#Laboratorio del Curso: Data Analysis with Python

#Módulo 04: Model Development

#Materia: Ciencia y Analítica de Datos

#Profesora: Dra. María de la Paz Rico Fernández

#Alumno: Francisco Javier Ramírez Arias

#Matrícula: A01316379

Objetivos

- Desarrollar modelos predictivos

```
#Se importan las diferentes librerias a utilizar.
import pandas as pd
import numpy as np
import matplotlib.pyplot as pl

path='https://cf-courses-data.s3.us.cloud-object-
storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DA0101EN-
SkillsNetwork/labs/Data%20files/automobileEDA.csv'
df = pd.read_csv(path)
df.head()
```

```
   symboling  normalized-losses         make aspiration num-of-
doors  \
0          3                122  alfa-romero        std          two

1          3                122  alfa-romero        std          two

2          1                122  alfa-romero        std          two

3          2                164         audi        std         four

4          2                164         audi        std         four


    body-style drive-wheels engine-location  wheel-base    length  ...
\
0  convertible          rwd           front        88.6  0.811148  ...

1  convertible          rwd           front        88.6  0.811148  ...

2    hatchback          rwd           front        94.5  0.822681  ...

3        sedan          fwd           front        99.8  0.848630  ...

4        sedan          4wd           front        99.4  0.848630  ...
```

```
   compression-ratio  horsepower  peak-rpm city-mpg highway-mpg
price  \
0                9.0       111.0    5000.0       21          27
13495.0
1                9.0       111.0    5000.0       21          27
16500.0
2                9.0       154.0    5000.0       19          26
16500.0
3               10.0       102.0    5500.0       24          30
13950.0
4                8.0       115.0    5500.0       18          22
17450.0

  city-L/100km  horsepower-binned  diesel  gas
0    11.190476             Medium       0    1
1    11.190476             Medium       0    1
2    12.368421             Medium       0    1
3     9.791667             Medium       0    1
4    13.055556             Medium       0    1

[5 rows x 29 columns]
```

#Modelo de Regresion Lineal y Regresion Lineal Multiple

```
#Libreria para llevar a cabo la Regresión Lineal
from sklearn.linear_model import LinearRegression

#Creamos el objeto de regresion lineal
lm = LinearRegression()
lm

LinearRegression()

#Definimos nuestras variables de entrada y salida
X = df[['highway-mpg']]
Y = df['price']

#Ajustamos el modelo lineal
lm.fit(X,Y)

LinearRegression()

#Realizamos predicciones
Yhat=lm.predict(X)
Yhat[0:5]

array([16236.50464347, 16236.50464347, 17058.23802179, 13771.3045085 ,
       20345.17153508])
```

```
#Cual es el valor del coeficiente a?
lm.intercept_
```

```
38423.3058581574
```

```
#Cual es el valor del coeficiente b?
lm.coef_
```

```
array([-821.73337832])
```

#Modelo lineal final estimado

##Precio = 38423.31 - 821.73 x highway-mpg

##Pregunta #1: Crea un objeto de regresion lineal llamado "lm1".

```
lm1 = LinearRegression()
lm1
```

```
LinearRegression()
```

##Pregunta #2: Entrena un modelo utilizando "engine-size" como la variable independiente y "precio" como la variabel dependiente

```
lm1.fit(df[['engine-size']], df[['price']])
lm1
```

```
LinearRegression()
```

##Pregunga #3: Encuentra la pendiente e intercepcion del modelo

```
print(lm1.coef_)
print(lm1.intercept_)
```

```
[[166.86001569]]
[-7963.33890628]
```

##Pregunta #4: Cuál es la ecuaciónn de la linea de predicción? Tu puedes utilizar x y yhat o "engine-size" or "price"

```
Yhat = -7963.34 + 166.86*X
```

```
Price = -7963.34 + 166.86*df['engine-size']
```

#Regresión Lineal Múltiple

```
#Desarrollemos el modelo utilizando las siguientes variables
Z = df[['horsepower', 'curb-weight', 'engine-size', 'highway-mpg']]
```

```
#Ajustamos el modelo utilizando las cuatro variables mencionadas
lm.fit(Z, df['price'])
```

```
LinearRegression()
```

```
#Valor de intercepción
lm.intercept_
```

```
-15806.62462632922
```

```
#Valores de los coeficientes
lm.coef_
```

```
array([53.49574423,  4.70770099, 81.53026382, 36.05748882])
```

