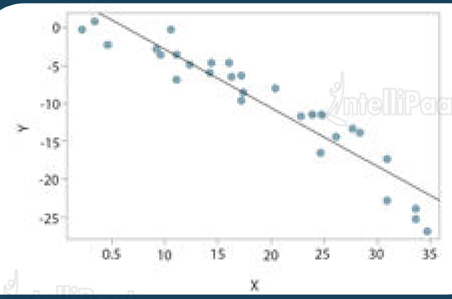
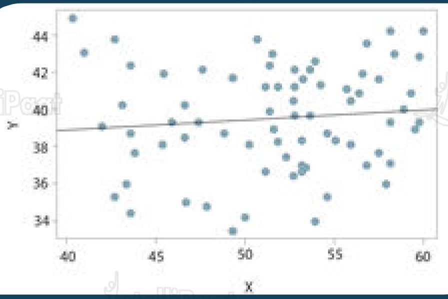
Assumptions in Linear Regression

Linearity

There should be ***linear*** and ***additive*** relationship between the dependent & independent variables!



Satisfies the assumption

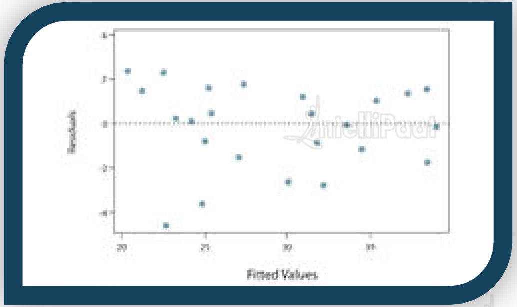


Doesn't satisfy the assumption

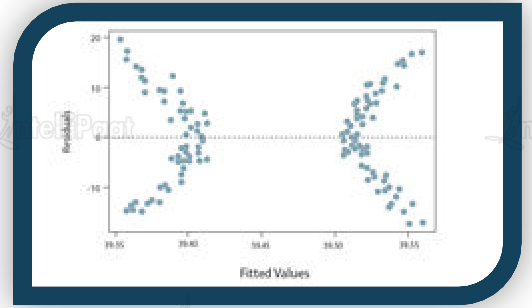
Assumptions in Linear Regression

Equal Error Variance

The residuals must have ***constant variance!***



If there is no pattern, data is random and hence, satisfies the condition



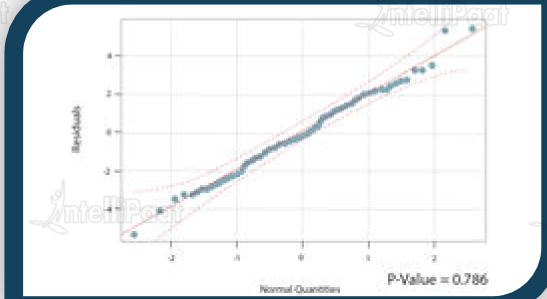
If there is a pattern, the data is not random & hence, doesn't satisfy the condition

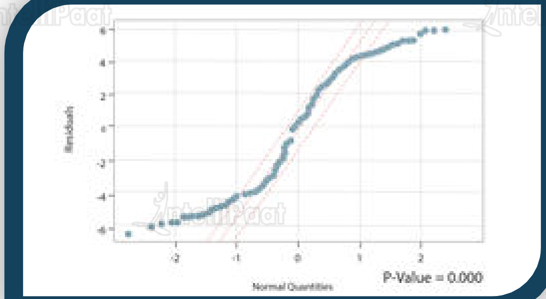
Assumptions in Linear Regression

Normality of Errors

The residuals must be ***normally distributed!***

Build a normal probability plot, if the residuals are closer to the fit line, the more normal they are





