Perform Simple Linear Regression on Auto Dataset

We have to perform simple linear regression On Auto data sets with mpg as the dependent variable and horsepower as an independent variable.

With the help of linear regression, we can find how horsepower and mpg variables are related. How much change will occur in Mpg because of horsepower and we can also get the nature of the relationship between the variables.

Linear regression is nothing but an attempt to model a relationship between the variables. In which one will be the independent variable and the other will be the dependent variable. In other words, fitting a Regression Y= a + b*X line on the data.

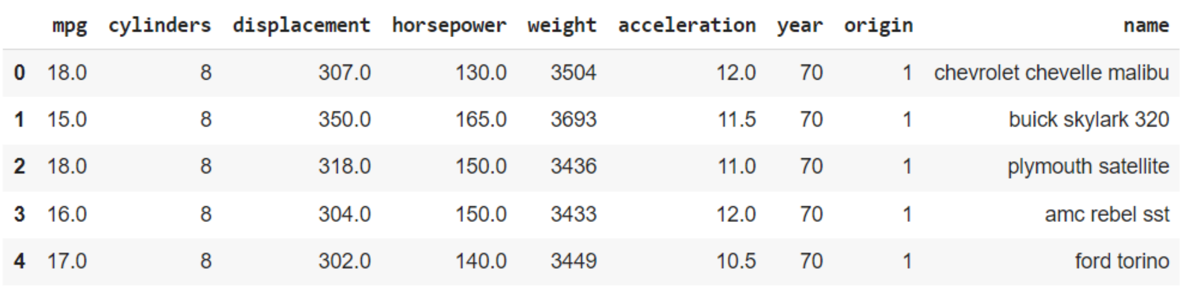
$$\mathbf{Y = a + b * X}$$

Y and X are dependent and independent variables, and a and b are parameters of the regression.

With the help of Regression, we can also predict the value of the Dependent variable with respect to any value of independent variables.

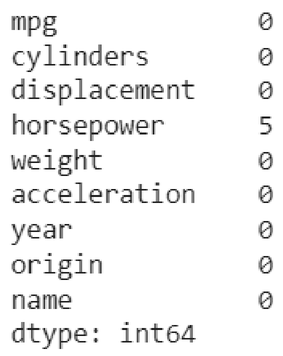
We will start the regression in python by calling our Auto datasets.

```
Auto = pd.read_csv("/content/Auto (1).csv")
Auto.head()
```



	mpg	cylinders	displacement	horsepower	weight	acceleration	year	origin	name
0	18.0	8	307.0	130.0	3504	12.0	70	1	chevrolet chevelle malibu
1	15.0	8	350.0	165.0	3693	11.5	70	1	buick skylark 320
2	18.0	8	318.0	150.0	3436	11.0	70	1	plymouth satellite
3	16.0	8	304.0	150.0	3433	12.0	70	1	amc rebel sst
4	17.0	8	302.0	140.0	3449	10.5	70	1	ford torino

After calling the data we will check whether NA value are present in the data.



```
mpg             0
cylinders       0
displacement    0
horsepower      5
weight          0
acceleration    0
year            0
origin          0
name            0
dtype: int64
```

We got the above output. From the fig, we can see that only the horsepower variable contains the NA value. So we will use the column mean on horsepower to replace the NA value with the mean of the column.

```
Auto=Auto.fillna(value=Auto['horsepower'].mean())
```

With the help of the above code, we will fill Na values in horsepower with the mean.

As we are only using two variables from the whole dataset, we will create a new dataset AutoR with only two variables Mpg and horsepower

```
AutoR=Auto[['mpg','horsepower']]
```

```
AutoR.head()
```

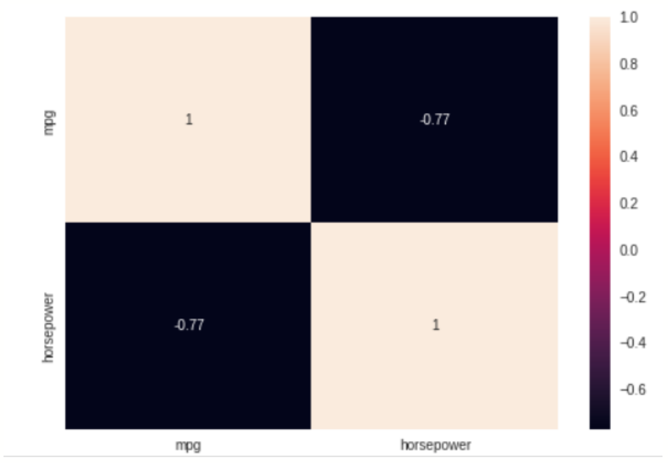
	mpg	horsepower
0	18.0	130.0
1	15.0	165.0
2	18.0	150.0
3	16.0	150.0
4	17.0	140.0

The above fig will be the output after creating of the new Dataset AutoR from the Auto dataset. This will help us to perform our regression task more easily.

Now we will see whether there is a correlation between the Mpg and horsepower.

```
AutoR.corr()
viz=sns.heatmap(AutoR.corr(),annot=True)
```

	mpg	horsepower
mpg	1.000000	-0.778427
horsepower	-0.778427	1.000000



From the above fig, we can see that there is a Negative correlation between the Mpg and horsepower. This means if we increase the value of any variable the value of the above carriable will decrease. And -0.77 indicate that there is a significant correlation between both variables.

