Laboratory Manual

For

Data Analytics Using Python

B.Tech (IT)

SEM VI

Dec. 2023

Faculty of Technology
Dharmsinh Desai
University Nadiad
www.ddu.ac.in

[1]

Table of Contents

[1]

Experiment 1

Aim: Introduction to python programming for data analytics

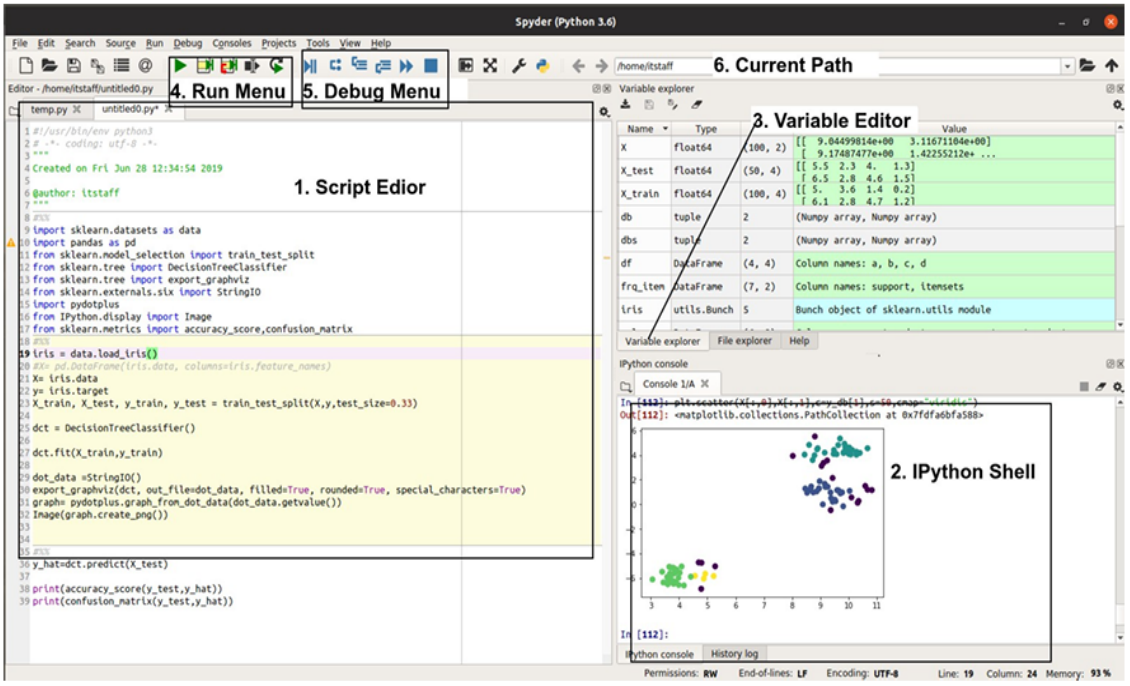
Tools/Apparatus: Anaconda Python/ Spyder IDE

Introduction to Python Programming for Data Mining

Spyder IDE

The Spyder IDE is a freely available Integrated Development Environment for scientific computing using python. It provides an easy to use interfacing for writing, executing, debugging and profiling python scripts as well as IPython command shell all at the same place. It also resembles the look and feel of other scientific computing IDE's like Matlab and RStudio hence the user will find it easy to migrate.

The following Figure shows various parts of the Spyder IDE, and they are explained subsequently



1. Editor Window:

It allows the user to write/edit python programs (referred to as scripts). It also supports intellisence as well as automatic syntax error detection and highlighting.

2. IPython Shell:

It allows the user to run python commands as well as it displays the output of python scripts. The graphs and charts will also be visible in this area

3. Workspace / Variable Window:

Workspace allows you to see all the objects such as datasets and variables active in the memory. Clicking on the name of a dataset in your workspace will bring up a spreadsheet of the data

4. Run Menu:

Allows the users to run the current python script

5. Debug Menu:

Allows the user to debug program by inserting break points, single stepping, step through, step into, step over etc.

