_sinRM, X = X_todos, y = y_to

fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(14, 9))

axes[0, 0].scatter(y_todos, cv_predicciones, edgecolors=(1, 0, 1), alpha = 0.4)
axes[0, 0].plot([y_todos.min(), y_todos.max()], [y_todos.min(), y_todos.max()],'k--'
axes[0, 0].set_title('Valor predicho vs valor real', fontsize = 12, fontweight = "no
axes[0, 0].set_xlabel('Real')
axes[0, 0].set_ylabel('Predicción')
axes[0, 0].tick_params(labelsize = 10)

#axes[0, 1].scatter(list(range(len(y_train))), y_train - cv_prediccones, edgecolors=
axes[0, 1].scatter(cv_predicciones, y_todos - cv_predicciones, edgecolors=(1, 0, 1),

axes[0, 1].axhline(y = 0, linestyle = '--', color = 'black', lw=2)
axes[0, 1].set_title('Residuos del modelo', fontsize = 12, fontweight = "normal")
axes[0, 1].set_xlabel('Predicciones')
axes[0, 1].set_ylabel('Residuo')
axes[0, 1].tick_params(labelsize = 10)

sns.histplot(data = y_todos - cv_predicciones,stat = "density",kde  = True,line_kws=

axes[1, 0].set_title('Distribución residuos del modelo', fontsize = 12,fontweight =
axes[1, 0].set_xlabel("Residuo")
axes[1, 0].tick_params(labelsize = 10)


sm.qqplot( y_todos - cv_predicciones, fit = True, line  = 'q', ax  = axes[1, 1], col
axes[1, 1].set_title('Q-Q residuos del modelo', fontsize = 12, fontweight = "normal"
axes[1, 1].tick_params(labelsize = 10)

fig.tight_layout()
plt.subplots_adjust(top=0.9)
fig.suptitle('Diagnóstico residuos', fontsize = 14, fontweight = "bold");
```



- El análisis gráfico de los errores (residuos) de las precciones sobre la validación cruzada sin RM reflejan que la función de distribución se aproxima a la normal
- Sin embargo, el gráfico de residuos vs. predicciones se observa que los errores son más grades cuanto mayor el el valor de la predicción de la variable objetivo.
- Este mimo efecto se observa en la gráfica qqplot donde las colas de los valores inferiores y superiores se alejan de los valores teóricos.
- Como conclusión el modelo se ajusta bien en los valores centrales de la variable objetivo, pero pierde precisión en los valores más extremos, tienen problemas de dispersión irregular.
- En todos los casos la varianza de los residuos aumenta con los valores ajustados, esto indica que la variabilidad de los errores aumenta al aumentar su media como se esperaba al existir outliers de la variable objetivo MEDV.
- Para mejorar estos resultados se podría utilizar pruebas de igualdad de varianza (complementarias a los análisis gráficos), considera utilizar transformaciones de las variables o modelar la heterogeneidad encontrada con modelos generalizados (GLM) o modelos mixtos (MM).

5.4.2) Modelo5 : Randon Forest Regressor

a) Métricas de la Validación Cruzada

In [82]:
```
# Validación cruzada
# ==============================================================================
cv = KFold(n_splits=10, random_state=123, shuffle=True)

modelo5s_r2_scores = cross_val_score(modelo5_sinRM, X_todos, y_todos, cv = cv, scori
modelo5s_r2_mean = np.mean(modelo5s_r2_scores)
modelo5s_mse_scores = cross_val_score(modelo5_sinRM, X_todos, y_todos, cv = cv,scori
modelo5s_mse_media= abs(np.mean(modelo5s_mse_scores))
modelo5s_rmse=np.sqrt(modelo5s_mse_media)

print("1. MSE de cada n-splits: ", modelo5s_mse_scores)
```

```
print("2. MSE media: ",modelo5s_mse_media)
print("3. RMSE: ",modelo5s_rmse)
print("4. R^2 de cada n_splits", modelo5s_r2_scores)
print("5. R^2 media :" ,modelo5s_r2_mean)
```

