R. M.K

COLLEGE OFENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Date: 14-05-20
Target audience : II & III Year CSE Students

Facilitator: Dr.D.PAULRAJ, Prof. & HoD, CSE

Convenor : Dr.M.Vigilson Prem,Prof. CSE
Co-Ordinators: Dr.K.Akila, ASP CSE
Ms.S.Indra Priyadharshini ASP,CSE
Mr.B.Hariharan, AP, CSE

Topic : Artificial Intelligence, Machine Learning, Linear Regression
(Hands on session)

Summary:

To predict the future outcome by training the machine with past data using **Linear Regression model.** For this hands on session, a classic example of home price prediction data has been used. Machine will predict the price of a building when given its area. Sample data has been taken from Kaggle and demonstrated using Jupyter Notebook, Pandas, MatplotLib, Seaborn, Numpy and Scikit Learn.

Tools Used

Jupyter

The **Jupyter** Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and explanatory text. **Uses** include: data cleaning and transformation, numerical simulation, statistical modeling, machine learning and much more.

Pandas

In computer programming, **pandas** is a software library written for the Python programming language for data manipulation and analysis.

Seaborn

Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

Matplotlib

Matplotlib is a plotting library for the **Python** programming language and its numerical mathematics extension NumPy.

%matplotlib

%**matplotlib** is a magic function in IPython. ... %**matplotlib inline** sets the backend of **matplotlib** to the '**inline**' backend: With this backend, the output of plotting commands is displayed **inline** within frontends like the Jupyter notebook, directly below the code cell that produced it.

Scikit-learn

Scikit-learn is probably the most useful library for machine learning in Python.
The **sklearn** library contains a lot of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction.

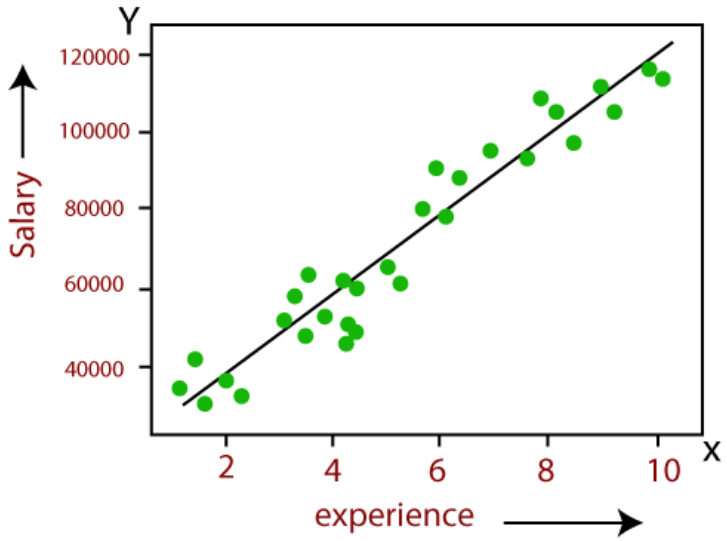
Summary of the technical session.

- **Importance of Artificial Intelligence**

  Artificial Intelligence has become prevalent recently. People across different disciplines are trying to apply AI to make their tasks a lot easier. For example, economists are using AI to predict future market prices to make a profit, doctors use AI to classify whether a tumor is malignant or benign, meteorologists use AI to predict the weather, HR recruiters use AI to check the resume of applicants to verify if the applicant meets the minimum criteria for the job, etcetera.