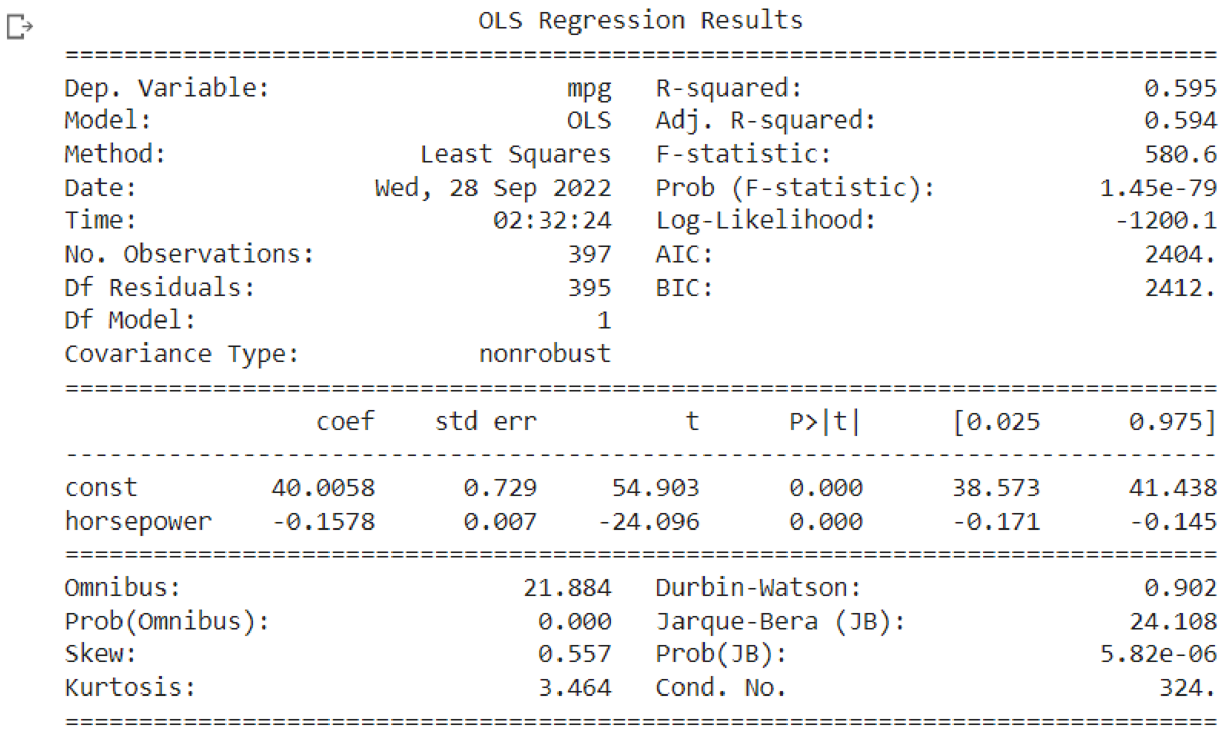
Now we will assign X and y values to perform our Regression. Remember X is the independent variable which is horsepower and y is the independent variable which is Mpg.

```
y=AutoR[['mpg']]
X=AutoR[['horsepower']]
```

Now, we have Assigned X and Y values we will now perform our Regression with the help of OLS which means Least square regression.

```
X=sm.add_constant(X)
model=sm.OLS(y,X)
results=model.fit()
print(results.summary())
```

The reason the constant is added in your regression model—**it forces the residuals to have that crucial zero mean**. Furthermore, if you don't include the constant in your regression model, you are actually setting the constant to equal zero.



```
                            OLS Regression Results                            
==============================================================================
Dep. Variable:                    mpg   R-squared:                       0.595
Model:                            OLS   Adj. R-squared:                  0.594
Method:                 Least Squares   F-statistic:                     580.6
Date:                Wed, 28 Sep 2022   Prob (F-statistic):           1.45e-79
Time:                        02:32:24   Log-Likelihood:                -1200.1
No. Observations:                 397   AIC:                             2404.
Df Residuals:                     395   BIC:                             2412.
Df Model:                           1                                         
Covariance Type:            nonrobust                                         
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         40.0058      0.729     54.903      0.000      38.573      41.438
horsepower    -0.1578      0.007    -24.096      0.000      -0.171      -0.145
==============================================================================
Omnibus:                       21.884   Durbin-Watson:                   0.902
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               24.108
Skew:                           0.557   Prob(JB):                     5.82e-06
Kurtosis:                       3.464   Cond. No.                         324.
==============================================================================
```

The above table is a summary of the linear regression we performed on Mpg as the dependent variable and horsepower as an independent variable.

After performing the Stats model OLS we got this table in which we got 397 observations which we will explain.

We can see **DF residual** in the table which is also called the **Degree of freedom of the residuals**. It is calculated with the formula n-k-1 where n is the number of observations, and k is the number of independent variables. We have 397 observations as we can see in the table above. We have used 1 independent variable which we can verify with **DF model** in the table which tells us how many independent variables we have used to perform this regression. So, if we calculate **DF residual** by the formula, we get **397-1-1 = 395** same as we can see in the table.