#Modelo lineal multiple final estimado

##Precio = -15806.62 - 53.4957*horsepower +4.7077*curb-waigth + 81.5302*engine-size + 36.0574*highway-mpg

##Pregunta 1: Crea y entrena un modelo de Regresión Lineal Múltiple con el nombre de "lm2" donde la variable de respuesta sea el "precio" y las variables predictor sean "normalized-losses" and "highway-mpg"

```
lm2 = LinearRegression()
lm2.fit(df[['normalized-losses', 'highway-mpg']], df['price'])
```

```
LinearRegression()
```

##Pregunta 2: Encuentra el coeficiente del modelo

```
lm2.coef_
```

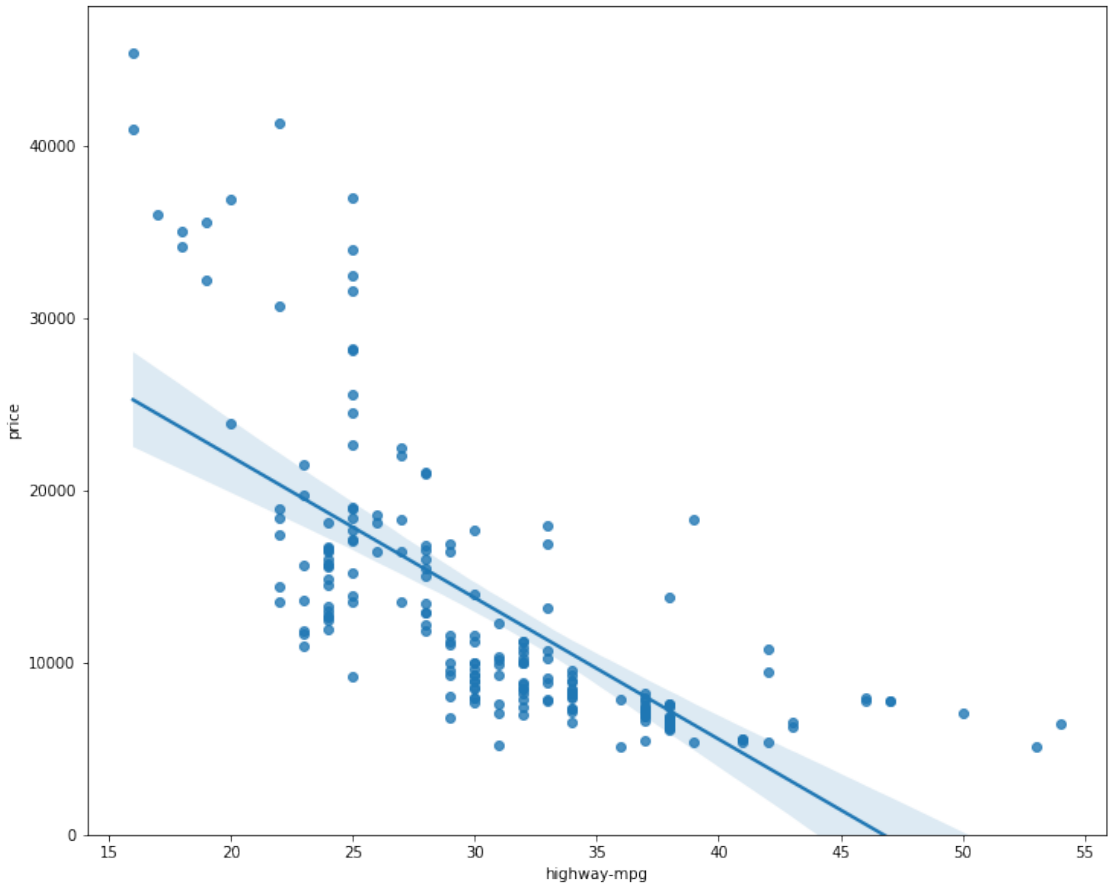
```
array([   1.49789586, -820.45434016])
```