6. Current Path:

Allows the user to view/change current working directory

Python is an Interpreted Language

Every python program (script) is a collection of statements separated by a new line and when executed python interpreter will read the script line by line and execute each line print the output and go to the next line until an error occurs or all the statements are not executed.

IPython Shell:

IPython shell is an interface to python interpreter where you can directly executing lines as commands.

```
In [1]: a =10 #do not generate any output
In [2]: b =20 #do not generate any output
In [3]: a + b #generate output is printed on next line
Out[3]: 30
```

**In[x] referes to the input statement x. Out[x] refers to the output of statment x

Variables

```
a=10
b=20.2
name="ABC"
flag=True
```

**Python will automatically determine the type of variable based on value assigned

Mathematical and Logical Operators

+,-,*,%, / (floating point division), // (integer division), , **(power operator)
<, <=, >, >=
and, or, not

Composite DataType:

```
Tuple: t=(10,2.0,True,"abc") #immutable
List: l=[10,2.0,True,"abc"] #mutable
```

**Zero based array like indexing, also possible to give negative idex

```
Dictionary: d ={'code':123, 'name':'abc'} #mutable
```

**Key-value pair,

Conditions:

```
x=10
if x<=15 and x>=5: #block statement stars with ":"
     print("small") #all the statement within block are indented
elif x>=15 and x<20:
    print("medium")
else:
    print("large")
```

Loops:

1. While Loop

```
i=0
while i<10:
     print(i)
     i=i+1
print("End")
```

2. For Loop

```
lst=["A","B","C","D"]
for ele in lst:
     print(ele)
print("End")
```

Functions:

```
def fibonacci(n):#function block begins
     a=0          #statement inside function block
     b=1
     print(a)
     while b<=n: #while block with in function block
          print(b)#statemet inside while block a,b
          = b,a+b
n=int(input("Enter Number:"))#statement outside function block
fib(n)
```

Package Mangement using pip (python install package):

pip is a command line based package manager for python. It allows the user to search, download and install new packages as well as list, update, and remove existing packages in the current python environment. Following are the list of useful commands which can be executed from the terminal/command prompt.

```
pip –help (or -h)
pip install package_name==version_number
pip install -r requirement.txt
pip uninstall package_name==version_number
pip show package_name
pip list
pip freeze > requirement.txt
```

Handling multi-dimensional data and element-wise operators using Numpy

Numpy is a numerical computing library in python designed to handle multi-dimensional data in the form of vector and to perform mathematical computation directly on the vectors (without operating at the element level) similar to Matlab.

1. Install Numpy if not already exist

```
pip install numpy
```

2. Import Numpy in the source code

```
import numpy as np
```

****np** is an alias it can be any name but np is most accepted acronym for numpy

Creating a Data vector

```
a= np.array([10,20,30,40])
b=np.array([[1,2,3],[4,5,6],[7,8,9]])
c=np.random.uniform(size=4).reshape(2,2)
```

3. Accessing Elements of the vector

```
a[0], a[2:], a[:3], a[-1],a[-3:],a[1:2]
b[0,2], b[0,:], b[1,:], b[1:2,0:1]
```

4. Element wise operators

```
A=np.array([[1,2,3],[4,5,6],[7,8,9]])
A+10 #scalar to vector operations
Out:[[10,20,30],[40,50,60],[70,80,90]]
B=np.array([[1,1,1],[2,2,2],[3,3,3]])
A+B #element wise addition
A*B #element wise multiplication
C=np.array([1,1,1])
A+C #adding vector to matrix through broadcasting
```

5. Matrix operators

```
np.dot(A,B) #dot product of two matrix
np.transpose(A) #transpose of a matrix
np.cross(A,B) #cross product of two matrix
```

6. Mathematical Operators

```
np.sin(A) #applying sine functions to each element of A
```

**The other widely used functions are np.log, np.abs, np.exp etc

Experiment 2

Aim: To perform creating and accessing DataVector, MathsOperations, DataFramecreation, reading & writing, Handling DataFrame using Pandas and Numpy.