Covariance type tells the relationship between the independent and dependent variables. As there are less outliers and there is significance relationship between the variables which means we are getting the regression line on the graph the table show **non robust** Covariance Type.

Now we have **R squared** value which tells us how much variation in dependent variables can be explained through independent variables. As our R squared value is 0.595 which means 59.5 % variation in Y can be explained from X. maximum value of R can be 1 so the high R squared value means the regression is good.

The constant term is the intercept which we can use to draw a regression line. During regression, OLS removes some values of the independent variable which do not have much impact on the dependent variable. So, the constant is the average of the values which were removed.

The coefficient term of this regression is -0.1578, it tells that if the independent variable rises by 1 unit we will see a decrease in the dependent Variable by -0.1578. Which tells the negative relationship between independent and dependent variables.

To understand the P value significance we have to first understand the Null hypothesis. According to Null hypothesis there is no correlation between the variables. If P values is $P<0.05$ the Null hypothesis is rejected which means there is relationship between variables. If wee see the table P value is 0.0 which means Null hypothesis is rejected and there is relationship between the variables.

The above terms are the relevant term that we required for this Question. The above terms tell the nature of the relationship between the independent and dependent variables.

```
from sklearn import linear_model


model_sl = linear_model.LinearRegression(fit_intercept=
True)

x_train = Auto['horsepower'].values.reshape(-1, 1)
y_train = Auto['mpg']
model_sl.fit(x_train, y_train)


model_sl.intercept_, model_sl.coef_
model_sl.predict(np.array([98]).reshape(-1, 1))
```

```
array([24.5370278])
```

Now we have done the regression we can predict the value of Mpg for any value of horsepower. Before the prediction, we will use a linear model on the OLS which will give us intercept and coefficient which will help us to predict the Mpg value.

As you can see we are predicting the value of Mpg when the value of horsepower is 98. From the above fig, we can see that value of mpg is 24.53 when horsepower is 98.

We can Predict the value of Mpg for any value of Horsepower. Let’s predict the value of mpg when horsepower is 200.

```
model_sl.predict(np.array([200]).reshape(-1, 1))
```

```
array([8.436865])
```

So the value of Mpg when horsepower is 200 is 8.43.

Now if we compare the two Answers we can see that there is a negative correlation between Mpg and Horsepower. With the increase in horsepower value, there is the decrease in the value of mpg.

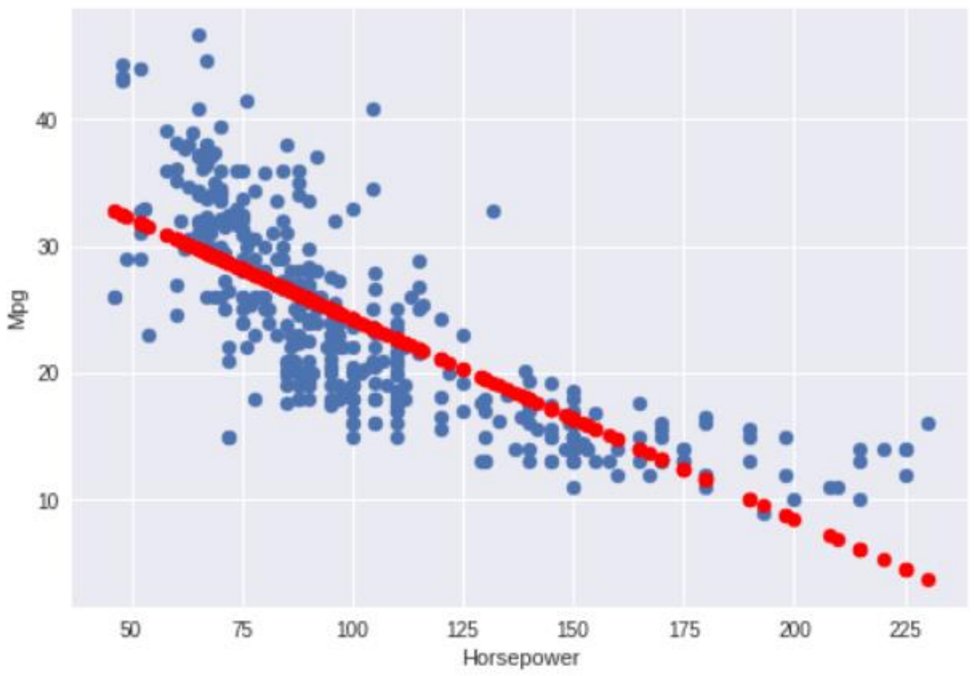
Now we have to to find the 95% confidence and prediction intervals of our Data.

```
        mean    mean_se  mean_ci_lower  mean_ci_upper  obs_ci_lower  \
0  24.537028  0.253797      24.038067      25.035989     14.722193   

   obs_ci_upper  
0     34.351862  
```

From above fig we can see that 95% confidence interval our from 14.722 to 34.35 and the prediction interval is from 24.03 to 24.035