##Grafica de Regresion

```
import seaborn as sns
%matplotlib inline
```

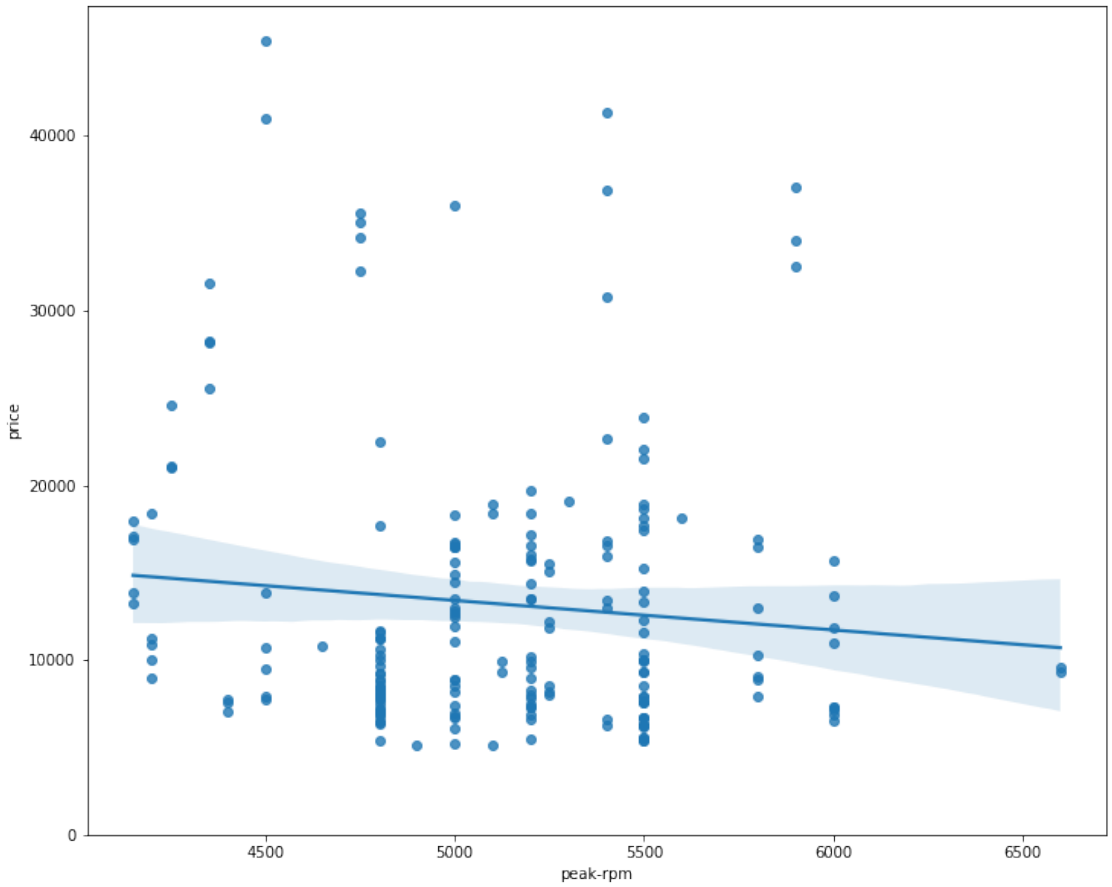
```
width = 12
height = 10
pl.figure(figsize=(width, height))
sns.regplot(x="highway-mpg", y="price", data=df)
pl.ylim(0,)
```

```
(0.0, 48155.81889354323)
```




```
pl.figure(figsize=(width, height))
sns.regplot(x="peak-rpm", y="price", data=df)
pl.ylim(0,)
```

```
(0.0, 47414.1)
```



