MODELOS Y SIMULACION

En la clase de hoy aplicaremos las técnicas de selección discutidas la última clase: Forward Stepwise Selection, Backward Stepwise Selection y Backward Stepwise Selection with p-values.

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

Ya se encuentra disponible la solucion a la practica 1 en Google Drive.

Inteligencia Artificial y espionaje

El tema para discutir esta semana es cómo esta cambiando el espionaje a nivel mundial y como el aprovechamiento de la Inteligencia Artificial es una oportunidad y un desafio para los distintos paises u organizaciones. Esta conversacion surge a partir de un entrevista publicada por el HAI (Human-centered Artificial Intelligence) de la Universidad de Stanford. El articulo se encuentra siguiendo el siguiente link: https://hai.stanford.edu/news/re-imagining-espionage-era-artificial-intelligence

Discutimos en clase sobre el mismo.

In [104]:
```python
#Importamos las librerias que vamos a usar hoy
import pandas as pd
import numpy as np
from sklearn import linear_model
from sklearn.metrics import r2_score
import statsmodels.api as sm
import operator
import matplotlib.pyplot as plt
```

In [105]:
```python
#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_objetivo = boston['MEDV']
```

In [106]:
```python
#Cargamos el modelo y lo entrenamos
regr = linear_model.LinearRegression()
regr.fit(var_explicativas, var_objetivo)
```

Out[106]:
```
LinearRegression()
```

In [107]:
```python
#Veamos los coeficientes
regr.coef_
```

Out[107]:
```
array([-1.08011358e-01,  4.64204584e-02,  2.05586264e-02,  2.68673382e+00,
       -1.77666112e+01,  3.80986521e+00,  6.92224640e-04, -1.47556685e+00,
        3.06049479e-01, -1.23345939e-02, -9.52747232e-01,  9.31168327e-03,
       -5.24758378e-01])
```

In [108]:
```python
#Veamos el termino independiente
regr.intercept_
```

Out[108]:
```
36.45948838509015
```

In [109]:
```python
#Corremos el modelo para obtener las predicciones
prediccion_precios = regr.predict(var_explicativas)
```

In [110]:
```python
#Obtenemos el R2
r2 = r2_score(var_objetivo, prediccion_precios)
print('R2: ', round(r2, 3))
```

```
R2:  0.741
```

Calculemos R^2 Ajustado

Segun lo visto en clase, definimos a R^2 como:

$$R^2 = 1 - \frac{RSS}{TSS}$$

donde RSS es Residual Sum of Squares y se calcula como:

$$RSS = \Sigma(y_i - Y_i)^2$$

y TSS:

$$TSS = \Sigma(y_i - y_{med-i})^2$$

Para R^2_{ajus} queremos penalizar la incorporacion de nuevas variables:

$$R^2_{ajus} = 1 - \frac{RSS/(n-d-1)}{TSS/(n-1)}$$

remplazando terminos:

$$R^2_{ajus} = 1 - (1 - R^2)(\frac{n-1}{n-d-1})$$

In [111]:
```python
#En base a lo definido anteriormente calculemos R2 ajustado
#N es el numero de ocurrencias
n = var_objetivo.size
#d es el numero de variables explicativas
d = var_explicativas.columns.size
print('n: ', n, 'd: ', d)
```

```
n:  506 d:  13
```

In [112]:
```python
#Planteamos nuestra formula y veamos el resultado
r2_ajust = 1 - (1 - r2) * ((n - 1) / (n - d - 1))
print('R2 Ajustado: ', round(r2_ajust, 3))
print('R2: ', round(r2, 3))
```

```
R2 Ajustado:  0.734
R2:  0.741
```

Una nueva libreria: StatsModels

Esta libreria permite hacer estimaciones para diferentes modelos estadísticos, además de permitir hacer pruebas estadísticas y exploración de data https://www.statsmodels.org/stable/index.html.

Respecto a SKLearn, para Regresión Lineal las diferencias no son significativas. Podemos leer este artículo donde se comparan ambos módulos: https://becominghuman.ai/stats-models-vs-sklearn-for-linear-regression-f19df95ad99b

¿Por que lo utilizaremos? porque nos brinda muchos detalles estadísticos que SKLearn no.