There are many libraries in python from which we can plot the graph. Now we will plot the regression line .

```
plt.scatter(x_train,y_train)
plt.scatter(x_train,ypred, color='red')
plt.show()
```



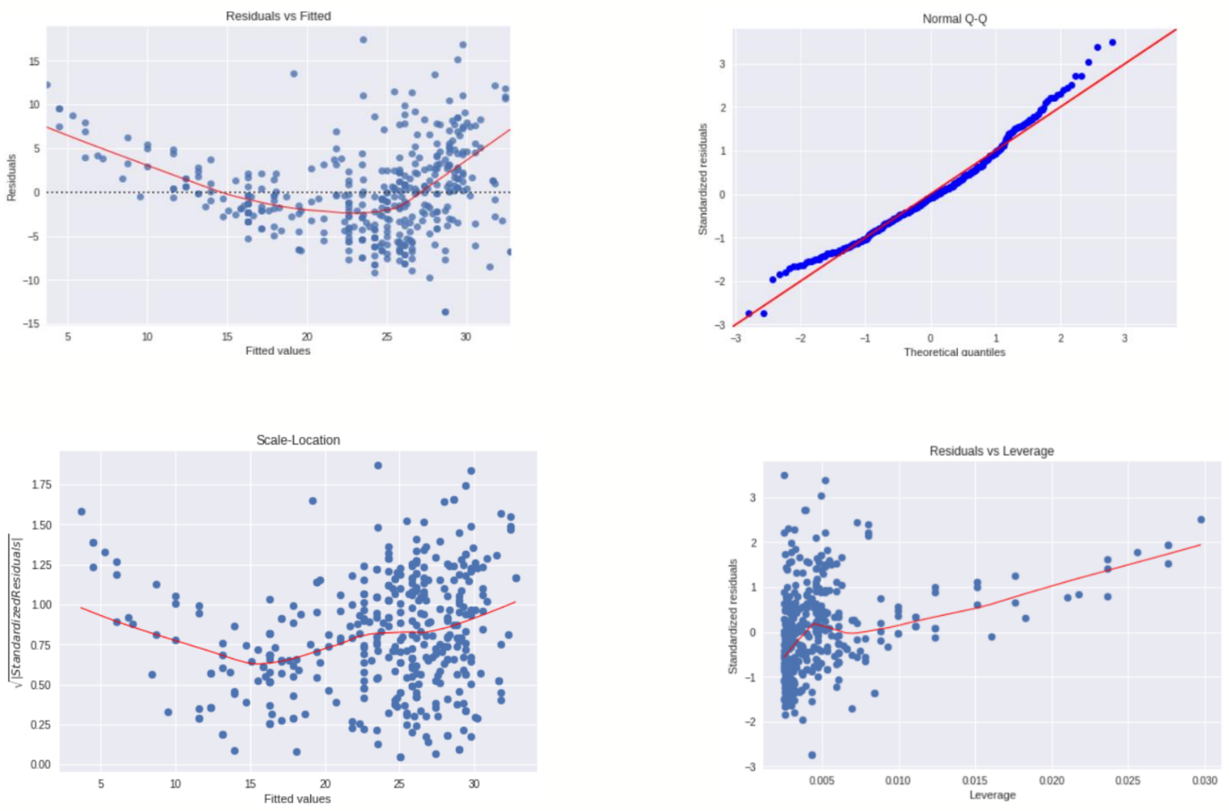
A regression line can be used to predict the value of y for a given value of x. Regression analysis identifies a regression line. The regression line shows **how much and in what direction the response variable changes when the explanatory variable changes**.

The fig shows that the red line is the regression line, which shows the value of Mpg decrease when the value of Mpg increases. This shows the slope is negative which tells about the negative relationship between the two variables.

Diagnostic plots.

```
model1_fitted_y = results.fittedvalues
model1_norm_residuals = results.get_influence().resid_studentized_internal
model1_norm_residuals_abs_sqrt = np.sqrt(np.abs(model1_norm_residuals))
model1_leverage = results.get_influence().hat_matrix_diag

# plot 1: residuals vs. fitted values
plot1 = plt.figure(1)
sns.residplot(model1_fitted_y, Auto['mpg'], lowess=True, line_kws={'color'
:'red', 'lw':1})
plot1.axes[0].set_title('Residuals vs Fitted')
plot1.axes[0].set_xlabel('Fitted values')
plot1.axes[0].set_ylabel('Residuals')

# plot 2: normal Q-Q
plot2 = sm.qqplot(model1_norm_residuals, fit=True, line='45')
plot2.axes[0].set_title('Normal Q-Q')
plot2.axes[0].set_xlabel('Theoretical quantiles')
plot2.axes[0].set_ylabel('Standardized residuals');

# plot 3: scale-location
plot3 = plt.figure(3)
plt.scatter(model1_fitted_y, model1_norm_residuals_abs_sqrt)
sns.regplot(model1_fitted_y, model1_norm_residuals_abs_sqrt, scatter=False
, ci=False, lowess=True, line_kws={'color':'red', 'lw':1})
plot3.axes[0].set_title('Scale-Location')
plot3.axes[0].set_xlabel('Fitted values')
plot3.axes[0].set_ylabel('$\sqrt{|Standardized Residuals|}$');

# plot 4: residuals vs. leverage
plot4 = plt.figure(4)
```

```
plt.scatter(model1_leverage, model1_norm_residuals)
sns.regplot(model1_leverage, model1_norm_residuals, scatter=False, ci=False, lowess=True, line_kws={'color':'red', 'lw':1})
plot4.axes[0].set_title('Residuals vs Leverage')
plot4.axes[0].set_xlabel('Leverage')
plot4.axes[0].set_ylabel('Standardized residuals')
```