Hands-on: Linear Regression

Demo- Linear Regression

- We will be using the Boston House Prices Dataset, with 506 rows and 13 attributes with a target column.
- In this example we will train two models to predict the price.
- First, considering 'lstat' as independent and 'medv' as dependent variable.
- Second, considering 'medv' as dependent and the rest of the attributes as independent.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat	medv
2	0.00632	18	2.31	0	0.538	6.575	65.2	4.09	1	296	15.3	396.9	4.98	24
3	0.02731	0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.9	9.14	21.6
4	0.02729	0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
5	0.03237	0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
6	0.06905	0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.9	5.33	36.2
7	0.02985	0	2.18	0	0.458	6.43	58.7	6.0622	3	222	18.7	394.12	5.21	28.7
8	0.08829	12.5	7.87	0	0.524	6.012	66.6	5.5605	5	311	15.2	395.6	12.43	22.9
9	0.14455	12.5	7.87	0	0.524	6.172	96.1	5.9505	5	311	15.2	396.9	19.15	27.1
10	0.21124	12.5	7.87	0	0.524	5.631	100	6.0821	5	311	15.2	386.63	29.93	16.5
11	0.17004	12.5	7.87	0	0.524	6.004	85.9	6.5921	5	311	15.2	386.71	17.1	18.9
12	0.22489	12.5	7.87	0	0.524	6.377	94.3	6.3467	5	311	15.2	392.52	20.45	15
13	0.11747	12.5	7.87	0	0.524	6.009	82.9	6.2267	5	311	15.2	396.9	13.27	18.9
14	0.09378	12.5	7.87	0	0.524	5.889	39	5.4509	5	311	15.2	390.5	15.71	21.7
15	0.62976	0	8.14	0	0.538	5.949	61.8	4.7075	4	307	21	396.9	8.26	20.4
16	0.63796	0	8.14	0	0.538	6.096	84.5	4.4619	4	307	21	380.02	10.26	18.2
17	0.62739	0	8.14	0	0.538	5.834	56.5	4.4986	4	307	21	395.62	8.47	19.9
18	1.05393	0	8.14	0	0.538	5.935	29.3	4.4986	4	307	21	386.85	6.58	23.1
19	0.7842	0	8.14	0	0.538	5.99	81.7	4.2579	4	307	21	386.75	14.67	17.5

Simple Linear Regression

Loading the Boston dataset:

1

```
In [1]: import pandas as pd
        data = pd.read_csv('Boston1.csv')
```

Having a glance at the shape:

2

```
In [4]: data.head()
```

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	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
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	36.2

Simple Linear Regression

Having a glance at the dependent and independent variables:

1

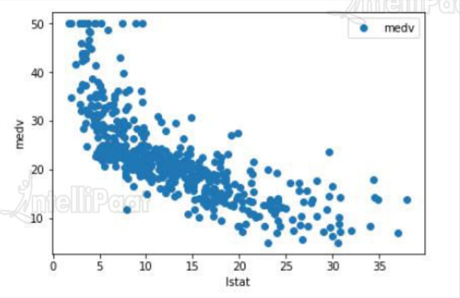
```
In [6]: data_ = data.loc[:,['lstat','medv']]
        data_.head(5)
```

	lstat	medv
0	4.98	24.0
1	9.14	21.6
2	4.03	34.7
3	2.94	33.4
4	5.33	36.2

Visualizing the change in the variables

2

```
In [10]: import matplotlib.pyplot as plt
         data.plot(x='lstat',y='medv',style='o')
         plt.xlabel('lstat')
         plt.ylabel('medv')
         plt.show()
```



Simple Linear Regression

Divide the data into independent and dependent variables:

1

```
In [13]: X = pd.DataFrame(data['lstat'])
         y = pd.DataFrame(data['medv'])
```

Split the data into train and test set:

2

```
In [14]: from sklearn.model_selection import train_test_split
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1)
```

Shape of the train and test sets:

3

```
In [16]: print(X_train.shape)
         print(X_test.shape)
         print(y_train.shape)
         print(y_test.shape)
```

```
(404, 1)
(102, 1)
(404, 1)
(102, 1)
```

Simple Linear Regression

Train the algorithm:

1

```
In [11]: from sklearn.linear_model import LinearRegression
         regressor = LinearRegression()
         regressor.fit(X_train, y_train)
```

Retrieve the intercept:

2

```
In [12]: print(regressor.intercept_)
```

```
[34.33497839]
```

Retrieve the slope:

3

```
In [13]: print(regressor.coef_)
```

```
[[-0.92441715]]
```

Simple Linear Regression

Predicted value:

1
```
In [23]: y_pred
```

Predicted
27.374117
27.697663
16.955936
26.847199
24.915168
24.055460
29.990218

Actual value:

2
```
In [25]: y_test
```

medv
28.2
23.9
16.6
22.0
20.8
23.0
27.9

Simple Linear Regression

Evaluating the Algorithm:

```
In [26]: from sklearn import metrics
         import numpy as np
         print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
         print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
         print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

```
Mean Absolute Error: 5.078127727696938
Mean Squared Error: 46.99482091954711
Root Mean Squared Error: 6.855276866731723
```

Multiple Linear Regression

Loading the Boston dataset again:

