In [0]:
```
#This is a supplementary material to the lecture "Linear Regression" to quickly revise, whenever neede
d
```

In [0]:
```
#Linear Regression tries to find a linear relation between the input features (say 'x') and the contin
uous spectrum output (say 'y')
#In other words, it tries to find the best possible line to fit the training data points
#after that, it uses the same linear function to predict the output for the unseen data
```

In [0]:
```
#Linear regression will learn some coefficients and an intercept for the linear line and then, uses th
e same parameters to predict on new input values
```

In [0]:
```
#Let's take an example of california housing dataset available in sklearn
```

In [0]:
```
#import packages
import numpy as np
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

In [0]:
```
#Load the data
housing = datasets.fetch_california_housing()
housing
```

```
Downloading Cal. housing from https://ndownloader.figshare.com/files/5976036 to /root/scikit_learn_da
ta
```

Out[0]:
```
{'DESCR': '.. _california_housing_dataset:\n\nCalifornia Housing dataset\n--------------------------
\n\n**Data Set Characteristics:**\n\n    :Number of Instances: 20640\n\n    :Attribute Information:\n        - MedInc
median income in block\n        - HouseAge      median house age in block\n        - AveRooms      av
erage number of rooms\n        - AveBedrms     average number of bedrooms\n        - Population    bl
ock population\n        - AveOccup      average house occupancy\n        - Latitude      house block
latitude\n        - Longitude     house block longitude\n\n    :Missing Attribute Values: None\n\nThi
s dataset was obtained from the StatLib repository.\nhttp://lib.stat.cmu.edu/datasets/\n\nThe target
variable is the median house value for California districts.\n\nThis dataset was derived from the 199
0 U.S. census, using one row per census\nblock group. A block group is the smallest geographical unit
for which the U.S.\nCensus Bureau publishes sample data (a block group typically has a population\nof
600 to 3,000 people).\n\nIt can be downloaded/loaded using the\n:func:`sklearn.datasets.fetch_califor
nia_housing` function.\n\n.. topic:: References\n\n    - Pace, R. Kelley and Ronald Barry, Sparse Spa
tial Autoregressions,\n      Statistics and Probability Letters, 33 (1997) 291-297\n',
 'data': array([[   8.3252    ,   41.        ,    6.98412698, ...,    2.55555556,
          37.88      , -122.23      ],
        [   8.3014    ,   21.        ,    6.23813708, ...,    2.10984183,
          37.86      , -122.22      ],
        [   7.2574    ,   52.        ,    8.28813559, ...,    2.80225989,
          37.85      , -122.24      ],
        ...,
        [   1.7       ,   17.        ,    5.20554273, ...,    2.3256351 ,
          39.43      , -121.22      ],
        [   1.8672    ,   18.        ,    5.32951289, ...,    2.12320917,
          39.43      , -121.32      ],
        [   2.3886    ,   16.        ,    5.25471698, ...,    2.61698113,
          39.37      , -121.24      ]]),
 'feature_names': ['MedInc',
  'HouseAge',
  'AveRooms',
  'AveBedrms',
  'Population',
  'AveOccup',
  'Latitude',
  'Longitude'],
 'target': array([4.526, 3.585, 3.521, ..., 0.923, 0.847, 0.894])}
```

In [0]:

```
x, y = housing['data'], housing['target']
print(x.shape)
print(y.shape)
```

```
(20640, 8)
(20640,)
```

In [0]:

```
#so we have 20640 data points and each data point has 8 features and one output value, which is average house value in units of 100,000.
#split it in train and test parts
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state = 0)  #20% of the total data will be test data

#random_state will ensure the same data goes to train and test each time you run the program
```

In [0]:

```
lr = LinearRegression()
lr.fit(x_train, y_train)
```

Out[0]:

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

In [0]:

```
#let's see coefficients and intercept of the linear function, it has fit the trainin data on
print('coefficients:',  lr.coef_)
print('intercept: ', lr.intercept_)
```

```
coefficients: [ 4.33333407e-01  9.29324337e-03 -9.86433739e-02  5.93215487e-01
 -7.56192502e-06 -4.74516383e-03 -4.21449336e-01 -4.34166041e-01]
intercept:  -36.85856910680116
```

In [0]:

```
#let's predict for the test data i.e. unseen data
y_pred = lr.predict(x_test)
```

In [0]:

```
#let's see score of our model
score_test = lr.score(x_test, y_test)
score_train = lr.score(x_train, y_train)
print('Training score: ', score_train)
print('Testing score: ', score_test)
```

```
Training score:  0.6088968118672871
Testing score:  0.5943232652466175
```

In [0]:

```
#Thanks, happy Coding!
```

In [ ]:

```
#To download .ipynb notebook, right click the following url and choose 'save link as'
https://ninjasfiles.s3.amazonaws.com/0000000000003732.ipynb
```