As we have to see whether the residual exhibit nonlinear pattern we will analyze the residual vs fitted plot.
If the red line in the center of the plot follows the horizontal pattern we say that the residual has a linear pattern. If we analyze the plot we can see that the red line is horizontal but roughly deviate but it is not too much that means the residual follows a linear pattern and the dataset is appropriate for linear regression

The Normal Q-Q whether the residual is normally distributed or not.If the residual falls on the straight diagonal line means residual are normally distributed.

If we analyze our plot we can see that residual falls on the straight line , but some residuals deviate at the end of the line but they are not many , so we can declare that rsiduals are normally distributed.

The Standarised Residual plot is to check assumption of equal variance among the the residual in our regression models.
By the definition the red line on the plot whould follow horzontal pattern and if we see our plot the red line follow the horizontal path with very little deviation , so we can say that assumption of equal variance were not violated in this case.

The map shows the influential points on the plot. The plot which changes the direction of the line have more influential. We can see the point far from other may have more influential but also have high redidual.

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import linear_model
import sklearn
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
import statsmodels.api as sm
import seaborn as sns
import plotly.express as px
```

In [2]:
```
Auto = pd.read_csv("/content/Auto (1).csv")
Auto.head()
```

Out[2]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	year	origin	name
0	18.0	8	307.0	130.0	3504	12.0	70	1	chevrolet chevelle malibu
1	15.0	8	350.0	165.0	3693	11.5	70	1	buick skylark 320
2	18.0	8	318.0	150.0	3436	11.0	70	1	plymouth satellite
3	16.0	8	304.0	150.0	3433	12.0	70	1	amc rebel sst
4	17.0	8	302.0	140.0	3449	10.5	70	1	ford torino

In [3]:
```
Auto.isnull().sum()
```

Out[3]:
```
mpg             0
cylinders       0
displacement    0
horsepower      5
weight          0
acceleration    0
year            0
origin          0
name            0
dtype: int64
```

In [4]:
```
Auto=Auto.fillna(value=Auto['horsepower'].mean())
```

In [5]:
```
Auto.isnull().sum()
```

Out[5]:
```
mpg             0
cylinders       0
displacement    0
horsepower      0
weight          0
acceleration    0
year            0
origin          0
name            0
dtype: int64
```

In [6]:
```
AutoR=Auto[['mpg','horsepower']]
AutoR.head()
```

Out[6]:

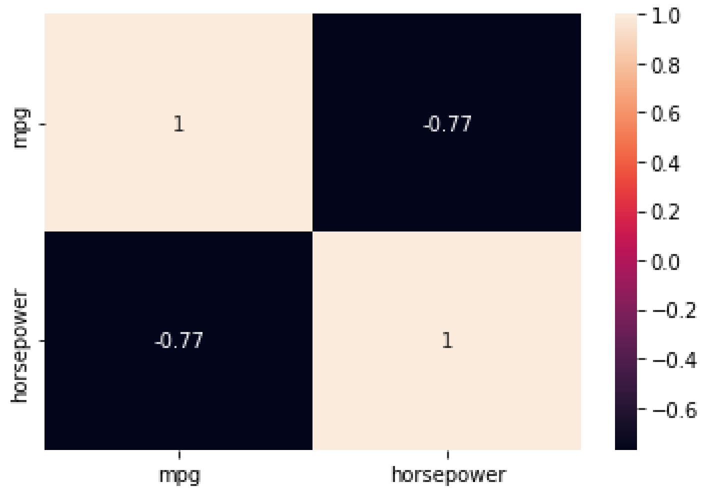
	mpg	horsepower
0	18.0	130.0
1	15.0	165.0
2	18.0	150.0
3	16.0	150.0
4	17.0	140.0

In [7]:
```
AutoR.corr()
```

Out[7]:

	mpg	horsepower
mpg	1.000000	-0.771441
horsepower	-0.771441	1.000000

In [8]:
```
viz=sns.heatmap(AutoR.corr(),annot=True)
```



In [9]:
```
y=AutoR[['mpg']]
X=AutoR[['horsepower']]
```

In [10]:
```
X=sm.add_constant(X)
model=sm.OLS(y,X)
results=model.fit()
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    mpg   R-squared:                       0.595
Model:                            OLS   Adj. R-squared:                  0.594
Method:                 Least Squares   F-statistic:                     580.6
Date:                Mon, 03 Oct 2022   Prob (F-statistic):           1.45e-79
Time:                        01:05:35   Log-Likelihood:                -1200.1
No. Observations:                 397   AIC:                             2404.
Df Residuals:                     395   BIC:                             2412.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         40.0058      0.729     54.903      0.000      38.573      41.438
horsepower    -0.1578      0.007    -24.096      0.000      -0.171      -0.145
==============================================================================
Omnibus:                       21.884   Durbin-Watson:                   0.902
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               24.108
Skew:                           0.557   Prob(JB):                     5.82e-06
Kurtosis:                       3.464   Cond. No.                         324.
==============================================================================