Tools/Apparatus: Anaconda Python/ Spyder IDE

Theory:

pandas is the most widely used library for managing data in python. It is built over the top of numpy library (i.e. pandas object support all operations of numpy). It is also the default input type used in many other data visualization and machine learning libraries like matplotlib, plotply, sklearn, etc.

Two types of data objects are supported in pandas namely DataSeries and DataFrame. DataSeries is a 1D data vector with name whereas DataFrame is a tabular representation of data with names rows and columns

Procedure:

1. Import pandas

```
import pandas as pd
```

****pd** is an alias can be anything but it is most widely used standard

2. Create DataSeries

```
ds = pd.Series([10,20,30])
```

**elements in the DataSeries are automatically index using integer starting from "0"

```
ds = pd.Series([10,20,30],index=['A','B','C']) #custom index
ds.kesy() #gives you index
ds.values() #gives you values
ds.items() #gives key- value pair as a tuple
ds[0] #or ds['A'] return first element
ds[0:2] #get element staring from 0 but not including 2
ds[ds>10] #returns all element >10
```

3. Create DataFrame

```
df = pd.DataFrame([[1,2,3],[4,5,6]],columns=['A','B'], dtype=int)
df.A #access column A
df.B #access column B
disc = {'name':['A','B','C'], 'age':[20,30,40]}
df = pd.DataFrame(disc)#DataFrame from dictionary
df.name # or df['name']
df['weight']=[60,69,70] #add new column
df.name[0]='X' #change a single value
df.iloc[:2, 0:1] #get the first two rows of column 0 and 1
df.loc[:2,'A']
```

4. Reading Data from File

`df=pd.read_csv("data.csv")` **other options are read_excel, read_json, read_html, read_table, read_clipboard, read_sql_query, read_sql_table, read_hdf and many more

5. Reading the inbuilt datasets

```
import sklearn.datasets as data
df = data.load_iris()
```

**the other options are load_diabetes, load_boston, load_digits, load_breast_cancer, etc.

6. Performing Mathematical Operations

```
np.sum(ds) #sum all elements of DataSeries
np.mean(df) #perform column wise mean
np.sin(df) #find sin of each element
ds1= pd.Series({'A':10,'B':20,'C':30}, dtype=int)
ds2 = pd.Series({'X':20,'Y':50,'Z':60}, dtype=int)
ds= ds1+ds2 #element wise operation
```

7. Handling Missing Values

```
df.isnull() #finding missing values
df.dropna() #Drop all rows with missing values
ds = pd.Series([3,np.nan,4,np,nan])
ds.fillna(method='bfill') #fill the missing values using next values
[3,4,4,np.nan]
ds.fillna(method='ffill')
[3,3,4,4]
```

8. Data Normalization

8.1. Min-Max normalization

Implement your logic to normalize dataframe using following equation,

$$v' = \frac{v - min_A}{max_A - min_A}(new\_max_A - new\_min_A) + new\_min_A$$

8.2. Z-score normalization

Implement your logic to normalize dataframe using following equation,

$$z = \frac{x - \mu}{\sigma}$$

$\mu =$ Mean

$\sigma =$ Standard Deviation

Experiment 3

Aim: To perform DataPre-processing and Wrangling tasks like data cleaning, transformation, join, re-shape, string manipulation sample data-set

Tools/Apparatus: Anaconda Python/ Spyder IDE

Procedure:

Data Wrangling is a crucial topic for Data Science and Data Analysis. Pandas Framework of Python is used for Data Wrangling. Pandas is an open-source library in Python specifically developed for Data Analysis and Data Science. It is used for processes like data sorting or filtration, Data grouping, etc.