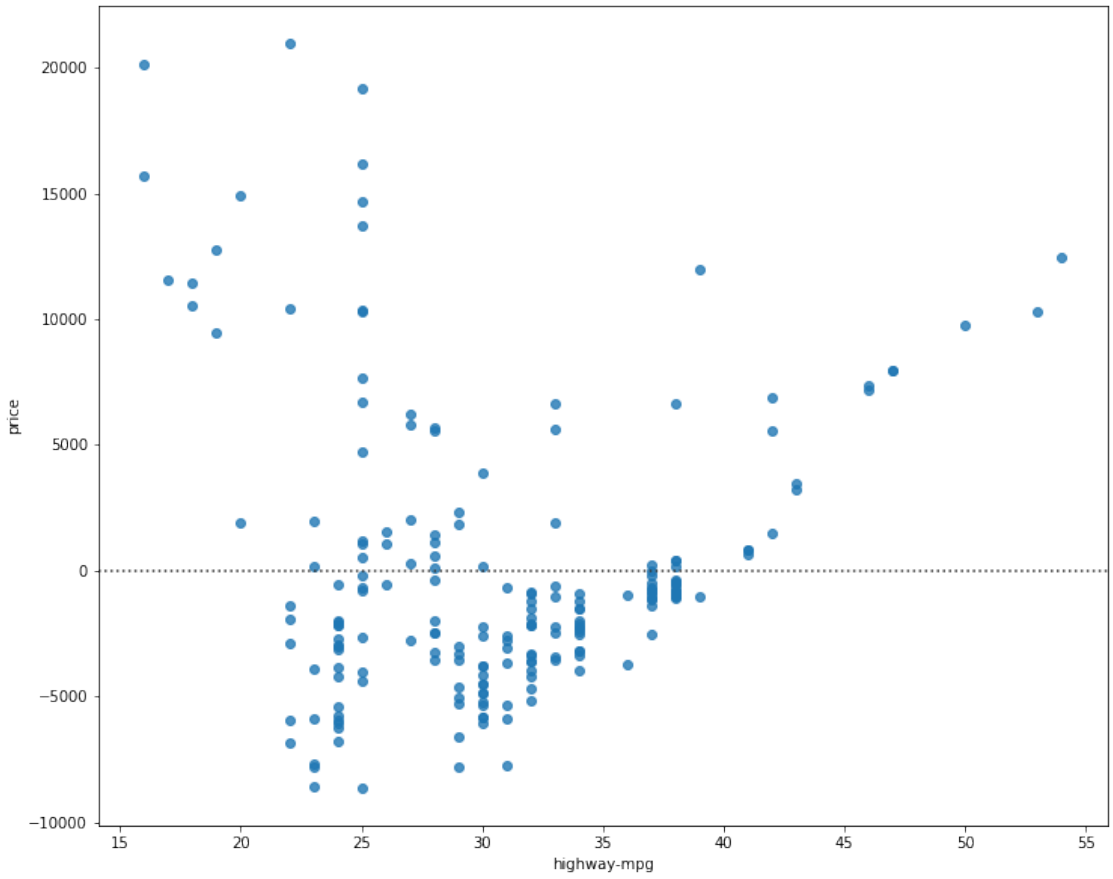
##Pregunta 3: Dada las graficas anteriores, es "peak-rpm" o "highway-mpg" mas correlacionada con el precio? Utiliza el metodo ".corr()" para verificar la respuesta

```
df[["peak-rpm", "highway-mpg", "price"]].corr()
```

```
             peak-rpm  highway-mpg     price
peak-rpm     1.000000    -0.058598 -0.101616
highway-mpg -0.058598     1.000000 -0.704692
price       -0.101616    -0.704692  1.000000
```

##Grafica Residual

```
width = 12
height = 10
pl.figure(figsize=(width, height))
sns.residplot(x=df['highway-mpg'],y=df['price'])
pl.show()
```