In [113]:
```python
#En StatsModels debemos agregar nosotros mismo la constante para el armado de nuestro modelo.
var_explicativas = sm.add_constant(var_explicativas)
var_explicativas.head()
```

Out[113]:

	const	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
0	1.0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98
1	1.0	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14
2	1.0	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03
3	1.0	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94
4	1.0	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33

In [114]:
```python
#Definimos nuestro modelo, OLS significa Ordinary Least Squares. Esto indica la forma en la que
#se calcularan los parametros
model = sm.OLS(var_objetivo, var_explicativas)
```

In [115]:
```python
#Entrenamos el modelo (notemos las diferencias con sklearn)
regr = model.fit()
```

In [116]:
```python
#Veamos todos los datos que nos entrega StatsModels
print(regr.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   MEDV   R-squared:                       0.741
Model:                            OLS   Adj. R-squared:                  0.734
Method:                 Least Squares   F-statistic:                     108.1
Date:                Tue, 31 Aug 2021   Prob (F-statistic):          6.72e-135
Time:                        16:16:37   Log-Likelihood:                -1498.8
No. Observations:                 506   AIC:                             3026.
Df Residuals:                     492   BIC:                             3085.
Df Model:                          13
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         36.4595      5.103      7.144      0.000      26.432      46.487
CRIM          -0.1080      0.033     -3.287      0.001      -0.173      -0.043
ZN             0.0464      0.014      3.382      0.001       0.019       0.073
INDUS          0.0206      0.061      0.334      0.738      -0.100       0.141
CHAS           2.6867      0.862      3.118      0.002       0.993       4.380
NOX          -17.7666      3.820     -4.651      0.000     -25.272     -10.262
RM             3.8099      0.418      9.116      0.000       2.989       4.631
AGE            0.0007      0.013      0.052      0.958      -0.025       0.027
DIS           -1.4756      0.199     -7.398      0.000      -1.867      -1.084
RAD            0.3060      0.066      4.613      0.000       0.176       0.436
TAX           -0.0123      0.004     -3.280      0.001      -0.020      -0.005
PTRATIO       -0.9527      0.131     -7.283      0.000      -1.210      -0.696
B              0.0093      0.003      3.467      0.001       0.004       0.015
LSTAT         -0.5248      0.051    -10.347      0.000      -0.624      -0.425
==============================================================================
Omnibus:                      178.041   Durbin-Watson:                   1.078
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              783.126
Skew:                           1.521   Prob(JB):                    8.84e-171
Kurtosis:                       8.281   Cond. No.                     1.51e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.51e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

In [117]:
```python
#Para acceder al R2 y R2 Ajustado
r2 = regr.rsquared
r2_adj = regr.rsquared_adj
print('R2: ', round(r2, 3))
print('R2 Ajustado: ', round(r2_adj, 3))
```

```
R2:  0.741
R2 Ajustado:  0.734
```

In [118]:
```python
#Y lo mas interesante: Los p-values
p_values = regr.pvalues
print(regr.pvalues)
```

```
const      3.283438e-12
CRIM       1.086810e-03
ZN         7.781097e-04
INDUS      7.382881e-01
CHAS       1.925030e-03
NOX        4.245644e-06
RM         1.979441e-18
AGE        9.582293e-01
DIS        6.013491e-13
RAD        5.070529e-06
TAX        1.111637e-03
PTRATIO    1.308835e-12
B          5.728592e-04
LSTAT      7.776912e-23
dtype: float64
```

In [119]:
```python
#Cuales son mayores a 0.05
print("Variables which p value is > 0.05:")
for index, value in p_values.items():
    if value > 0.05:
        print(index)
```