```
1. MSE de cada n-splits:  [-3.38705633 -2.58036788 -2.23656383 -2.50371142 -2.035118
39 -2.13095483
 -2.87407618 -2.21666852 -3.11299667 -2.93838032]
2. MSE media:  2.601589436945506
3. RMSE:  1.6129443378323711
4. R^2 de cada n_splits [0.82504014 0.64685642 0.87442264 0.87239721 0.91457849 0.90
636305
 0.82347976 0.80338433 0.76942102 0.76963735]
5. R^2 media : 0.8205580404705959
```

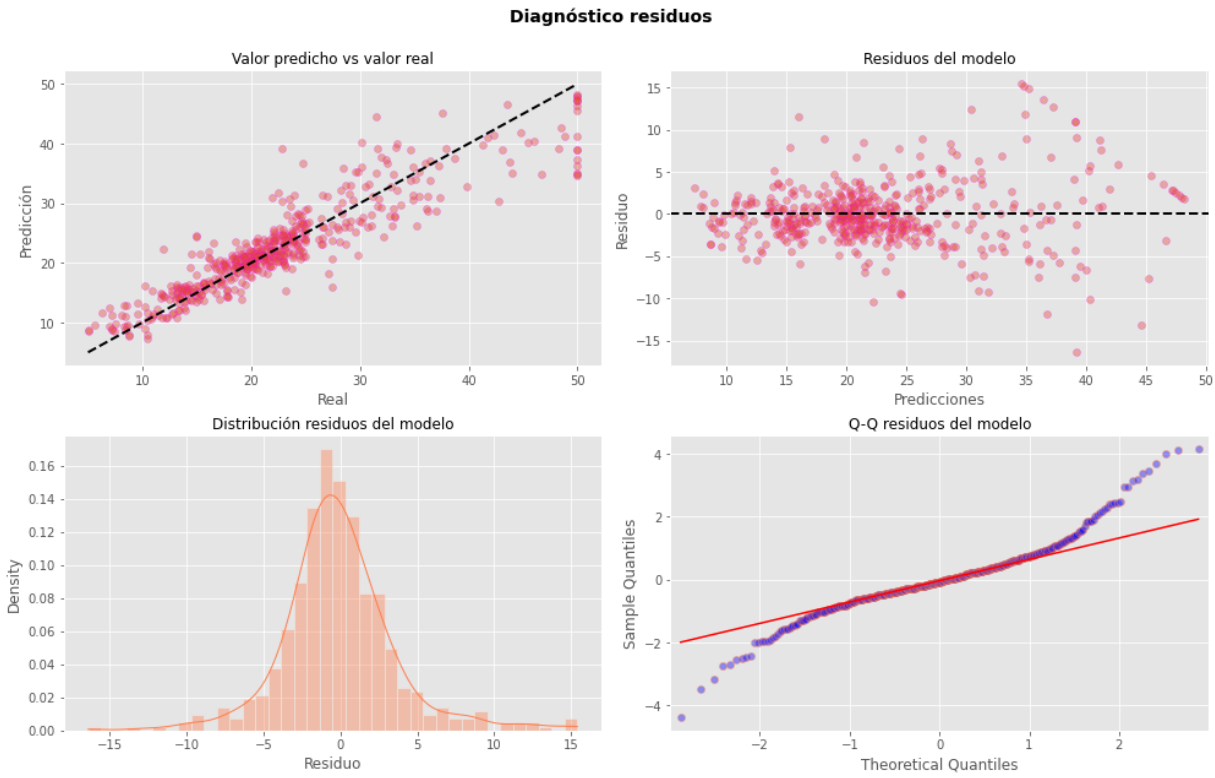
In [83]:

```
model_rdos_cruzados= np.append(model_rdos_cruzados,[[modelo5s_mse_media,modelo5s_rms
```

b) Gráficos - Diagnóstico errores (residuos) de las predicciones de validación cruzada

In [84]:

```
# Gráficos: Diagnóstico errores (residuos) de las predicciones de validación cruzada
# ==============================================================================
cv_predicciones = cross_val_predict(estimator = modelo5_sinRM, X = X_todos, y = y_to

fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(14, 9))

axes[0, 0].scatter(y_todos, cv_predicciones, edgecolors=(1, 0, 1), alpha = 0.4)
axes[0, 0].plot([y_todos.min(), y_todos.max()], [y_todos.min(), y_todos.max()],'k--'
axes[0, 0].set_title('Valor predicho vs valor real', fontsize = 12, fontweight = "no
axes[0, 0].set_xlabel('Real')
axes[0, 0].set_ylabel('Predicción')
axes[0, 0].tick_params(labelsize = 10)

