Práctica 6. Regresión Lineal

Dataset: Housing

Housing (Ridge, Lasso, Polynomial, Interaction terms)

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
from sklearn import metrics
from sklearn.linear_model import LinearRegression,Ridge, Lasso
from sklearn.metrics import mean_squared_error,r2_score
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import PolynomialFeatures
```

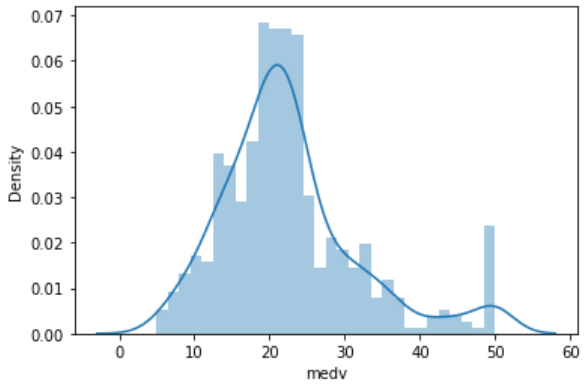
In [2]:
```
housing=pd.read_csv('BostonHousing.csv')
housing.head(3)
```

Out[2]:

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	b	lstat	medv
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	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	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	17.8	392.83	4.03	34.7

In [3]:
```
sns.distplot(housing['medv'], bins=30)
plt.show()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2551: FutureWarning: `distplot` is a deprec
ated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



In [4]:
```
correlation_matrix = housing.corr().round(2)
sns.heatmap(data=correlation_matrix, annot=True)
```

Out[4]: <AxesSubplot:>

In [5]:
```
X = housing.loc[:, housing.columns != 'medv']
Y = housing['medv']
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2, random_state=5)
print(X_train.shape, X_test.shape ,Y_train.shape, Y_test.shape)
```

```
(404, 13) (102, 13) (404,) (102,)
```

Modelo de Regresión Lineal

In [6]:
```
modeloLineal = LinearRegression()
modeloLineal.fit(X_train, Y_train)
print('Interseccion:', modeloLineal.intercept_)
print('Pendiente:', modeloLineal.coef_)
```

```
Interseccion: 37.91248700974999
Pendiente: [-1.30799852e-01  4.94030235e-02  1.09535045e-03  2.70536624e+00
 -1.59570504e+01  3.41397332e+00  1.11887670e-03 -1.49308124e+00
  3.64422378e-01 -1.31718155e-02 -9.52369666e-01  1.17492092e-02
 -5.94076089e-01]
```

In [7]:
```
y_train_predict = modeloLineal.predict(X_train)
rmse = (np.sqrt(mean_squared_error(Y_train, y_train_predict)))
RLinealScoreE = r2_score(Y_train, y_train_predict) # es el r2

y_test_predict = modeloLineal.predict(X_test)
rmse = (np.sqrt(mean_squared_error(Y_test, y_test_predict)))
RLinealScoreT= r2_score(Y_test, y_test_predict)
```

Técnica Ridge

In [8]:
```
ridge = Ridge(alpha=0.1).fit(X_train, Y_train)
RidgeScoreE=ridge.score(X_train, Y_train)
RidgeScoreT=ridge.score(X_test, Y_test)
```

Técnica Lasso

In [9]:
```
lasso = Lasso(alpha=0.1, max_iter=10).fit(X_train, Y_train)
LassoScoreE=lasso.score(X_train, Y_train)
LassoScoreT=lasso.score(X_test, Y_test)
features=np.sum(lasso.coef_ != 0)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:529: ConvergenceWarning:
Objective did not converge. You might want to increase the number of iterations. Duality gap: 4764.01239391258,
tolerance: 3.4704285816831684
  model = cd_fast.enet_coordinate_descent(
```

Polynomial

In [10]:
```
grado2 = PolynomialFeatures(degree=2)
X_grado2 = grado2.fit_transform(X_train)
lm_grado2 = modeloLineal.fit(X_grado2, Y_train)

y_grado2_fit_train = lm_grado2.predict(grado2.fit_transform(X_train))
grado2_r2E = r2_score(Y_train, y_grado2_fit_train) #lm.predict(X_quad))
#grado2_r2E = r2_score(Y_train, y_grado2_fit)
y_grado2_fit = lm_grado2.predict(grado2.fit_transform(X_test))
grado2_r2T = r2_score(Y_test, y_grado2_fit) #lm.predict(X_quad))
```

In [11]:
```
grado3 = PolynomialFeatures(degree=3)
X_grado3 = grado3.fit_transform(X_train)
lm_grado3 = modeloLineal.fit(X_grado3, Y_train)
y_grado3_fit = lm_grado3.predict(grado3.fit_transform(X_test))
grado3_r2 = r2_score(Y_test, y_grado3_fit) #lm.predict(X_quad))
```

Interaction terms

In [12]:
```
interaction = PolynomialFeatures(degree=2, include_bias=False, interaction_only=True)
X_inter = interaction.fit_transform(X_train)
X_test_inter = interaction.fit_transform(X_test)
```

In [13]:
```
model_train = modeloLineal.fit(X_inter, Y_train)
model_test = modeloLineal.fit(X_test_inter, Y_test)

y_train_predict = model_train.predict(X_inter)
rmse = (np.sqrt(mean_squared_error(Y_train, y_train_predict)))
InteractionScoreE= r2_score(Y_train, y_train_predict)

y_test_predict = model_test.predict(X_test_inter)
rmse = (np.sqrt(mean_squared_error(Y_test, y_test_predict)))
InteractionScoreT = r2_score(Y_test, y_test_predict)
```

Resumen

In [14]:
```
cabeceras= ["Entrenamiento","Test","Features"]
valores= [[RLinealScoreE,RidgeScoreE,LassoScoreE,grado2_r2E,InteractionScoreE ],
          [RLinealScoreT,RidgeScoreT,LassoScoreT,grado2_r2T,InteractionScoreT],
          ["--","--",features,"--","--"]]
resumen = pd.DataFrame(data=valores, index=cabeceras,
                       columns = ["Modelo Lineal","Ridge","Lasso","Polinomial Grado 2","Interaction Terms"])
resumen
```

Out[14]:

	Modelo Lineal	Ridge	Lasso	Polinomial Grado 2	Interaction Terms
Entrenamiento	0.738339	0.738285	0.726529	0.931557	-1.92183
Test	0.733449	0.732294	0.710041	0.869494	0.990486
Features	--	--	12.000000	--	--

Interpretación

Interpretación
La regresión lineal aplicada a la polinomial cuadrática ofrece un mejor ajuste en los estratos de entrenamiento y de prueba
Asimismo, aplicando la interacción entre las variables se llega a un ajuste casi perfecto

In [ ]:
```
```