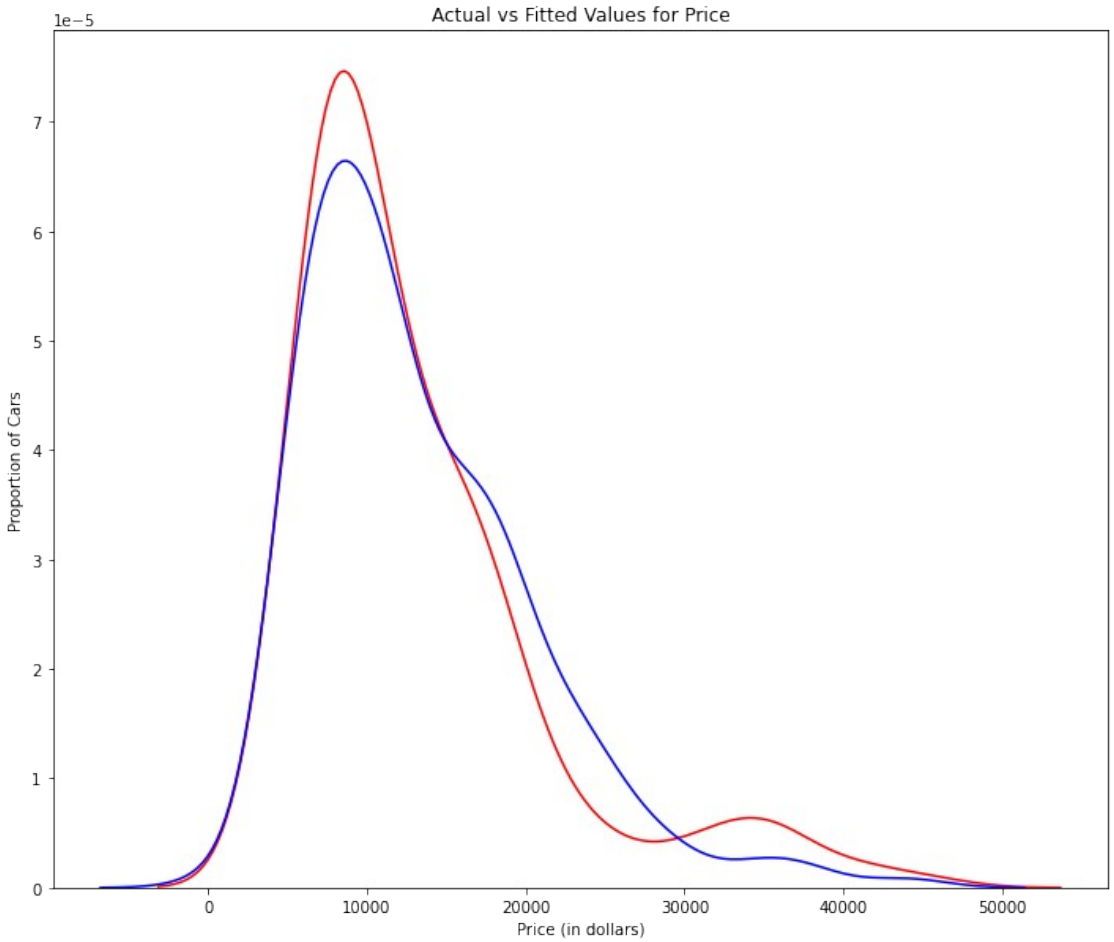


##Podemos observar que un modelo no-lineal sea más adecuado para los datos.

```
#Regresión Lineal Múltiple
Y_hat = lm.predict(Z)

pl.figure(figsize=(width, height))


ax1 = sns.distplot(df['price'], hist=False, color="r", label="Actual
Value")
sns.distplot(Y_hat, hist=False, color="b", label="Fitted Values" ,
ax=ax1)


pl.title('Actual vs Fitted Values for Price')
pl.xlabel('Price (in dollars)')
pl.ylabel('Proportion of Cars')

pl.show()
pl.close()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619:
FutureWarning: `distplot` is a deprecated 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 `kdeplot` (an axes-
level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619:
FutureWarning: `distplot` is a deprecated 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 `kdeplot` (an axes-
level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
```