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

/usr/local/lib/python3.7/dist-packages/statsmodels/tsa/tsatools.py:142: FutureW
arning: In a future version of pandas all arguments of concat except for the ar
gument 'objs' will be keyword-only
  x = pd.concat(x[::order], 1)
```

In [11]:
```
xnew = np.array([[1., 98.]])
ynew = results.get_prediction(xnew)
print(ynew.summary_frame(alpha=0.05))
```

```
        mean   mean_se  mean_ci_lower  mean_ci_upper  obs_ci_lower  \
0  24.537028  0.253797      24.038067      25.035989     14.722193

   obs_ci_upper
0     34.351862
```

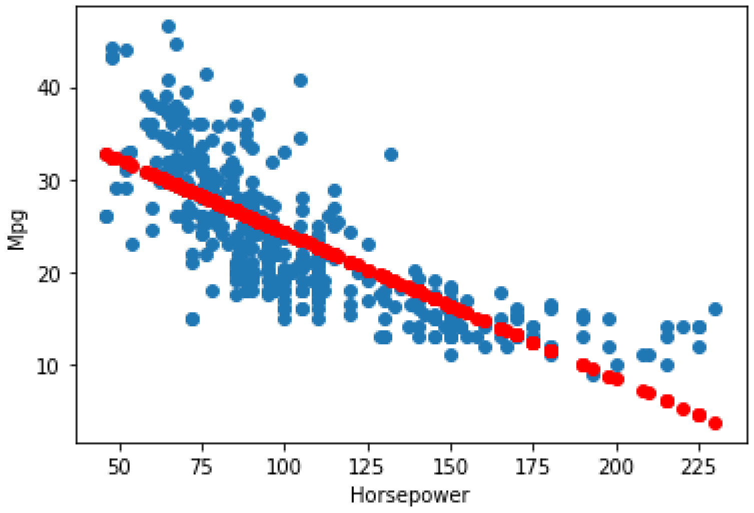
In [12]:
```
ypred = results.predict(X)
ypred
```

Out[12]:
```
0      19.485996
1      13.961431
2      16.329102
3      16.329102
4      17.907549
         ...
392    26.431165
393    31.797886
394    26.746854
395    27.536078
396    27.062544
Length: 397, dtype: float64
```

In [17]:
```
from sklearn import linear_model


model_sl = linear_model.LinearRegression(fit_intercept=True)

x_train = Auto['horsepower'].values.reshape(-1, 1)
y_train = Auto['mpg']
model_sl.fit(x_train, y_train)