```
Variables which p value is > 0.05:
INDUS
AGE
```

In [120]:
```python
#Ordenemoslos de mayor a menor
regr.pvalues.sort_values(ascending=False)
```

Out[120]:
```
AGE        9.582293e-01
INDUS      7.382881e-01
CHAS       1.925030e-03
TAX        1.111637e-03
CRIM       1.086810e-03
ZN         7.781097e-04
B          5.728592e-04
RAD        5.070529e-06
NOX        4.245644e-06
const      3.283438e-12
PTRATIO    1.308835e-12
DIS        6.013491e-13
RM         1.979441e-18
LSTAT      7.776912e-23
dtype: float64
```

Forward Stepwise Selection

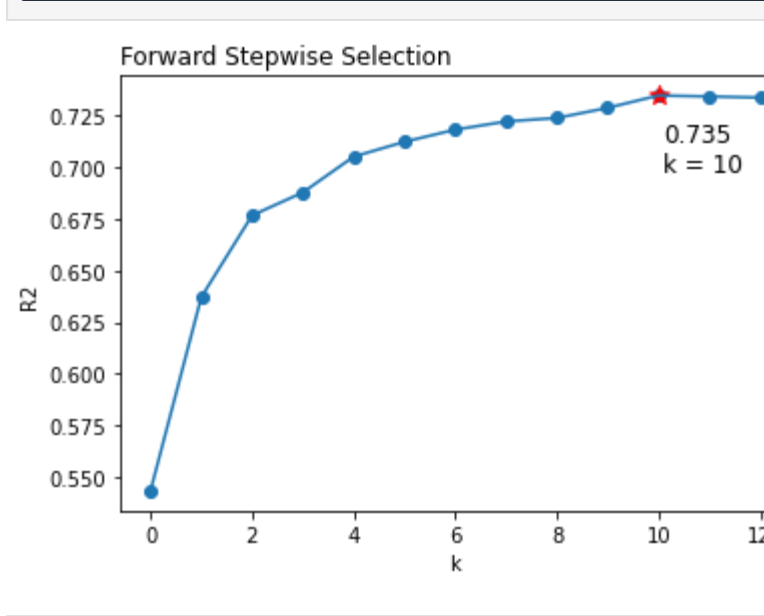
Partimos de un modelo sin variables predictoras. Luego comenzamos a agregar de a 1 variable, eligiendo en cada ronda aquel que tenga mayor R2 A medida que vamos conformando nuestros modelos, almacenamos el R2 ajustado Terminando la iteracion, graficamos los R2 ajustados y seleccionamos el mejor modelo.

In [121]:
```python
#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 [122]:
```python
# Seteamos las condiciones iniciales necesarias
variables = ['const']
iterate_columns = var_explicativas.columns.drop('const',1)
r2_adj = []
vars_size = iterate_columns.size
var_model = {}
# Iteramos k veces, siendo k la cantidad de variables
for k in range(0, vars_size):
    # Acumulamos los valores de R2 para cada variable en los modelos que armaremos
    r2 = {}
    # Iteramos sobre todas las variables disponibles para la ronda
    for var in iterate_columns:
        # Fijamos que variables seran las que definiran nuestro modelo
        var_explicativa = var_explicativas[variables + [var]]
        model = sm.OLS(var_objetivo, var_explicativa)
        regr = model.fit()
        # Almacenamos el R2 para cada set de variables que se prueba
        r2[var] = regr.rsquared_adj
    # Seleccionamos aquella con mayor R2
    var_max_r2 = max(r2.items(), key=operator.itemgetter(1))[0]
    # Almacenamos las variables que describen a ese modelo
    var_model[k] = var_max_r2
    # Almacenamos el valor de R2
    r2_adj.append(r2[var_max_r2])
    # Eliminamos la variable para que deje de estar en consideracion en el ciclo siguiente
    iterate_columns = iterate_columns.drop(var_max_r2,1)
    # Agregamos la variable seleccionada en esta vuelta para que la cuente en el modelo
siguiente
    variables.append(var_max_r2)
#Seteamos el valor de la ronda k donde R2 es maximo
r2_max_index = r2_adj.index(max(r2_adj))
```

In [123]:
```python
def r2_variation(vars_size, r2_adj, title, x_label, y_label):
    """
    Plot the scatter and the curve that shows how R2 value vary over
    every iteration. A title and labels for the graph must be included.
    Also, highlight the point where there is a maximum R2 value. Add information
    about that point. The amount of independent variables is given.
    """
    # Graficamos la solucion
    # Los valores para señalar cada iteracion
    x = np.arange(vars_size)
    # El punto donde R2 es maximo
    r2_max_x = r2_adj.index(max(r2_adj))
    r2_max_y = max(r2_adj)
    # Titulo de nuestro grafico y nombre de los ejes
    plt.title(title, loc = 'left')
    plt.xlabel(x_label)
    plt.ylabel(y_label)
    # Graficamos la recta con la variacion del R2 en cada vuelta
    plt.plot(x, r2_adj)
    # Graficamos dispersion de puntos de K vs R2 ajustado
    plt.scatter(x, r2_adj)
    # Graficamos el punto donde R2 es maximo y añadimos su valor en el grafico
    plt.scatter(r2_max_x, r2_max_y, marker='*', s=100, color='red')
    plt.text(r2_max_x * (1 + 0.01), r2_max_y * (1 - 0.02), round(r2_max_y, 3), fontsize=12)
    plt.text(r2_max_x * (1 + 0.01), r2_max_y * (1 - 0.04), 'k = ' + str(r2_max_x), fontsize=12)
    plt.show()
```