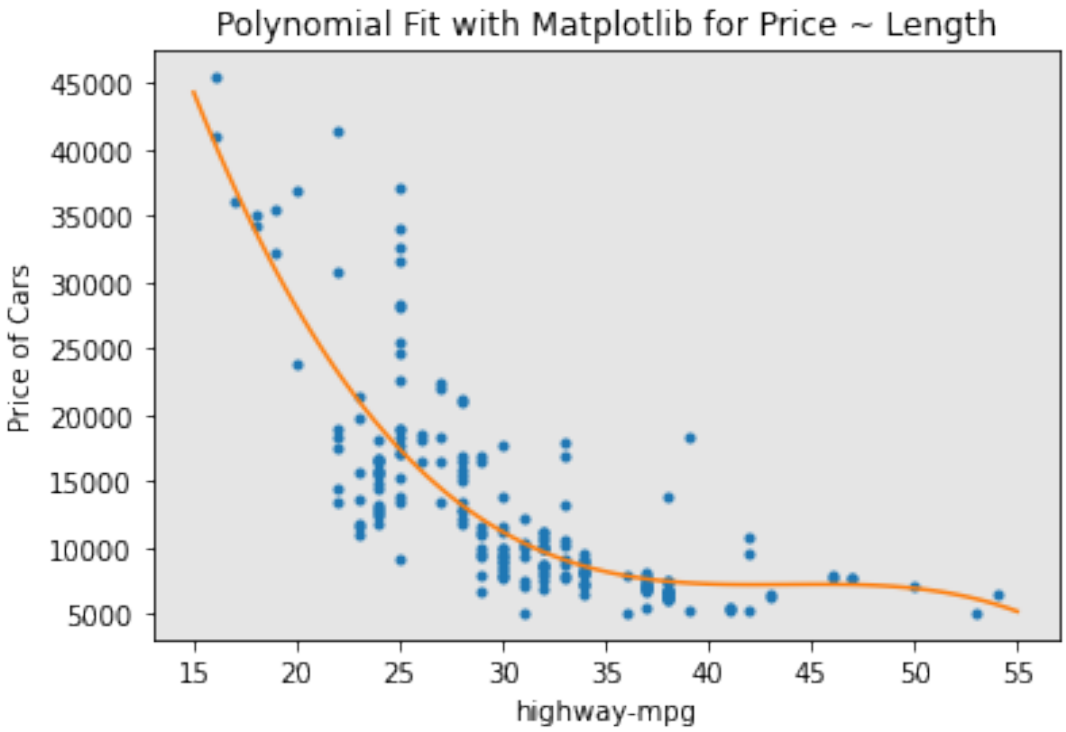
##Regresión Polinomial y Pipelines

```
#Función para gráficar los datos
def PlotPolly(model, independent_variable, dependent_variabble, Name):
    x_new = np.linspace(15, 55, 100)
    y_new = model(x_new)

    pl.plot(independent_variable, dependent_variabble, '.', x_new,
y_new, '-')
    pl.title('Polynomial Fit with Matplotlib for Price ~ Length')
    ax = pl.gca()
    ax.set_facecolor((0.898, 0.898, 0.898))
```

```
    fig = pl.gcf()
    pl.xlabel(Name)
    pl.ylabel('Price of Cars')

    pl.show()
    pl.close()

#Definimos las variables
x = df['highway-mpg']
y = df['price']

#Polinomio de 3er Orden
f = np.polyfit(x, y, 3)
p = np.poly1d(f)
print(p)
```

```
        3         2
-1.557 x + 204.8 x - 8965 x + 1.379e+05
```

```
PlotPolly(p, x, y, 'highway-mpg')
```



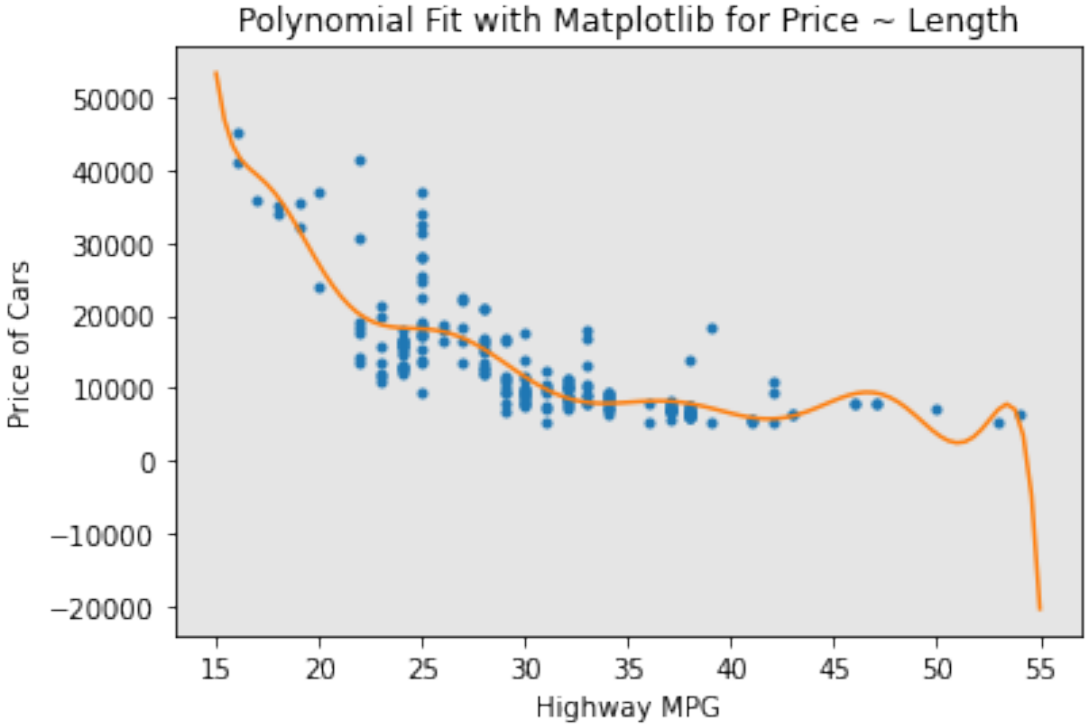

```
np.polyfit(x, y, 3)
```

```
array([-1.55663829e+00,  2.04754306e+02, -8.96543312e+03,
1.37923594e+05])
```