Data wrangling in Python deals with the below functionalities:

1. **Data exploration:** In this process, the data is studied, analyzed, and understood by visualizing representations of data.
2. **Dealing with missing values:** Most of the datasets having a vast amount of data contain missing values of *NaN, they are needed to be taken* care of by replacing them with mean, mode, the most frequent value of the column, or simply by dropping the row having a *NaN* value.
3. **Reshaping data:** In this process, data is manipulated according to the requirements, where new data can be added or pre-existing data can be modified.
4. **Filtering data:** Some times datasets are comprised of unwanted rows or columns which are required to be removed or filtered

Experiment 4

Aim: To perform Data Visualization and plotting techniques like Lineplot, Barchart, Piechart, Boxplot using Matplotlib libraries.

Tools/Apparatus: Anaconda Python/ Spyder IDE

Procedure:

Data Visualization

Import Chart Library
import matplotlib.pyplot as plt

Line Chart

```
plt.plot([10,20,30,40,50])
```

```
plt.plot([1,2,4,8,16,32,64], 'ro-')
```

**Last one is formating string, first character is color 'r'-red, second is marker 'o'-solid circle, third is line type '-'-dashed line. Other options are as follows

Color: 'r'-red,'g'-green,'b'-blue,'c'-cyan,'m'-magenta,'y'-yellow,'k'-black

Marker: 'o'-solid dot, '.'- small dot, '^'-triangle, 's'-square, '+'-cross, '*'-star

Line: noline, '-' solid, '--' dash, '-.' dash-dot, ':' dots

**Change Graph Display settings
Tools->Preferences->IPython Console->Graphics->Beckend set to 'Automatic' (instead of inline)

```
plt.title("Exponetial Graph")
plt.xlabel("Power of 2")
plt.ylabel("Values")
plt.plot([1,2,4,8,16,32,64],
'ro-')
plt.plot([1,2,3,4,5,6,7], 'g^--')
```

Scatter Plot

```
import sklearn.datasets
as data iris =
data.load_irirs()
plt.scatter(x=iris.data[:,0], y=iris.data[:,1])
```

Bar Chart/Histogram

```
plt.hist(iris.data[:,0], 10)
```

Pie Chart

```
plt.pie(sizes=[10,30,60,90],labels=['A','B','C','D'],autopct='%
1.1f%%')
plt.legend()
```

Box Plot

```
plt.boxplot(iris.data[:,0])
```

Subplots

```
plt.figure(1)
plt.subplot(2
,1,1)
plt.plot([1,2,4,8,16,32,64], 'ro-')
plt.subplot(2,1,2)
plt.plot([1,2,3,4,5,6,7], 'g^--')
```

Plotting in Pandas

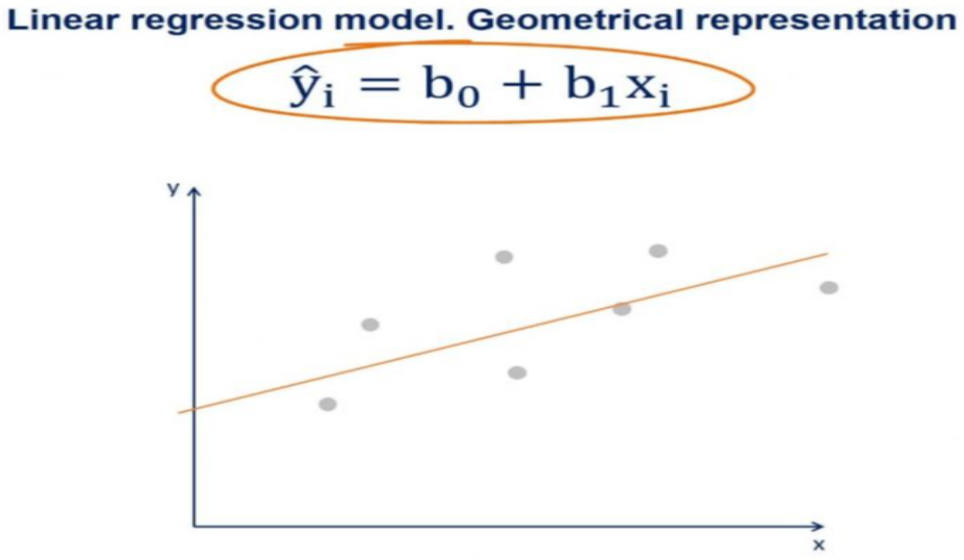
```
import pandas as pd
import sklearn.datasets as data

df = data.load_iris()

df = pd.DataFrame(data=df.data, columns=df.feature_names)
df.plot() #default plot type is line chart for each feature
df.plot(kind='box') #plot the boxplot of each feature
df.plot(kind='hist') #plot the histogram of each feature
```

