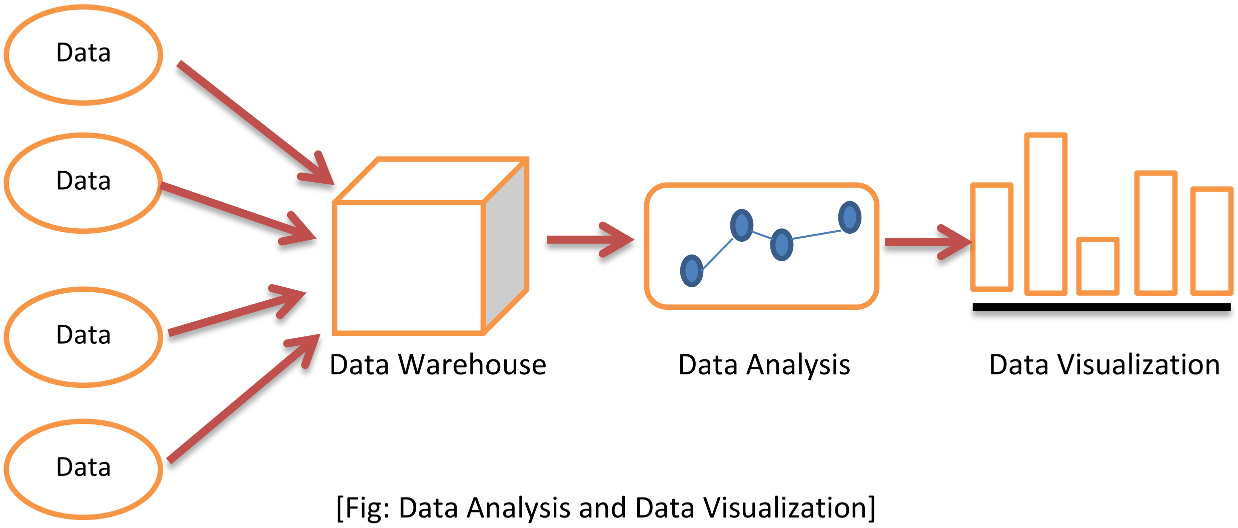
5. Data Science & Machine Learning in Python

Understand the problem By Understanding the Data

Data plays an important role in our lives. For example, chain of hospitals contains data related to medical reports and prescriptions of their patients. A bank contains thousands of customer's transactions details. Share market data represents minutes-to-minute changes in the values of the shares. In this way, the entire world is roaming around huge data.

Every piece of data is precious as it may affect the business organization which is using that data. So, we need some mechanism to store that data. Moreover, data may come from various sources. For example, in a business organization, we may get data from sales department, purchase department, production department, etc.

Once the data is stored, we should be able to retrieve it based on some pre-requisites. A business company wants to know about how much amount they had spent in the last 6 months on purchasing the raw material or how many items had been found defective in their production unit. We have to retrieve the data as per the needs of the business organization. This is called data analysis or data analytics. A person who does data analysis is called 'data analyst'.



[Fig: Data Analysis and Data Visualization]

Data Frame:

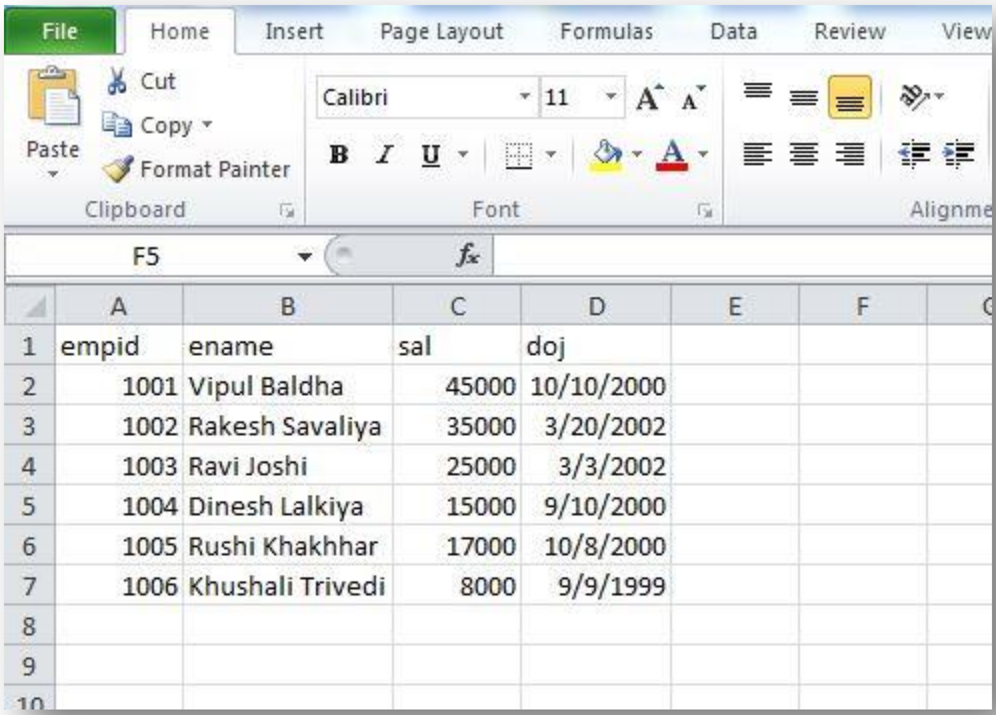
Data Frame is an object that is useful in representing data in the form of rows and columns. For example, the data may come from a file or an excel spreadsheet or from a Python sequence like a list or tuple. We can represent that data in the form of a data frame. Once the data is stored into the data frame, we can perform various operations that are useful in analyzing and understanding the data.