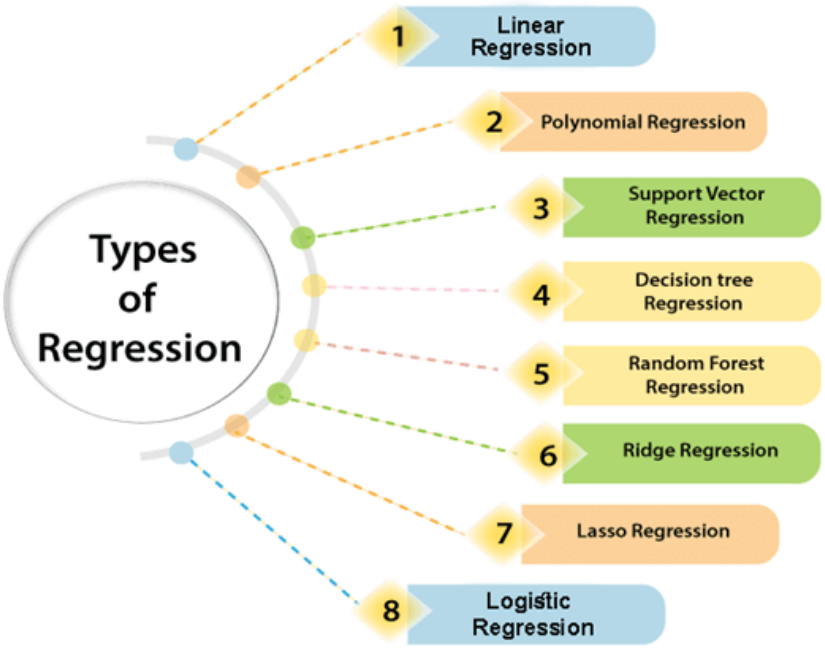
- **Feature of Linear Regression**

  - Linear Regression is one of the most simple Machine learning algorithm that comes under Supervised Learning technique and used for solving regression problems.
  - It is used for predicting the continuous dependent variable with the help of independent variables.
  - The goal of the Linear regression is to find the best fit line that can accurately predict the output for the continuous dependent variable.
  - If single independent variable is used for prediction then it is called Simple Linear Regression and if there are more than two independent variables then such regression is called as Multiple Linear Regression.
  - By finding the best fit line, algorithm establish the relationship between dependent variable and independent variable. And the relationship should be of linear nature.
  - The output for Linear regression should only be the continuous values such as price, age, salary, etc. The relationship between the dependent variable and independent variable can be shown in below image:



- **Types of Regression**

There are various types of regressions which are used in data science and machine learning. Each type has its own importance on different scenarios, but at the core, all the regression methods analyze the effect of the independent variable on dependent variables. Here we are discussing some important types of regression which are given below:



Hands-On Session

Session : 1

Example – 1 (Jupyter Notebook)

In [2]:
```
import pandas as pd
import matplotlib.pylab as plt
import seaborn as sns
%matplotlib inline
```

In [3]:
```
df = pd.read_csv('homeprices.csv')
```

In [4]:
```
df
```

Out[4]:

	area	price
0	2600	550000
1	3000	565000
2	3200	610000
3	3600	680000
4	4000	725000
5	1000	211000
6	1500	316500
7	2300	485300
8	3540	746940
9	4120	869320
10	4560	962160
11	5490	1158390
12	3460	730060
13	4750	1002250
14	2300	485300
15	9000	1899000
16	8600	1814600
17	7100	1498100

In [5]:
```
x=df.drop('price',axis=1)
```

In [6]:
```
x.head()
```

Out[6]:

	area
0	2600
1	3000
2	3200
3	3600
4	4000

In [7]:
```
y=df.drop('area',axis=1)
```

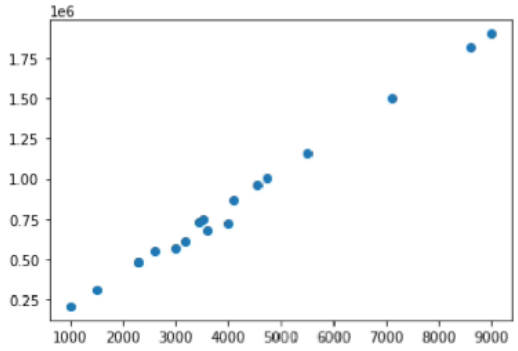
In [8]:
```
y.head()
```

Out[8]:

	price
0	550000
1	565000
2	610000
3	680000
4	725000

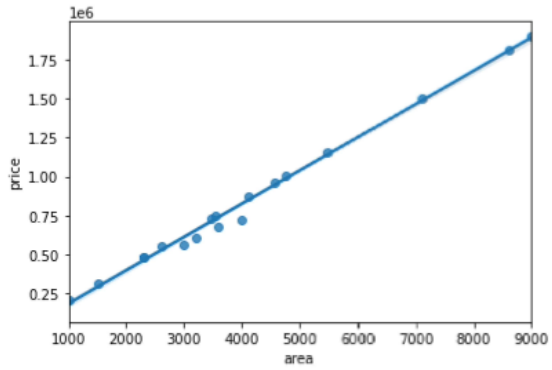
In [9]:
```
plt.scatter(x,y)
```

Out[9]:
```
<matplotlib.collections.PathCollection at 0x21e18837940>
```



In [10]:
```
sns.regplot(x,y)
```

Out[10]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x21e188a6400>
```



In [11]:
```
from sklearn.linear_model import LinearRegression
```

In [12]:
```
model = LinearRegression()
```

In [13]:
```
model.fit(x,y)
```

Out[13]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=Fals
e)
```