model_sl.predict(np.array([98]).reshape(-1, 1))
```

Out[17]:
```
array([24.5370278])
```

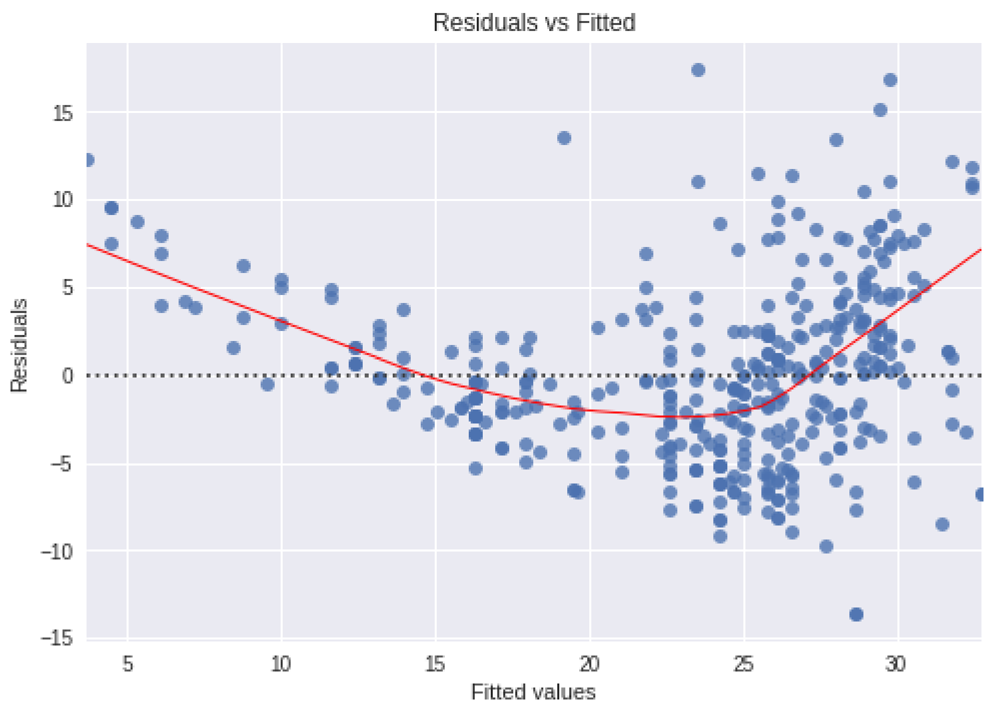
In [14]:
```
plt.scatter(x_train,y_train)
plt.scatter(x_train,ypred, color='red')
plt.xlabel("Horsepower")
plt.ylabel("Mpg")
plt.show()
```

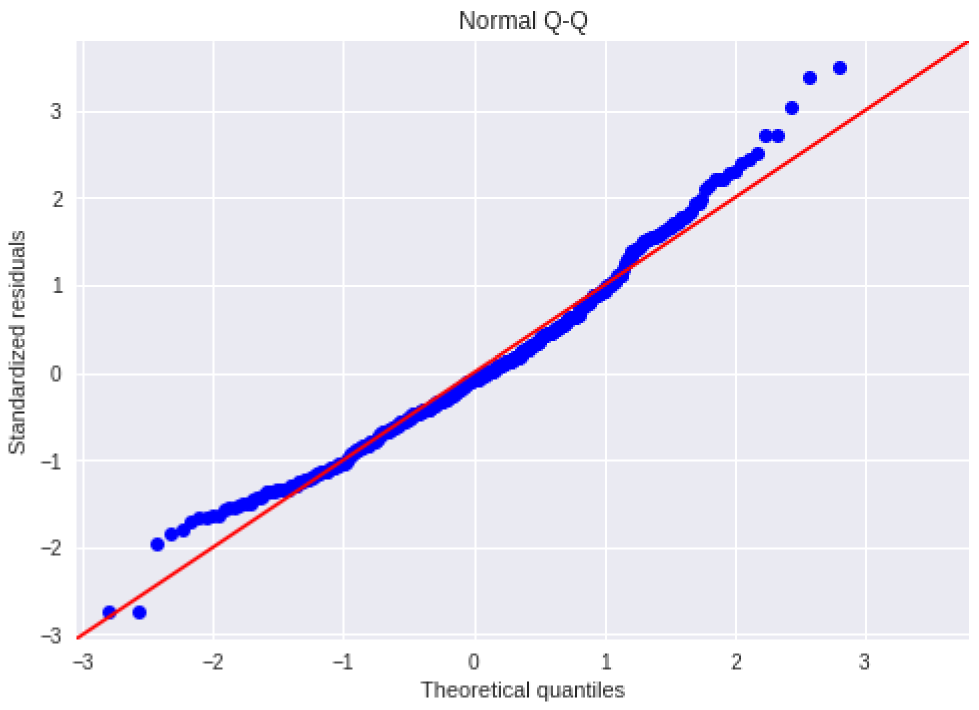


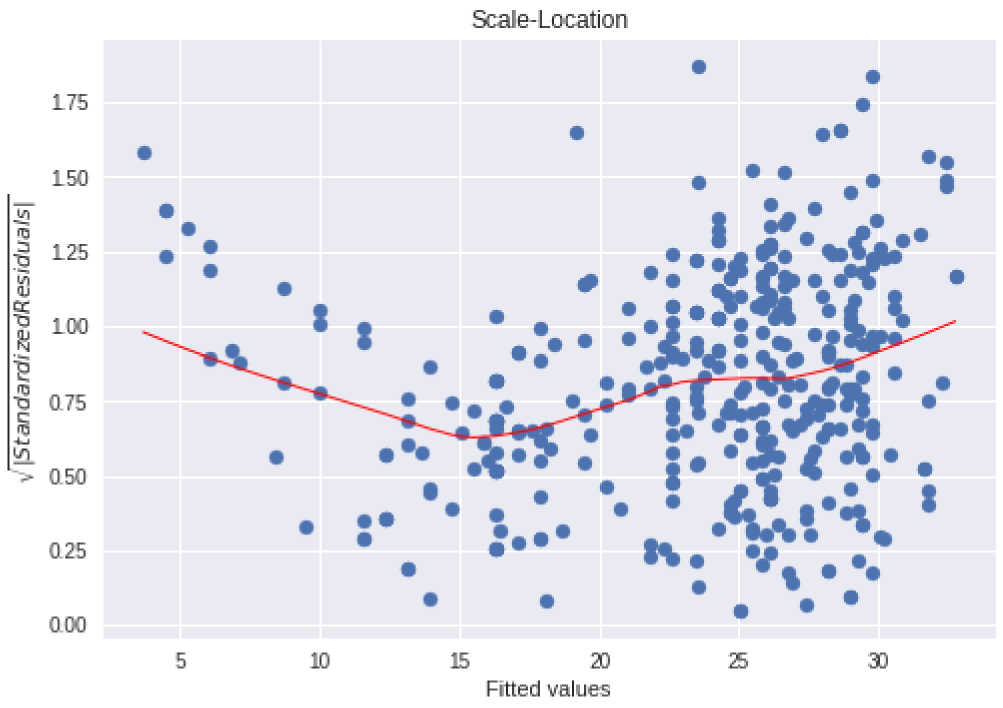
In [18]:
```
model1_fitted_y = results.fittedvalues
model1_norm_residuals = results.get_influence().resid_studentized_internal
model1_norm_residuals_abs_sqrt = np.sqrt(np.abs(model1_norm_residuals))
model1_leverage = results.get_influence().hat_matrix_diag