Data frame generally created from .csv (comma separated values) files, Excel Files, Python Dictionaries, list or tuples or list of dictionaries.

Python contains ***pandas*** which is a package useful for data analysis and manipulation. Also ***xlrd*** is a package that is useful to retrieve data from Excel files.

cmd->pip install pandas
cmd->pip install xlrd

Creating Data Frame From an EXCEL Spreadsheet: (create empdata.xlsx)

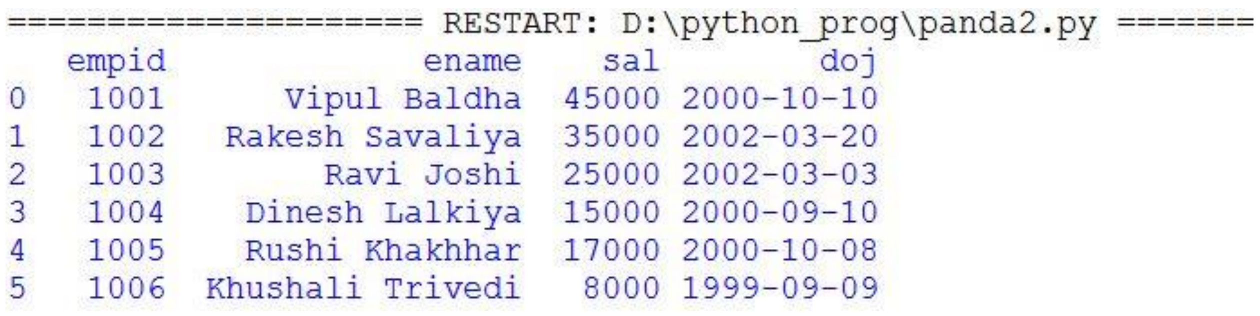


	A	B	C	D
1	empid	ename	sal	doj
2	1001	Vipul Baldha	45000	10/10/2000
3	1002	Rakesh Savaliya	35000	3/20/2002
4	1003	Ravi Joshi	25000	3/3/2002
5	1004	Dinesh Lalkiya	15000	9/10/2000
6	1005	Rushi Khakhhar	17000	10/8/2000
7	1006	Khushali Trivedi	8000	9/9/1999

Prog: To get Excel data to Python (using read_excel())

```
            import pandas as pd
            df=pd.read_excel("D:\python_prog\empdata.xlsx","Sheet1")
            print(df)
```

output:



```
==================== RESTART: D:\python_prog\panda2.py =======
   empid              ename    sal        doj
0   1001       Vipul Baldha  45000 2000-10-10
1   1002    Rakesh Savaliya  35000 2002-03-20
2   1003         Ravi Joshi  25000 2002-03-03
3   1004     Dinesh Lalkiya  15000 2000-09-10
4   1005     Rushi Khakhhar  17000 2000-10-08
5   1006  Khushali Trivedi   8000 1999-09-09
```

Prog: Creating Data Frame from .csv files

```
import pandas as pd
df=pd.read_csv("D:\python_prog\empdata.csv")
print(df)
```

Prog: Creating Data Frame from a python Dictionary

```
            import pandas as pd
            empdata={"empid":[1001,1002,1003],"ename":["vipul baldha","rakesh
Savaliya","Ravi Joshi"],"sal":[45000,35000,25000]}
            df=pd.DataFrame(empdata)
            print(df)
```

Prog: Create Data Frame with different operations

```
import pandas as pd
df=pd.read_csv("D:\python_prog\empdata.csv")
print(df)

r,c=df.shape #it return a tuple that contains number of rows and columns
```

```
print(r) #row
print(c) #column
print(df.head(2)) #df.head() returns first 5 rows
print(df.tail(2))    #df.tail() returns last 5 rows

print(df[2:5])  #Retrieving a Range of Rows
print(df[::2])   #to display alternate rows

print(df.columns)  #To retrieving column names
print(df[['empid','sal']]) #retrieving column data

print("maximum salary is = ",df['sal'].max()) #finding maximum values from column
print("Minimum salary is = ",df['sal'].min()) #finding minimum values

print(df.describe())  #displaying statistical information

print(df[df.sal>10000]) #performs queries on data


print(df[['empid']][df.sal>35000]) #performs queries

print(df.index) #knowing the index range

df1=df.set_index('empid') #setting a column as index

print(df1)

print(df1.loc[1004]) #locate the data from index

df1=df.sort_values('sal',ascending=False) #sorting the data

print(df1)

df1=df.fillna(0) #handling missing data

print(df1)

df1=df.fillna({'ename':'*****Name Missing*****','sal':0.0,'doj':'00-00-00'})
```

```
print(df1)

df1=df.dropna() #remove the missing data

print(df1)
```

Machine Learning in Python

- **Predictive Model Building:**

  1. **Balancing Performance,**
  2. **Complexity, and**
  3. **Big Data**

The goal of selecting and fitting a predictive algorithm is to achieve the best possible performance. Achieving performance goals involves three factors: complexity of the problem, complexity of the algorithmic model employed, and the amount and richness of the data available. In an e-commerce application, for example, good performance might mean returning correct search results or presenting ads that site visitors frequently click. In a genetic problem, it might mean isolating a few genes responsible for a heritable condition.

The goal of selecting and fitting a predictive algorithm is to achieve the best possible performance. Achieving performance goals involves three factors: complexity of the problem, complexity of the algorithmic model employed, and the amount and richness of the data available. The chapter includes some visual examples that demonstrate the relationship between problem and model complexity and then provides technical guidelines for use in design and development.