In [124]:
```python
r2_variation(vars_size, r2_adj, 'Forward Stepwise Selection', 'k', 'R2')
```



In [125]:
```python
var_model
```

Out[125]:
```
{0: 'LSTAT',
 1: 'RM',
 2: 'PTRATIO',
 3: 'DIS',
 4: 'NOX',
 5: 'CHAS',
 6: 'B',
 7: 'ZN',
 8: 'CRIM',
 9: 'RAD',
 10: 'TAX',
 11: 'INDUS',
 12: 'AGE'}
```

In [126]:
```python
# El modelo definitivo
variables = []
i = 0
while i <= r2_max_index:
    variables.append(var_model[i])
    i += 1
var_explicativa = var_explicativas[variables]
var_explicativa = sm.add_constant(var_explicativa)
model = sm.OLS(var_objetivo, var_explicativa)
regr = model.fit()
print(regr.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   MEDV   R-squared:                       0.741
Model:                            OLS   Adj. R-squared:                  0.735
Method:                 Least Squares   F-statistic:                     128.2
Date:                Tue, 31 Aug 2021   Prob (F-statistic):          5.54e-137
Time:                        16:16:37   Log-Likelihood:                -1498.9
No. Observations:                 506   AIC:                             3022.
Df Residuals:                     494   BIC:                             3076.
Df Model:                          11
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         36.3411      5.067      7.171      0.000      26.384      46.298
LSTAT         -0.5226      0.047    -11.019      0.000      -0.616      -0.429
RM             3.8016      0.406      9.356      0.000       3.003       4.600
PTRATIO       -0.9465      0.129     -7.334      0.000      -1.200      -0.693
DIS           -1.4927      0.186     -8.037      0.000      -1.857      -1.128
NOX          -17.3760      3.535     -4.915      0.000     -24.322     -10.430
CHAS           2.7187      0.854      3.183      0.002       1.041       4.397
B              0.0093      0.003      3.475      0.001       0.004       0.015
ZN             0.0458      0.014      3.390      0.001       0.019       0.072
CRIM          -0.1084      0.033     -3.307      0.001      -0.173      -0.044
RAD            0.2996      0.063      4.726      0.000       0.175       0.424
TAX           -0.0118      0.003     -3.493      0.001      -0.019      -0.005
==============================================================================
Omnibus:                      178.430   Durbin-Watson:                   1.078
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              787.785
Skew:                           1.523   Prob(JB):                    8.60e-172
Kurtosis:                       8.300   Cond. No.                     1.47e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.47e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

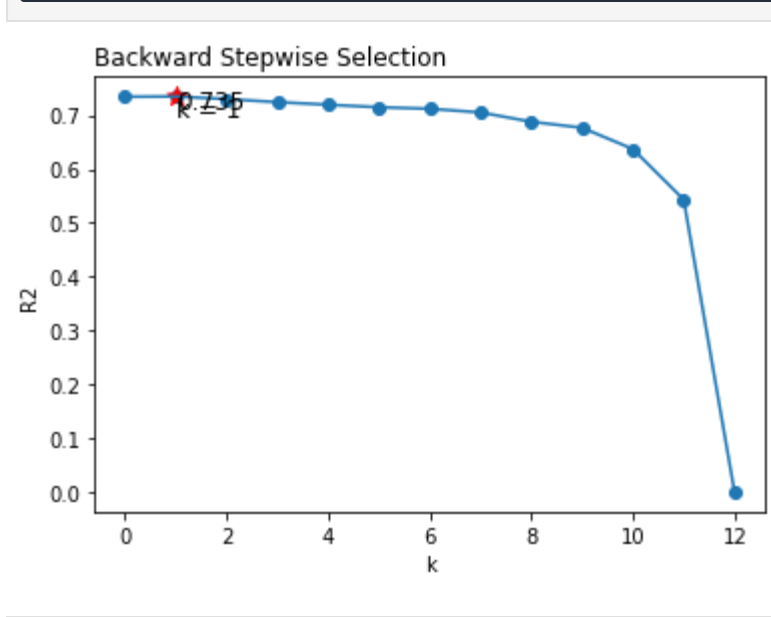
Backward Stepwise Selection

Partimos de un modelo definido por todas variables predictoras. Luego comenzamos a eliminar de a 1 variable, eligiendo en cada ronda aquel que tenga mayor R2 A medida que vamos conformando nuestros modelos, almacenamos el R2 ajustado Terminando la iteracion, graficamos los R2 ajustados y seleccionamos el mejor modelo.

In [127]:
```python
boston = pd.read_csv('boston.csv')
var_explicativas = boston.drop('MEDV', 1)
var_explicativas = sm.add_constant(var_explicativas)
var_objetivo = boston['MEDV']
```

In [128]:
```python
# Seteamos las condiciones iniciales necesarias
variables = var_explicativas.columns
iterate_columns = variables.drop('const', 1)
vars_size = iterate_columns.size
r2_adj = []
var_model = {}
# Iteramos k veces, siendo k la cantidad de variables
for k in range(0, vars_size):
    # Acumulamos los valores de R2 para cada variable en los modelos que armaremos
    r2 = {}
    # Indice para insertar las variables temporalmente removidas en su lugar
    i = 1
    # Iteramos sobre todas las variables disponibles para la ronda
