MODELOS Y SIMULACION

En la clase de hoy veremos Regresion No Lineal: Polinómica y Logarítmica. Recordaremos como valernos de ajustes lineales para tendencias no lineales. Ademas aplicaremos ANOVA para la seleccion de variables

Para esta semana, siguen abiertas las Office Hours con el mismo link de siempre.

Ya se encuentra disponible la solucion a la practica 2 en Google Drive. La revisaremos.

Meet AIVA: AI Music

Hoy vamos a conocer una startup que genera musica con AI. Escuchemos primero una de sus composiciones https://www.youtube.com/watch?v=wtZ-GJ3xaeU&list=PLv7BOfa4CxsHAMHQj0ScPXSbgBlLglRPo&index=66

Conozcamos un poco de su CEO and Founder: https://www.youtube.com/watch?v=wYb3Wimn01s

Debatimos ¿ Es arte ?

Regresion No Lineal: Polinomial

Veamos como mejorar nuestro R2 del modelo de Regresion Lineal de la practica 1

In [2]:
```
import pandas as pd
import statsmodels.api as sm
from sklearn.preprocessing import PolynomialFeatures
import matplotlib.pyplot as plt
import numpy as np
```

In [3]:
```
#Traemos el dataset de Boston y armaremos nuestro modelo con todas las variables
boston = pd.read_csv('boston.csv')
var_explicativas = boston[['LSTAT']]
var_objetivo = boston[['MEDV']]
```

In [4]:
```
#Creamos los valores del polinomio a partir de X
polynomial_features=PolynomialFeatures(degree=2)
xp = polynomial_features.fit_transform(var_explicativas)
#Devuelve X^0, X^1, X^2
print(xp)
```

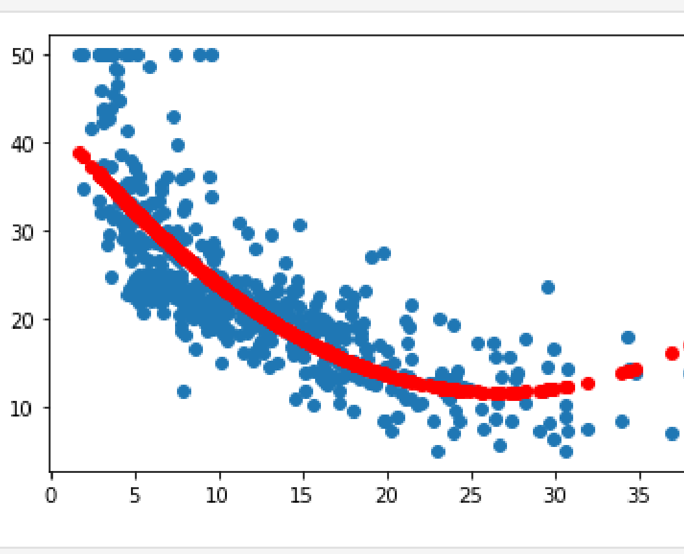
```
[[ 1.      4.98   24.8004]
 [ 1.      9.14   83.5396]
 [ 1.      4.03   16.2409]
 ...
 [ 1.      5.64   31.8096]
 [ 1.      6.48   41.9904]
 [ 1.      7.88   62.0944]]
```

In [5]:
```
#Creamos y entrenamos a nuestro modelo.
model = sm.OLS(var_objetivo, xp)
regr = model.fit()
print(regr.summary())
```