##Pregunta 4: Crea un polinomio de 11 orden con la variable X y Y

```
f1 = np.polyfit(x, y, 11)
p1 = np.poly1d(f1)
print(p1)
PlotPolly(p1, x, y, 'Highway MPG')
```

```
            11             10             9           8         7
-1.243e-08 x  + 4.722e-06 x  - 0.0008028 x + 0.08056 x - 5.297 x
          6        5             4             3             2
 + 239.5 x - 7588 x + 1.684e+05 x - 2.565e+06 x + 2.551e+07 x -
1.491e+08 x + 3.879e+08
```




```
#Importamos la libreria para realizar transformaciones polinomiales
sobre diferentes caracteristicas
from sklearn.preprocessing import PolynomialFeatures

#Objeto de caracteristicas polinomiales
pr = PolynomialFeatures(degree=2)
pr
```

```
PolynomialFeatures()
```

```
Z_pr = pr.fit_transform(Z)
```

```
Z.shape
```

```
(201, 4)
```

```
Z_pr.shape
```

```
(201, 15)
```

#Pipeline

```
#Importamos las librerias para el pipeline
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler

#Creamos el pipeline que incluye el modelo, estimador y su
correspondiente constructor
Input=[('scale',StandardScaler()), ('polynomial',
PolynomialFeatures(include_bias=False)), ('model',LinearRegression())]

#Ingresamos la lista, como argumento al pipeline constructor
pipe = Pipeline(Input)
pipe
```

```
Pipeline(steps=[('scale', StandardScaler()),
                ('polynomial',
PolynomialFeatures(include_bias=False)),
                ('model', LinearRegression())])
```

```
Z = Z.astype(float)
pipe.fit(Z,y)
```

```
Pipeline(steps=[('scale', StandardScaler()),
                ('polynomial',
PolynomialFeatures(include_bias=False)),
                ('model', LinearRegression())])
```

```
ypipe = pipe.predict(Z)
ypipe[0:4]
```

```
array([13102.74784201, 13102.74784201, 18225.54572197,
10390.29636555])
```

##Pegunta 5: Crea un pipeline que estandarice los datos, despues produce un predictor de regresion lineal utilizando las caracteristicas Z y el objetivo Y.

```
#Creamos la serie de pasos que lleva el pipeline
Input = [('scale',StandardScaler()), ('model', LinearRegression())]
pipe = Pipeline(Input)
pipe.fit(Z,y)
```

```
ypipe = pipe.predict(Z)
ypipe[0:10]
```

```
array([13699.11161184, 13699.11161184, 19051.65470233, 10620.36193015,
       15521.31420211, 13869.66673213, 15456.16196732, 15974.00907672,
       17612.35917161, 10722.32509097])
```

##Mediciones para la evaluación de las muestras

###Modelo de Regresion Lineal

```
#Ajuste
lm.fit(X,Y)
#Encontramos el R^2
print('The R-square is: ',lm.score(X,Y))
```

```
The R-square is:  0.4965911884339176
```

```
#Realizamos predicciones
Yhat=lm.predict(X)
print('The output of the first four predicted value is: ', Yhat[0:4])
```

```
The output of the first four predicted value is:  [16236.50464347
16236.50464347 17058.23802179 13771.3045085 ]
```

```
#Vamos a calcular el error cuadratico promedio
#necesitamos la siguiente libreria
from sklearn.metrics import mean_squared_error
```

```
#Calculamos el MSE
mse = mean_squared_error(df['price'], Yhat)
print('The mean square error of price and predicted value is: ', mse)
```

```
The mean square error of price and predicted value is:
31635042.944639888
```

##Modelo de Regresion Lineal Multiple

```
#Calulamos R^2
#Ajustamos el modelo
lm.fit(Z, df['price'])
# Find the R^2
print('The R-square is: ', lm.score(Z, df['price']))
```

```
The R-square is:  0.8093562806577457
```

```
#Realizamos la predicción
Y_predict_multifit = lm.predict(Z)
```

```
#Calculamos el MSE
print('The mean square error of price and predicted value using
multifit is: ', \
      mean_squared_error(df['price'], Y_predict_multifit))
```

```
The mean square error of price and predicted value using multifit is:
11980366.87072649
```