    for var in iterate_columns:
        #Eliminamos la variable para probar
        variables = variables.drop(var, 1)
        # Fijamos que variables seran las que definiran nuestro modelo
        var_explicativa = var_explicativas[variables]
        # Insertamos nuevamente la variable que sacamos en el lugar que estaba para la prox
ronda
        variables = variables.insert(i, var)
        # Entrenamos el modelo
        model = sm.OLS(var_objetivo, var_explicativa)
        regr = model.fit()
        # Almacenamos el R2 para cada set de variables que se prueba
        r2[var] = regr.rsquared_adj
        # Actualizador de indice
        i += 1
    # Seleccionamos aquella con mayor R2
    var_max_r2 = max(r2.items(), key=operator.itemgetter(1))[0]
    # Almacenamos el valor de R2
    r2_adj.append(r2[var_max_r2])
    # Almacenamos las variables que describen a ese modelo
    var_model[k] = var_max_r2
    # No itera mas sobre esa variable tampoco
    iterate_columns = iterate_columns.drop(var_max_r2, 1)
    # Dejamos de considerar como una variable posible para definir al modelo
    variables = variables.drop(var_max_r2, 1)
#Seteamos el valor de la ronda k donde R2 es maximo
r2_max = r2_adj.index(max(r2_adj))
```

In [129]:
```python
r2_variation(vars_size, r2_adj, 'Backward Stepwise Selection', 'k', 'R2')
```



In [130]:
```python
# El modelo definitivo
variables = var_explicativas.columns
i = 0
while i <= r2_max:
    variables = variables.drop(var_model[i], 1)
    i += 1
var_explicativa = var_explicativas[variables]
var_explicativa = sm.add_constant(var_explicativa)
model = sm.OLS(var_objetivo, var_explicativa)
regr = model.fit()
print(regr.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   MEDV   R-squared:                       0.741
Model:                            OLS   Adj. R-squared:                  0.735
Method:                 Least Squares   F-statistic:                     128.2
Date:                Tue, 31 Aug 2021   Prob (F-statistic):          5.54e-137
Time:                        16:16:38   Log-Likelihood:                -1498.9
No. Observations:                 506   AIC:                             3022.
Df Residuals:                     494   BIC:                             3076.
Df Model:                          11
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         36.3411      5.067      7.171      0.000      26.384      46.298
CRIM          -0.1084      0.033     -3.307      0.001      -0.173      -0.044
ZN             0.0458      0.014      3.390      0.001       0.019       0.072
CHAS           2.7187      0.854      3.183      0.002       1.041       4.397
NOX          -17.3760      3.535     -4.915      0.000     -24.322     -10.430
RM             3.8016      0.406      9.356      0.000       3.003       4.600
DIS           -1.4927      0.186     -8.037      0.000      -1.857      -1.128
RAD            0.2996      0.063      4.726      0.000       0.175       0.424
TAX           -0.0118      0.003     -3.493      0.001      -0.019      -0.005
PTRATIO       -0.9465      0.129     -7.334      0.000      -1.200      -0.693
B              0.0093      0.003      3.475      0.001       0.004       0.015
LSTAT         -0.5226      0.047    -11.019      0.000      -0.616      -0.429
==============================================================================
Omnibus:                      178.430   Durbin-Watson:                   1.078
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              787.785
Skew:                           1.523   Prob(JB):                    8.60e-172
Kurtosis:                       8.300   Cond. No.                     1.47e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.47e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Backward Stepwise Selection with P-Values