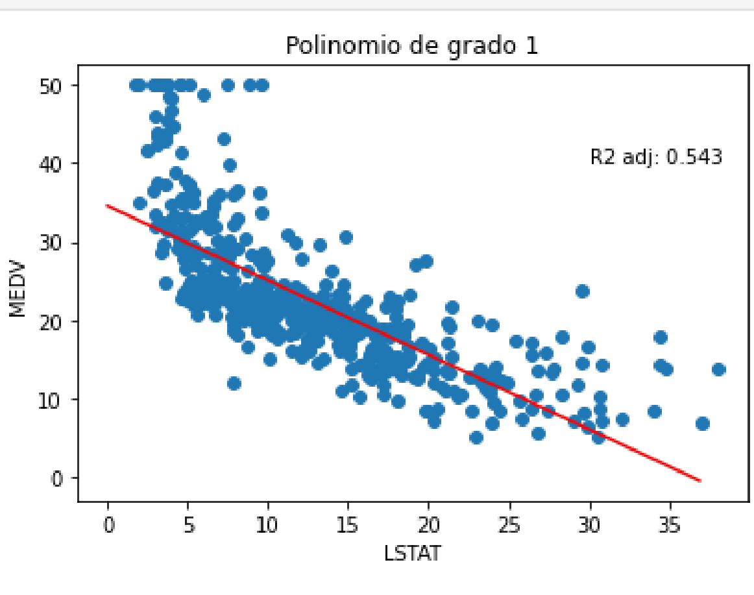
```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   MEDV   R-squared:                       0.641
Model:                            OLS   Adj. R-squared:                  0.639
Method:                 Least Squares   F-statistic:                     448.5
Date:                Sun, 05 Sep 2021   Prob (F-statistic):          1.56e-112
Time:                        12:25:40   Log-Likelihood:                -1581.3
No. Observations:                 506   AIC:                             3169.
Df Residuals:                     503   BIC:                             3181.
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         42.8620      0.872     49.149      0.000      41.149      44.575
x1            -2.3328      0.124    -18.843      0.000      -2.576      -2.090
x2             0.0435      0.004     11.628      0.000       0.036       0.051
==============================================================================
Omnibus:                      107.006   Durbin-Watson:                   0.921
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              228.388
Skew:                           1.128   Prob(JB):                     2.55e-50
Kurtosis:                       5.397   Cond. No.                     1.13e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.13e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

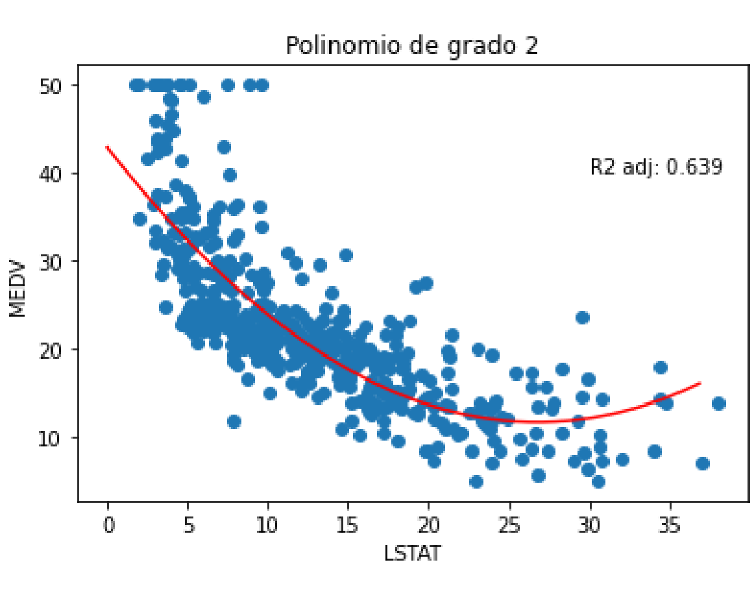
In [6]:
```
#Predizcamos los valores de precio
precios = regr.predict(xp)
```

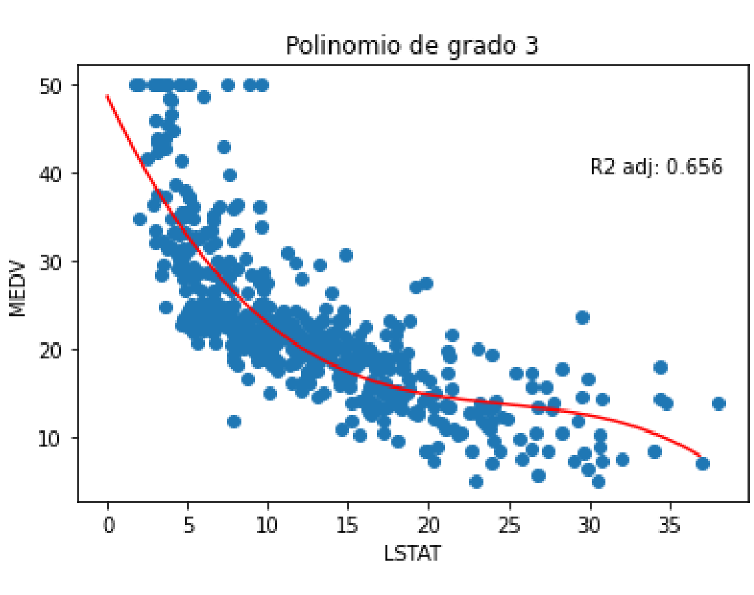