#axes[0, 1].scatter(list(range(len(y_train))), y_train - cv_prediccones, edgecolors=
axes[0, 1].scatter(cv_predicciones, y_todos - cv_predicciones, edgecolors=(1, 0, 1),

axes[0, 1].axhline(y = 0, linestyle = '--', color = 'black', lw=2)
axes[0, 1].set_title('Residuos del modelo', fontsize = 12, fontweight = "normal")
axes[0, 1].set_xlabel('Predicciones')
axes[0, 1].set_ylabel('Residuo')
axes[0, 1].tick_params(labelsize = 10)

sns.histplot(data = y_todos - cv_predicciones,stat = "density",kde  = True,line_kws=

axes[1, 0].set_title('Distribución residuos del modelo', fontsize = 12,fontweight =
axes[1, 0].set_xlabel("Residuo")
axes[1, 0].tick_params(labelsize = 10)


sm.qqplot( y_todos - cv_predicciones, fit = True, line  = 'q', ax  = axes[1, 1], col
axes[1, 1].set_title('Q-Q residuos del modelo', fontsize = 12, fontweight = "normal"
axes[1, 1].tick_params(labelsize = 10)

fig.tight_layout()
plt.subplots_adjust(top=0.9)
fig.suptitle('Diagnóstico residuos', fontsize = 14, fontweight = "bold");
```



- En el modelo 5_sinRM Randon Forest Regressor, el análisis gráfico de los errores (residuos) de las precciones sobre la validación cruzada reflejan que la función de distribución se aproxima a la normal y con mayor precisión que el modelo anterior, al concentrar los errores en una rango de [-10, a 10], mientras que en el Modelo 4 Decision Tree Regressor el rango es de [-15 a 15] aproximadamente.
- El gráfico de residuos vs. predicciones se observa que los errores son más grades cuanto mayor el el valor de la predicción de la variable objetivo.
- Este mismo efecto se observa en la gráfica qqplot donde las colas de los valores inferiores y superiores se alejan de los valores teóricos.
- Como conclusión el modelo se ajusta bien en los valores centrales de la variable objetivo, pero pierde precisión en los valores más extremos, tienen problemas de dispersión irregular. En todos los casos la varianza de los residuos aumenta con los valores ajustados, esto indica que la variabilidad de los errores aumenta al aumentar su media, como se preveía al darse outliers en la variable objetivo.
- Para mejorar estos resultados se podría utilizar pruebas de igualdad de varianza (complementarias a los análisis gráficos), considera utilizar transformaciones de las variables o modelar la heterogeneidad encontrada con modelos generalizados (GLM) o modelos mixtos (MM).

4.3 Sumario de Métricas de Validación Cruzada

In [85]:
```
row_indices = ["Modelo 4: Decision Tree Regressor", "Modelo 5: Random Forest Regress
column_names = ["MSE","RMSE", "R^2"]
resultados2 = pd.DataFrame(model_rdos_cruzados, index=row_indices,columns=column_nam
resultados2.sort_values(by="MSE")
```

Out[85]:

	MSE	RMSE	R^2
Modelo 5: Random Forest Regressor	2.081402	1.442706	0.891322
Modelo 5_sinRM: Random Forest Regressor	2.601589	1.612944	0.820558

	MSE	RMSE	R^2
Modelo 4: Decision Tree Regressor	2.997457	1.731316	0.766177
Modelo 4_sinRM : Decision Tree Regressor	3.564181	1.887904	0.659742

- La validación cruzada comparada con y sin variable explicativa RM, mejora sustancialmente las métricas de MSE y RMSE en los dos modelos, mientras que reduce R^2
- El modelo 5 Randon Forest Regressor tiene mejores métricas en ambos casos, incluyendo o no la variable explicativa RM, frente al modelo 4 Decision tree Regressor con y sin RM, por tanto sería el modelo que selecionaríamos para su mejora.
- La reducción de MSE y RMSE en el modelo 5 es consistente con las métricas obtenidas inicialmente al aplicar el X_train, sin embargo en el modelo 4 la reducción de estas métricas es muy superior y debería realizarse un análisis específico.