##Modelo de Ajustes Polinomiales

```
#Importamos la libreria para el calculo de R2-Score
from sklearn.metrics import r2_score
```

```
#Realizamos el calculo
r_squared = r2_score(y, p(x))
print('The R-square value is: ', r_squared)
```

```
The R-square value is:  0.674194666390652
```

```
#Obtenemos el MSE
mean_squared_error(df['price'], p(x))
```

```
20474146.426361218
```

#Prediccion y Toma de Decisiones

```
#Importamos librerias
import matplotlib.pyplot as plt
import numpy as np

%matplotlib inline

#Creamos datos
new_input=np.arange(1, 100, 1).reshape(-1, 1)

#Ajstamos el modelo
lm.fit(X, Y)
lm
```

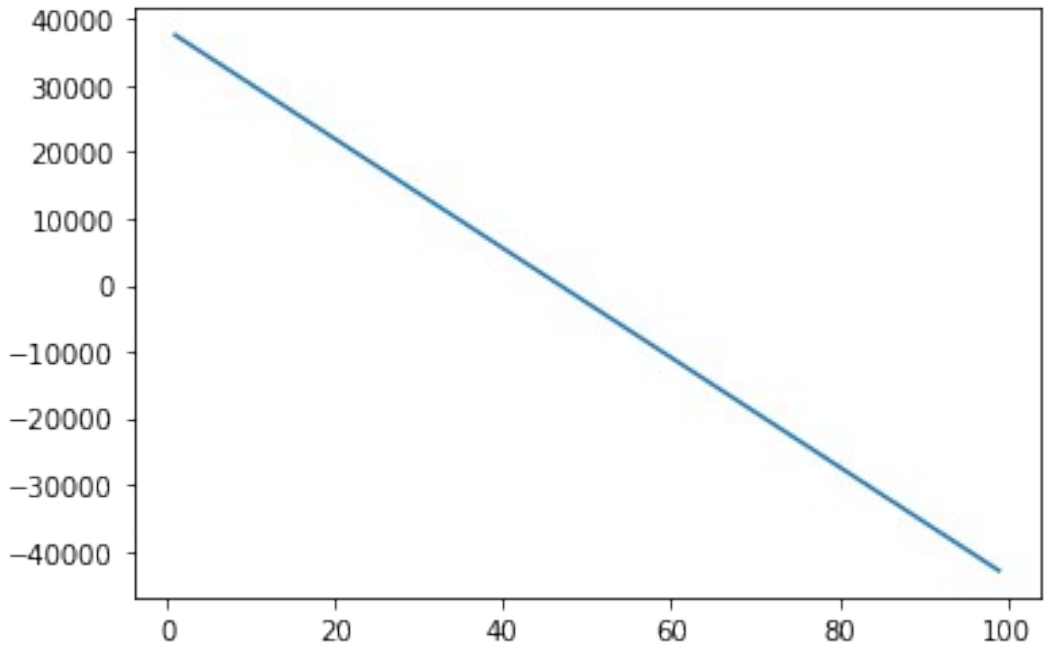
```
LinearRegression()
```

```
#Realizamos predicciones
yhat=lm.predict(new_input)
yhat[0:5]
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:451:
UserWarning: X does not have valid feature names, but LinearRegression
was fitted with feature names
  "X does not have valid feature names, but"
```

```
array([37601.57247984, 36779.83910151, 35958.10572319, 35136.37234487,
       34314.63896655])
```

```
#Graficamos
plt.plot(new_input, yhat)
plt.show()
```



##Toma de Decisiones: Determinando un buen modelo de ajuste Usuaalmetne, a mayor cantidad de variables utilizadas, mejor es nuestro modelo para predecir, pero esto no siempre es verdadero. Algunas veces no se cuentan con los suficientes datos, se puede caer en problemas numericos,o algunas variables no son utiles y actuan como ruido. Como resultado, tu debes revisar siempre el MSR y R^2.

##Conclusion La comparacion de estos tres modelos, nos indica que el modelos de regrsion lineal multiple puede ser capaz de predecir el precio de nuestro conjunto de datos. Este esultado tiene sentido debido a que tenemos 27 variables en total y sabemos que mas de una de estas variables son predictores potenciales del precio final del carro.