In [14]:
```
model.predict([[3540]])
```

Out[14]:
```
array([[727283.58712964]])
```

In [15]:
```
model.score(x,y)
```

Out[15]:
```
0.9940397303173175
```

In [16]:
```
#y=mx+c
m=model.coef_
```

In [17]:
```
m
```

Out[17]:
```
array([[213.25148381]])
```

In [18]:
```
c=model.intercept_
c
```

Out[18]:
```
array([-27626.6655722])
```

```
In [19]: y_predict = m*3540+c
         y_predict
Out[19]: array([[727283.58712964]])
```

Example 2: (Jupyter Notebook)

```
In [3]: import pandas as pd
        import matplotlib.pylab as plt
        import seaborn as sns
        %matplotlib inline
```

```
In [17]: df = pd.read_csv('canada.csv')
         df.head()
```

Out[17]:

	year	per capita income (US$)
0	1970	3399.299037
1	1971	3768.297935
2	1972	4251.175484
3	1973	4804.463248
4	1974	5576.514583

```
In [5]: x=df.drop('per capita income (US$)',axis=1)
        x.head()
```

Out[5]:

	year
0	1970
1	1971
2	1972
3	1973
4	1974

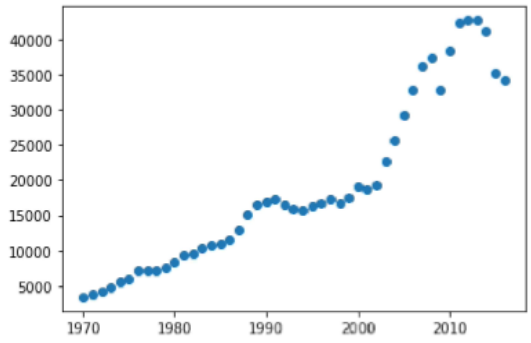
```
In [6]: y=df.drop('year',axis=1)
        y.head()
```

Out[6]:

	per capita income (US$)
0	3399.299037
1	3768.297935
2	4251.175484
3	4804.463248
4	5576.514583

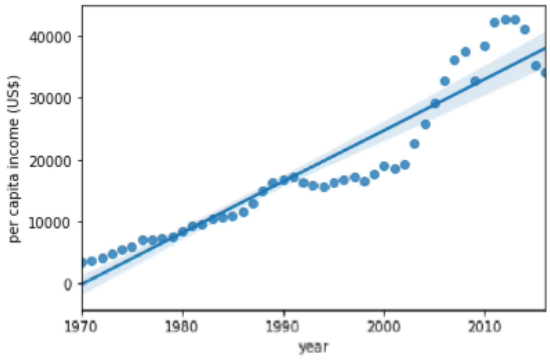
In [7]:
```
plt.scatter(x,y)
```

Out[7]:
```
<matplotlib.collections.PathCollection at 0x18c8d04a520>
```



In [8]:
```
sns.regplot(x,y)
```

Out[8]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x18c8afde250>
```



In [9]:
```
from sklearn.linear_model import LinearRegression
```

In [10]:
```
model = LinearRegression()
```

In [11]:
```
model.fit(x,y)
```

Out[11]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=Fals
e)
```

In [12]:
```
model.predict([[2019]])
```

Out[12]:
```
array([[40460.22901919]])
```

In [13]:
```
model.score(x,y)
```
Out[13]:
```
0.890916917957032
```
In [14]:
```
m=model.coef_
m
```
Out[14]:
```
array([[828.46507522]])
```
In [15]:
```
c=model.intercept_
c
```
Out[15]:
```
array([-1632210.75785546])
```
In [16]:
```
y_pred = m*2019+c
y_pred
```
Out[16]:
```
array([[40460.22901919]])
```
In [ ]:

******** Thanks **********