Experiment 5

Aim: To perform Regression Analysis using Sci-kit learn package in Python.

Tools/Apparatus: Anaconda Python/ Spyder IDE

Theory:



Procedure:

Using numpy and mathplotlib library implement linear regression for any sample data like salary and qualification.

Experiment 6

Aim: To perform Decision Tree Classification (DCT) using sklearn package in python

Tools/Apparatus: Anaconda Python/ Spyder IDE

Theory:

A decision tree is a flow chart like tree structure where each internal node(nonleaf) denotes a test on the attribute, each branch represents an outcome of the test ,and each leaf node(terminal node)holds a class label. Decision trees can be easily converted into classification rules.

Procedure:

1. Import necessary packages

```
import pandas as pd
import numpy as np
import sklearn.dataset as data
from sklearn.model_selection import train_test_split
```

2. Read or Generate the dataset for machine learning task

```
iris = data.load_iris()
```

3. Arrange data into a features matrix (X) and target vector (y)

```
X= iris.data
y= iris.target
```

4. Split data set into training and testing

```
X_train,X_test, y_train, y_test =
train_test_split(X,y,test_size=0.33,)
```

5. Choose a class of model by importing the appropriate estimator class from Scikit- Learn.

```
from sklearn.tree import DecisionTreeClassifier
dct = DecisionTreeClassifier()
```

6. Choose model hyperparameters by instantiating this class with desired values.

   Current keep all default. You can change parameter by changing the values of the arguments to DecisionTreeClassifier constructor

7. Fit the model to your training data by calling the **fit()** method of the model instance.

```
dct.fit(X_train,y_train)
```

8. Test the classifier on Test data

```
y_hat = dct.predict(X_test) # y_hat is predicted output
```

9. Compare predicted output with actual output to find accuracy

```
from sklearn.metrics import accuracy_score, confusion_matrix
print(accuracy_score(y_test,y_hat))
print(confusion_matrix(y_test,y_hat))
```

10. Visualizing Decision Tree

```
import pydotplus
from Ipython.display import Image
from sklearn.tree import export_graphviz
from sklearn.externals.six import StringIO
dot_data =StringIO()
export_graphviz(dct, out_file=dot_data, filled=True,
rounded=True, special_characters=True)
graph= pydotplus.graph_from_dot_data(dot_data.getvalue())
Image(graph.create_png())
```

Experiment 7

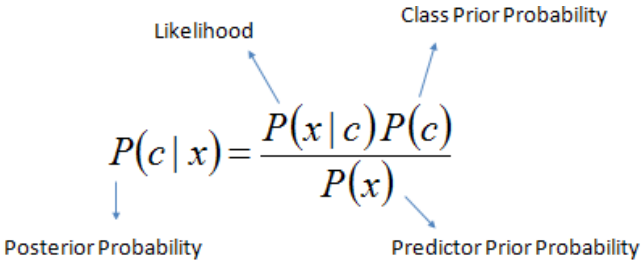
Aim: To perform Navie Bayes and K-NN (k-nearest neighbor) classification using sklearn package in python

Tools/Apparatus: Anaconda Python/ Spyder IDE

Theory:

The Naive Bayesian classifier is based on Bayes' theorem with independence assumptions between predictors. A Naive Bayesian model is easy to build, with no complicated iterative parameter estimation which makes it particularly useful for very large datasets. Despite its simplicity, the Naive Bayesian classifier often does surprisingly well and is widely used because it often outperforms more sophisticated classification methods.