Partimos de un modelo definido por todas variables predictoras. Luego eliminamos, en cada vuelta, a la variable con mayor P-Value A medida que vamos conformando nuestros modelos, almacenamos el R2 ajustado Terminando la iteracion, graficamos los R2 ajustados y seleccionamos el mejor modelo.

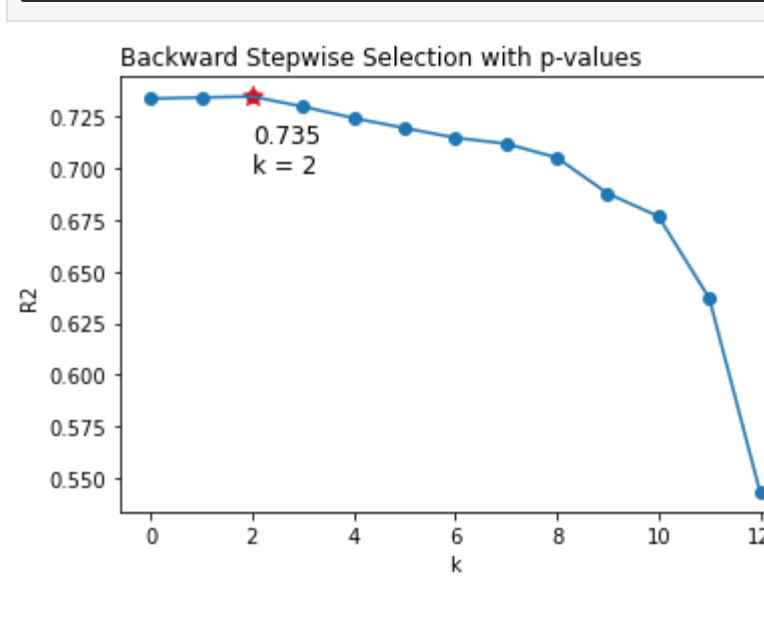
In [131]:
```python
boston = pd.read_csv('boston.csv')
var_explicativas = boston.drop('MEDV', 1)
var_explicativas = sm.add_constant(var_explicativas)
var_objetivo = boston['MEDV']
```

In [132]:
```python
variables = var_explicativas.columns
r2_adj = []
vars_out = []
for k in range(0, vars_size):
    var_explicativa = var_explicativas[variables]
    # Entrenamos el modelo
    model = sm.OLS(var_objetivo, var_explicativa)
    regr = model.fit()
    # Almacenamos el valor de R2
    r2_adj.append(regr.rsquared_adj)
    # Obtenemos los p-values sin considerar la constante
    p_values = regr.pvalues.drop('const')
    pvalue_index = p_values.argmax()
    pvalue_var = p_values.keys()[pvalue_index]
    # Eliminamos la variable cuyo p-value es el max grande
    variables = variables.drop(pvalue_var)
    # Almacenamos la variable que queda fuera en cada ronda
    vars_out.append(pvalue_var)
```

In [133]:
```python
vars_out
```

Out[133]:
```
['AGE',
 'INDUS',
 'CHAS',
 'ZN',
 'TAX',
 'CRIM',
 'RAD',
 'B',
 'NOX',
 'DIS',
 'PTRATIO',
 'RM',
 'LSTAT']
```

In [134]:
```python
r2_variation(vars_size, r2_adj, 'Backward Stepwise Selection with p-values', 'k', 'R2')
```



Esto es todo por hoy. La practica 2 sera lanzada al finalizar esta clase. Lo visto hoy quizas les presente dificultad. Es normal. Buena semana :)