In [7]:
```
plt.scatter(var_explicativas, var_objetivo)
plt.scatter(var_explicativas, precios, color='red')
plt.show()
```

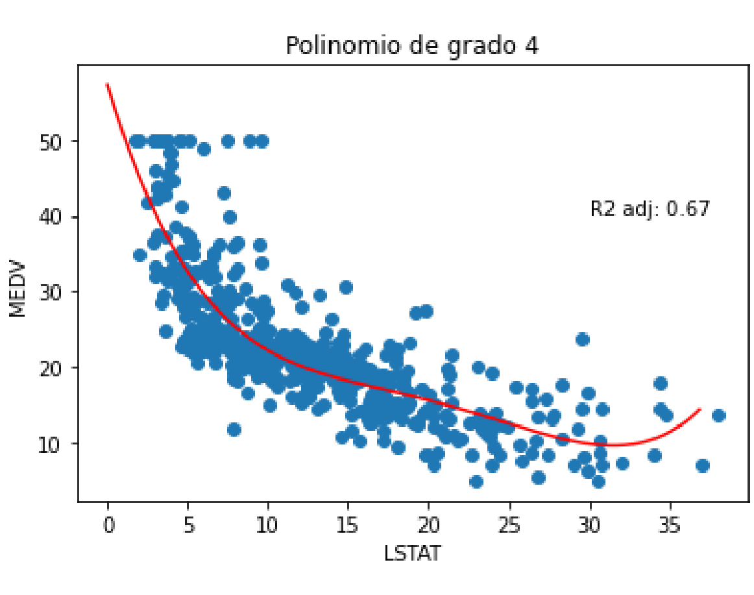


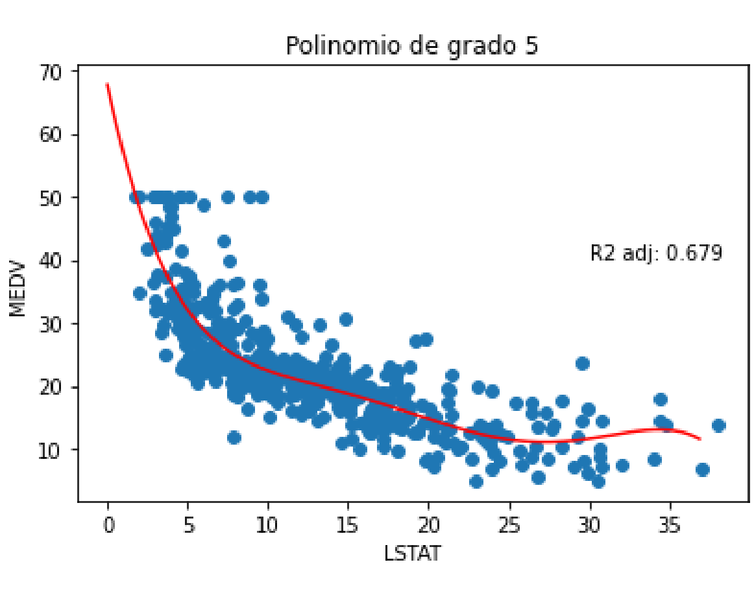
In [12]:
```
# Comparemos como cambia a medida que aumenta el grado del polinomio
for i in range(1, 10):
    # Obtengo los valores del polinomio a partir de x
    polynomial_features=PolynomialFeatures(degree=i)
    xp = polynomial_features.fit_transform(var_explicativas)
    # Entreno el modelo
    model = sm.OLS(var_objetivo, xp)
    regr = model.fit()
    # Grafico la dispersion de puntos
    plt.scatter(var_explicativas, var_objetivo)
    # Creo una columna con valores X para realizar la prediccion
    x = np.arange(37, step=0.1)
    x = x[:,None]
    xp = polynomial_features.fit_transform(x)
    # Obtengo los valores y a a partir de los x
    y = regr.predict(xp)
    # Grafico x vs y, para ver la grafica del polinomio
    plt.plot(x, y, color='red')
    # Añado valores
    plt.xlabel('LSTAT')
    plt.ylabel('MEDV')
    plt.title('Polinomio de grado ' + str(i))
    plt.text(30, 40, 'R2 adj: ' + str(round(regr.rsquared_adj,3)))
    plt.show()
```

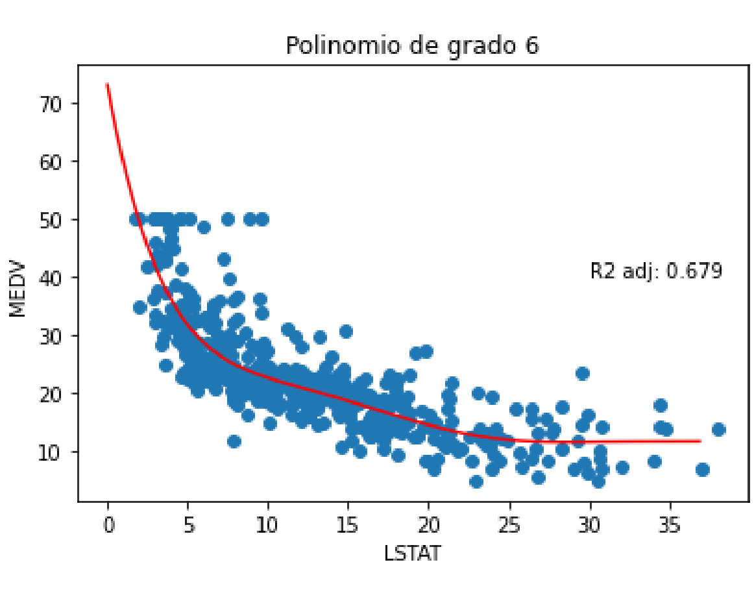


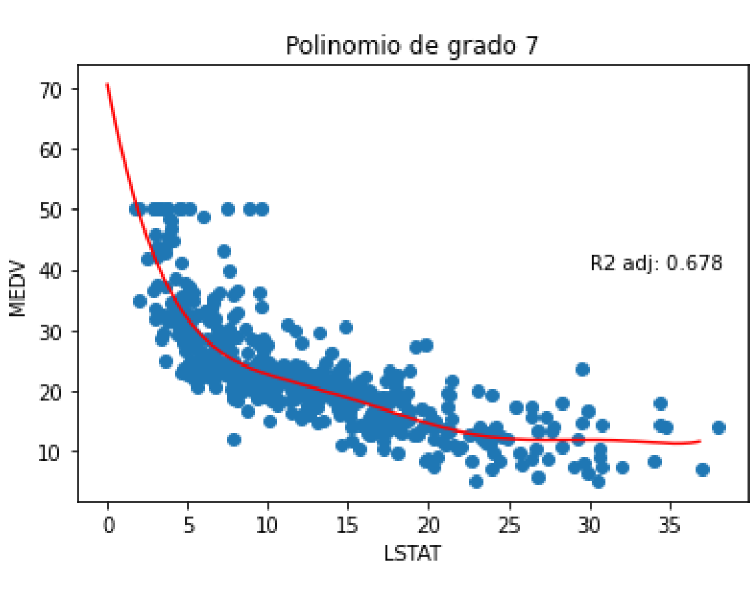


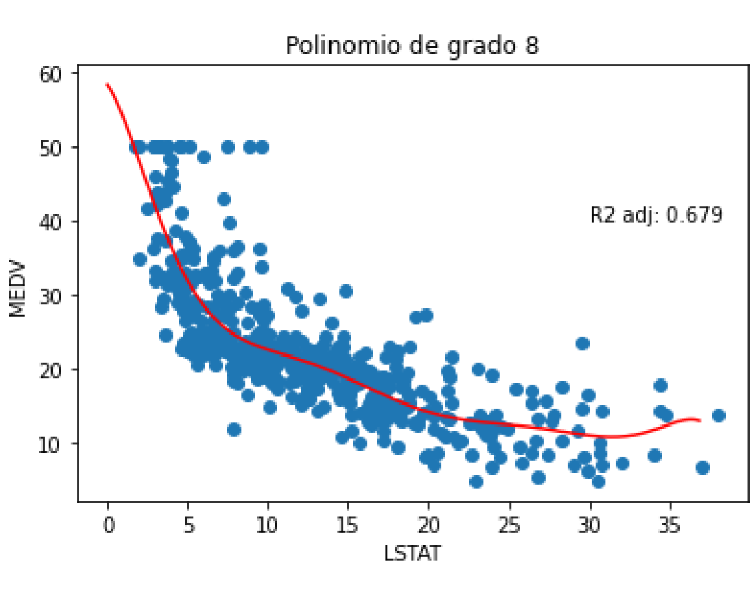


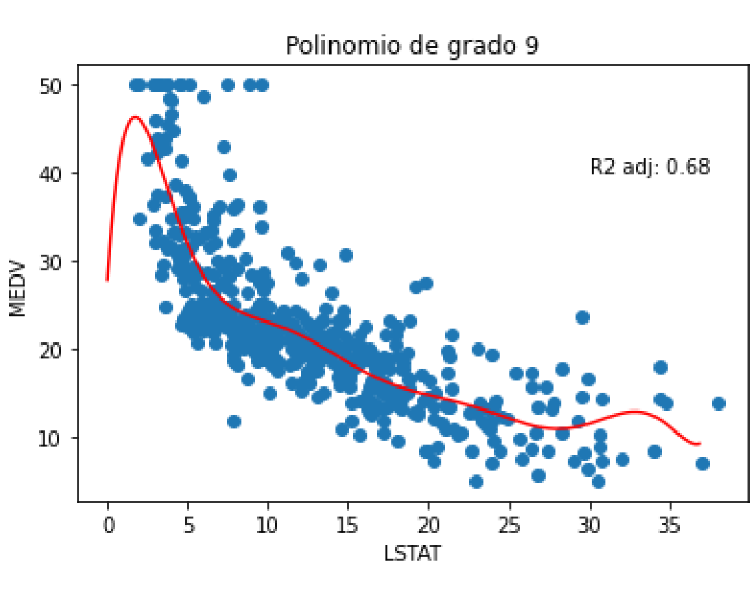












In [9]:
```
# El que mejor ajusta en este caso es un polinomio de grado 3
polynomial_features=PolynomialFeatures(degree=3)
xp = polynomial_features.fit_transform(var_explicativas)
# Entreno el modelo
model = sm.OLS(var_objetivo, xp)
regr = model.fit()
print(regr.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   MEDV   R-squared:                       0.658
Model:                            OLS   Adj. R-squared:                  0.656
Method:                 Least Squares   F-statistic:                     321.7
Date:                Sun, 05 Sep 2021   Prob (F-statistic):          1.78e-116
Time:                        12:25:47   Log-Likelihood:                -1568.9
No. Observations:                 506   AIC:                             3146.
Df Residuals:                     502   BIC:                             3163.
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         48.6496      1.435     33.909      0.000      45.831      51.468
x1            -3.8656      0.329    -11.757      0.000      -4.512      -3.220
x2             0.1487      0.021      6.983      0.000       0.107       0.191
x3            -0.0020      0.000     -5.013      0.000      -0.003      -0.001
==============================================================================
Omnibus:                      107.925   Durbin-Watson:                   0.906
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              258.171
Skew:                           1.088   Prob(JB):                     8.69e-57
Kurtosis:                       5.741   Cond. No.                     5.20e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 5.2e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Regresio No Lineal: Logaritmica