Bayes theorem provides a way of calculating the posterior probability, $P(c|x)$, from $P(c)$, $P(x)$, and $P(x|c)$. Naive Bayes classifier assume that the effect of the value of a predictor (x) on a given class (c) is independent of the values of other predictors. This assumption is called class conditional independence.



$$P(c \mid \mathrm{X}) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

- $P(c|x)$ is the posterior probability of *class* (*target*) given *predictor* (*attribute*).
- $P(c)$ is the prior probability of *class*.
- $P(x|c)$ is the likelihood which is the probability of *predictor* given *class*.
- $P(x)$ is the prior probability of *predictor*.

k Nearest Neighbours (kNN) is a simple algorithm that stores all the available training samples as template database and classifies new unknown sample by finding the majority class label of the k neighbours of unknown sample.

Procedure:

Same as Experiment 8, only change is to **replace the model object in step 5** with appropriate model

```
from sklearn.naive_bayes import GaussianNB

nb = GaussianNB()

from sklearn.neighbors import KneighborsClassifier

knn = KneighborsClassifier(n_neighbors=5)
```

Experiment 8

Aim: To perform KMeans and DBSCAN clustering using sklearn package in python.

Tools/Apparatus: Anaconda Python/ Spyder IDE

Theory:

K-Means clustering intends to partition *n* objects into *k* clusters in which each object belongs to the cluster with the nearest mean. This method produces exactly *k* different clusters of greatest possible distinction. The best number of clusters *k* leading to the greatest separation (distance) is not known as a priori and must be computed from the data. The objective of K-Means clustering is to minimize total intra-cluster variance, or, the squared error function:

Procedure:

1. Generate Random Data for clustering

```
from sklearn.datasets.samples_generator import make_blobs
import matplotlib.pyplot as plt
X, y_true = make_blobs(n_samples=100, centers=3, cluster_std=0.6)
plt.scatter(X[:,0],X[:,1], s=50)
```

2. Create KMeans Clustering Model

```
from sklearn.cluster import KMeans
km = Kmeans(n_clusters=3)
```

3. Run Clustering Algorithm to fit the data and find cluster labels

```
km.fit(X)
y_lab = km.predict(X)
```

4. Plot the data points using scatter-plot

```
plt.scatter(X[:, 0], X[:, 1], c=y_lab, s=50, cmap='viridis')
centers = kmeans.cluster_centers_
plt.scatter(centers[:, 0], centers[:, 1], c='black', s=200,
alpha=0.5);
```

For DBSCAN channge step 2 and step 4 as following

2. Create DBSCAN model

```
from sklearn.cluster import dbscan
dbs = dbscan(X)
```

3. Plot Data Points

```
plt.scatter(X[:, 0], X[:, 1], c=dbs[1], s=50, cmap='viridis')
```

Experiment 9

Aim: To perform text mining using textblob in python (TF-IDF generation, sentiment analysis, word-cloud, POS tagging)

Tools/Apparatus: Anaconda Python/ Spyder IDE

Theory: TextBlob creates a blob object from input Text. Blob object is made of Sentence Objects, each Sentence Object has WordList which is collection of Word objects present in the Sentence. TextBlob provides various operations as Blob level, sentence level and word level. For example, POS tagging is done at blob level. Sentiment analysis is done at sentence level. Inflection and Lemmatization done at word level.