```
In [1]: import pandas as pd
        import numpy as np
        dataset = pd.read_csv('Boston1.csv')
        dataset
```

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	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
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	36.2
5	0.02985	0.0	2.18	0	0.458	6.430	58.7	6.0622	3	222	18.7	394.12	5.21	28.7
6	0.08829	12.5	7.87	0	0.524	6.012	66.6	5.5605	5	311	15.2	395.60	12.43	22.9
7	0.14455	12.5	7.87	0	0.524	6.172	96.1	5.9505	5	311	15.2	396.90	19.15	27.1
8	0.21124	12.5	7.87	0	0.524	5.631	100.0	6.0821	5	311	15.2	386.63	29.93	16.5
9	0.17004	12.5	7.87	0	0.524	6.004	85.9	6.5921	5	311	15.2	386.71	17.10	18.9
10	0.22489	12.5	7.87	0	0.524	6.377	94.3	6.3467	5	311	15.2	392.52	20.45	15.0
11	0.11747	12.5	7.87	0	0.524	6.009	82.9	6.2267	5	311	15.2	396.90	13.27	18.9
12	0.09378	12.5	7.87	0	0.524	5.889	39.0	5.4509	5	311	15.2	390.50	15.71	21.7
13	0.62976	0.0	8.14	0	0.538	5.949	61.8	4.7075	4	307	21.0	396.90	8.26	20.4
14	0.63796	0.0	8.14	0	0.538	6.096	84.5	4.4619	4	307	21.0	380.02	10.26	18.2
15	0.62739	0.0	8.14	0	0.538	5.834	56.5	4.4986	4	307	21.0	395.62	8.47	19.9

Multiple Linear Regression

Setting up the dependent and the independent variables:

1

```
In [3]: X = pd.DataFrame(dataset.iloc[:,:-1])
        y = pd.DataFrame(dataset.iloc[:,-1])
```

Having a glance at the independent variable:

2

```
In [4]: X
```

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat
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	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33
5	0.02985	0.0	2.18	0	0.458	6.430	58.7	6.0622	3	222	18.7	394.12	5.21
6	0.08829	12.5	7.87	0	0.524	6.012	66.6	5.5605	5	311	15.2	395.60	12.43
7	0.14455	12.5	7.87	0	0.524	6.172	96.1	5.9505	5	311	15.2	396.90	19.15
8	0.21124	12.5	7.87	0	0.524	5.631	100.0	6.0821	5	311	15.2	386.63	29.93
9	0.17004	12.5	7.87	0	0.524	6.004	85.9	6.5921	5	311	15.2	386.71	17.10

Having a glance at the dependent variable:

3

```
In [5]: y
```

	medv
0	24.0
1	21.6
2	34.7
3	33.4
4	36.2
5	28.7
6	22.9
7	27.1
8	16.5
9	18.9

Multiple Linear Regression

Split the data into train and test sets:

1

```
In [6]: from sklearn.model_selection import train_test_split
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=5)
```

Having a glance at the shape of the train and test set :

2

```
In [7]: print(X_train.shape)
        print(X_test.shape)
        print(y_train.shape)
        print(y_test.shape)
```

```
(404, 13)
(102, 13)
(404, 1)
(102, 1)
```

Multiple Linear Regression

Train the algorithm:

1

```
In [8]: from sklearn.linear_model import LinearRegression
        regressor = LinearRegression()
        regressor.fit(X_train, y_train)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
        normalize=False)
```

Having a look at the coefficients that the model has chosen:

2

```
In [9]: v = pd.DataFrame(regressor.coef_,index=['Co-efficient']).transpose()
        w = pd.DataFrame(X.columns, columns=['Attribute'])
```

Concatenating the DataFrames to compare:

3

```
In [12]: coeff_df = pd.concat([w,v], axis=1, join='inner')
         coeff_df
```

	Attribute	Co-efficient
0	crim	-0.130800
1	zn	0.049403
2	indus	0.001095
3	chas	2.705366
4	nox	-15.957050
5	rm	3.413973
6	age	0.001119
7	dis	-1.493081
8	rad	0.364422
9	tax	-0.013172
10	ptratio	-0.952370
11	black	0.011749
12	lstat	-0.594076

Multiple Linear Regression

Comparing the predicted value to the actual value:

1

```
In [13]: y_pred = regressor.predict(X_test)
         y_pred = pd.DataFrame(y_pred, columns=['Predicted'])
         y_pred
```

Predicted
37.563118
32.144451
27.065736
5.670806
35.099826
5.858037
27.537085
31.810192
26.356348
22.772087
31.911830

2

```
In [14]: y_test
```

medv
37.6
27.9
22.6
13.8
35.2
10.4
23.9
29.0
22.8
23.2
33.2

Multiple Linear Regression

Evaluating the Algorithm:

```
In [15]: from sklearn import metrics
         print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
         print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
         print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

```
Mean Absolute Error: 3.2132704958423757
Mean Squared Error: 20.86929218377072
Root Mean Squared Error: 4.568292042303198
```