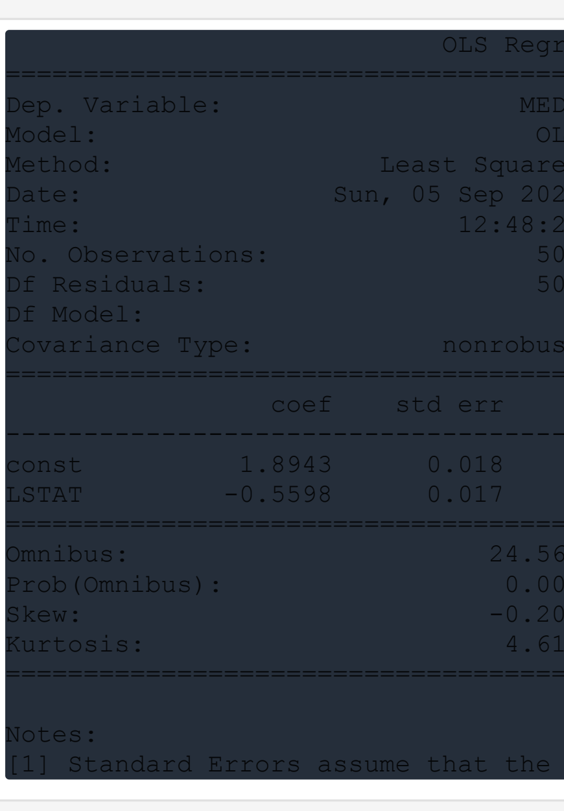
Prueben con esta herramienta https://miniwebtool.com/es/log-base-10-calculator/ que se calcula apropiadamente