# plot 1: residuals vs. fitted values
plot1 = plt.figure(1)
sns.residplot(model1_fitted_y, Auto['mpg'], lowess=True, line_kws={'color':'red',
plot1.axes[0].set_title('Residuals vs Fitted')
plot1.axes[0].set_xlabel('Fitted values')
plot1.axes[0].set_ylabel('Residuals')

# plot 2: normal Q-Q
plot2 = sm.qqplot(model1_norm_residuals, fit=True, line='45')
plot2.axes[0].set_title('Normal Q-Q')
plot2.axes[0].set_xlabel('Theoretical quantiles')
plot2.axes[0].set_ylabel('Standardized residuals');

# plot 3: scale-location
plot3 = plt.figure(3)
plt.scatter(model1_fitted_y, model1_norm_residuals_abs_sqrt)
sns.regplot(model1_fitted_y, model1_norm_residuals_abs_sqrt, scatter=False, ci=Fa
plot3.axes[0].set_title('Scale-Location')
plot3.axes[0].set_xlabel('Fitted values')
plot3.axes[0].set_ylabel('$\sqrt{|Standardized Residuals|}$');

# plot 4: residuals vs. leverage
plot4 = plt.figure(4)
plt.scatter(model1_leverage, model1_norm_residuals)
sns.regplot(model1_leverage, model1_norm_residuals, scatter=False, ci=False, lowe
plot4.axes[0].set_title('Residuals vs Leverage')
plot4.axes[0].set_xlabel('Leverage')
plot4.axes[0].set_ylabel('Standardized residuals')
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarnin
g: Pass the following variables as keyword args: x, y. From version 0.12, the o
nly valid positional argument will be `data`, and passing other arguments witho
ut an explicit keyword will result in an error or misinterpretation.
  FutureWarning
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarnin
g: Pass the following variables as keyword args: x, y. From version 0.12, the o
nly valid positional argument will be `data`, and passing other arguments witho
ut an explicit keyword will result in an error or misinterpretation.
  FutureWarning
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarnin
g: Pass the following variables as keyword args: x, y. From version 0.12, the o
nly valid positional argument will be `data`, and passing other arguments witho
ut an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

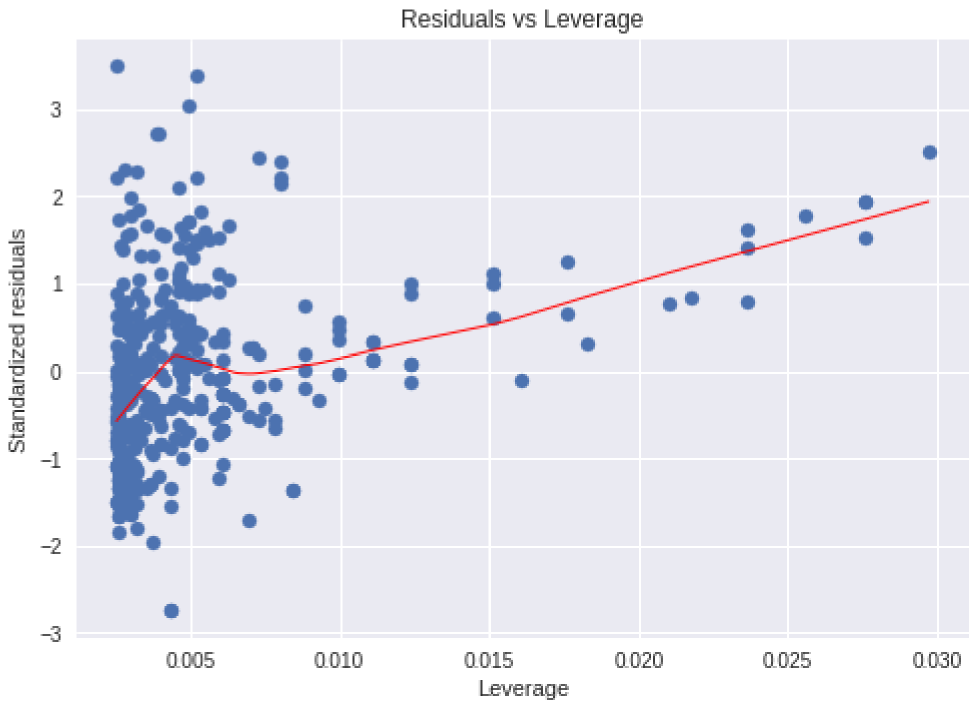
Out[18]:
```
Text(0, 0.5, 'Standardized residuals')
```


Residuals vs Fitted
Residuals
Fitted values
15
10
5
0
−5
−10
−15
5
10
15
20
25
30


Normal Q-Q
Standardized residuals
Theoretical quantiles
3
2
1
0
−1
−2
−3
−3
−2
−1
0
1
2
3



Scale-Location
$\sqrt{|Standardized Residuals|}$
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
5
10
15
20
25
30
Fitted values



Residuals vs Leverage
Standardized residuals
3
2
1
0
−1
−2
−3
0.005
0.010
0.015
0.020
0.025
0.030
Leverage