Procedure:

1. Install and Import TextBlob

```
terminal>>pip intsall textblob

from textblob import TextBlob
```

2. Create a TextBlob Object from Text

```
blob = TextBlob(``` This could be a multiline text or paragraph copy
from wiki

               or something else what ever you like.

               Good Bye```)

blob.correct() #will correct the spelling mistakes if any.
```

**We also have spellcheck() method in word object to check the spelling of single word but it will not correct it automatically

3. Part of Speech Tagging (POS Tags)

```
blob.tags # will return list of tuples containing word and
its tag, noun, verbe, etc.

blob.noun_phrase # will return words that are used as nouns
```

4. Tokenizations (Getting sentences and words)

```
blob.sentences

blob.words

blob.word_counts
```

5. Sentiment Analysis of sentences

```
for sentence in blob.sentences:

   print(sentence.sentiment)
```

6. Word Inflection and Lemmatizations

```
sentence[0].words[0].singularize()
sentence[0].words[-1].pluralize()
w = Word("octopi")w.lemmatize()
w = Word("done")
w.lemmantize("v")
Word("Beautiful").definitions
```

7. Integrating WordNet and Finding Synonyms

```
from textblob.wordnet import VERB,ADJ,ADV,NOUN
w = Word("Smart")
w.get_synset(pos=ADJ)
```

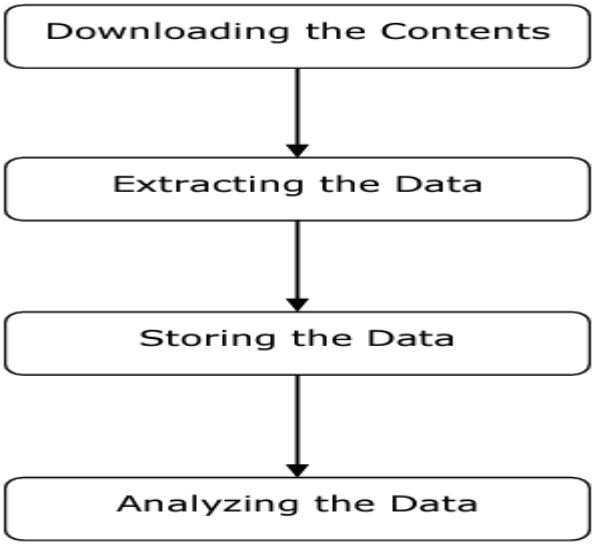
Experiment 10

Aim: To perform basic Web-Scraping tasks using the Beautiful soup package in Python.

Tools/Apparatus: Anaconda Python/ Spyder IDE

Procedure: Use Request, Beautiful-soap libraries

Web scraper may be defined as a software or script used to download the contents of multiple web pages and extracting data from it.



Step 1:

Downloading Contents from Web Pages In this step, a web scraper will download the requested contents from multiple web pages.

Step 2:

Extracting Data The data on websites is HTML and mostly unstructured. Hence, in this step, web scraper will parse and extract structured data from the downloaded contents.

Step 3:

Storing the Data Here, a web scraper will store and save the extracted data in any of the format like CSV, JSON or in database.

Step 4:

Analyzing the Data After all these steps are successfully done, the web scraper will analyze the data thus obtained.

References

[1] **Scikit-learn: Machine Learning in Python, Pedregosa *et al.*, JMLR 12, pp. 2825-2830, 2011.**

Books

[1] Data Mining: Concepts and Techniques, Third Edition

Jiawei Han, Micheline Kamber, Jian Pei, Morgan Kaufmann , 2011.

[2] Data mining,

Pieter Adriaans & Dolf Zantinge, Pearson Education Asia(2001).

Web

[1] https://scikit-learn.org/stable/tutorial/index.html

[2] https://diveintopython.org

[3] https://jakevdp.github.io/PythonDataScienceHandbook/

[4] http://www.datacamp.com/

[5] http://www.dataquest.io