In [21]:
```
# Seteamos las variables y las observemos
x_log = np.log10(var_explicativas)
y_log = np.log10(var_objetivo)
plt.scatter(x_log, y_log)
plt.xlabel('Log(LSTAT)')
plt.ylabel('Log(MEDV)')
plt.show()
```

In [24]:
```
# Entrenamos el modelo
x_log = sm.add_constant(x_log)
model = sm.OLS(y_log, x_log)
regr = model.fit()
print(regr.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   MEDV   R-squared:                       0.677
Model:                            OLS   Adj. R-squared:                  0.677
Method:                 Least Squares   F-statistic:                     1058.
Date:                Sun, 05 Sep 2021   Prob (F-statistic):          7.32e-126
Time:                        12:48:27   Log-Likelihood:                 443.35
No. Observations:                 506   AIC:                            -882.7
Df Residuals:                     504   BIC:                            -874.2
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          1.8943      0.018    103.603      0.000       1.858       1.930
LSTAT         -0.5598      0.017    -32.521      0.000      -0.594      -0.526
==============================================================================
Omnibus:                       24.565   Durbin-Watson:                   0.855
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               58.236
Skew:                          -0.205   Prob(JB):                     2.26e-13
Kurtosis:                       4.611   Cond. No.                         8.04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

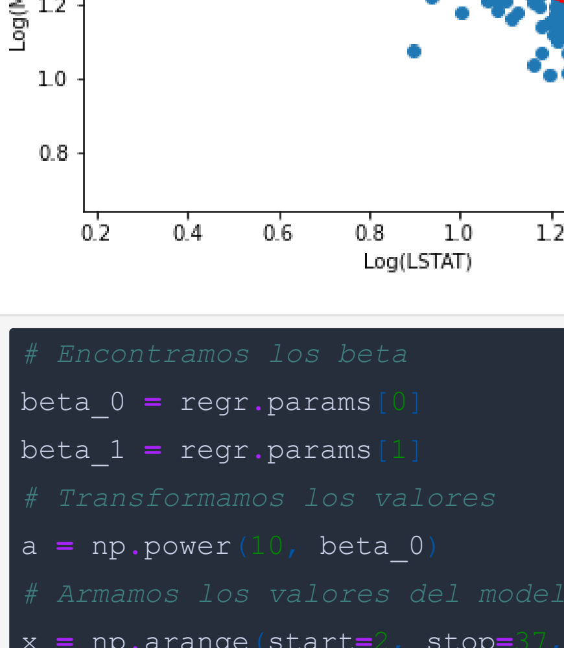
In [33]:
```
# Grafiquemos nuestro modelo
y_log_pred = regr.predict(x_log)
plt.scatter(x_log['LSTAT'], y_log)
plt.plot(x_log['LSTAT'], y_log_pred, 'r')
plt.xlabel('Log(LSTAT)')
plt.ylabel('Log(MEDV)')
plt.text(1.3, 1.6, 'R2 ADJ: ' + str(round(regr.rsquared_adj,3)))
plt.show()
```



In [70]:
```
# Encontramos los beta
beta_0 = regr.params[0]
beta_1 = regr.params[1]
# Transformamos los valores
a = np.power(10, beta_0)
# Armamos los valores del modelo
x = np.arange(start=2, stop=37, step=0.1)
x = x[:,None]
y_model = a * np.power(x, beta_1)
print(x.size, y_model.size)
```

```
350 350
```

In [73]:
```
# Visualicemos
plt.scatter(var_explicativas, var_objetivo)
plt.plot(x, y_model, 'r')
plt.xlabel('LSTAT')
plt.ylabel('MEDV')
plt.text(30, 40, 'R2 ADJ: ' + str(round(regr.rsquared_adj, 3)))
plt.show()
```



ANOVA para la seleccion de variables

In [84]:
```
# Traemos el dataset de Boston y armaremos nuestro modelo con todas las variables
boston = pd.read_csv('boston.csv')
var_explicativas = boston.drop('MEDV', 1)
var_explicativas = sm.add_constant(var_explicativas)
var_objetivo = boston[['MEDV']]
```

In [111…
```
# Creemos modelos donde incorporamos variables incrementales
model_1 = sm.OLS(var_objetivo, var_explicativas[['const', 'LSTAT']]).fit()
model_2 = sm.OLS(var_objetivo, var_explicativas[['const', 'LSTAT', 'RM']]).fit()
model_3 = sm.OLS(var_objetivo, var_explicativas[['const', 'LSTAT', 'RM', 'CRIM']]).fit()
model_4 = sm.OLS(var_objetivo, var_explicativas[['const', 'LSTAT', 'RM', 'CRIM', 'NOX']]).fit()
```

In [113…
```
table = sm.stats.anova_lm(model_1, model_2, model_3, model_4, typ=1)
# M2 admite mas variables, P value de comparacion M2 con M1 cercano a 0
# M3 es un modelo apropiado, P value de comparacion M3 con M2 cercano a 0,0014
# M4 ya no es un modelo apropiado, P value de comparacion M4 con M3 de 0,963
print(table)
```

```
   df_resid           ssr  df_diff      ss_diff           F        Pr(>F)
0     504.0  19472.381418      0.0          NaN         NaN           NaN
1     503.0  15439.309201      1.0  4033.072217  133.566398  1.460454e-27
2     502.0  15127.888385      1.0   311.420816   10.313566  1.405242e-03
3     501.0  15127.825615      1.0     0.062771    0.002079  9.636519e-01
```