BANSILAL RAMNATH AGARWAL CHARITABLE TRUST'S

VIIT, PUNE

Lab Manual

Data Science and Machine Learning

For

TY - B. Tech Pattern 2020

Name: Pratik Kanhaiya Rathod

Roll No : 321056

Gr NO : 22010885

Div : A Batch : A_2

Issue Date:

Copy No: 01

Copy Holder: Practical IC

Assignment No.: 1

Problem Statement:

Perform the following operations using R/Python on suitable data sets, read data from different formats (like csv, xls), indexing and selecting data, sort data, describe attributes of data, checking data types of each column, counting unique values of data, format of each column, converting variable data type (e.g., from long to short, vice versa), identifying missing values and fill in the missing values.

Objective:

To perform operations such as read data sets from different formats (like csv,xls), indexing and selecting data, sort data, describe attributes of data, checking data types of each column, counting unique values of data, format of each column. converting variable data type (e.g., from long to short, vice versa), identifying missing values and fill in the missing values using R/Python language.

Theory:

It is a process of inspecting, cleansing, transforming, and modeling data so that we can derive some useful information from the data and use it for future predictions. Data analysis tools make it easier for users to process and manipulate data, analyze the relationships and correlations between data sets, and it also helps to identify patterns and trends for interpretation. Here, we will be using python to study data analysis in-depth and study all the important aspects of data analysis.

Libraries Used:

1) **Pandas:** Pandas is a Python library for data analysis. Started by Wes McKinney in 2008 out of a need for a powerful and flexible quantitative analysis tool, pandas has grown into one of the most popular Python libraries. It has an extremely active community of contributors**.**

2) **Numpy:** NumPy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, Fourier transform, and matrices. NumPy was created in 2005 by Travis Oliphant. It is an open-source project and you can use it freely.

Algorithm:

1) Start
2) Read a csv/xls file using R/Python.
3) Import all the libraries essential for the operations.

4) Create an object for easy access of the data set.
5) Use the object for various operations on the dataset.
6) Use the object for various attribute manipulation on the dataset.
7) End

Input:

Sample Input 1: Telecom churn dataset (telecom_churn.csv)
Sample Input 2: Test dataset (test.csv)

Output:

Importing required libraries-

```
import pandas as pd
import numpy as np
from google.colab import drive
drive.mount("/content/drive")
df = pd.read_csv("/content/drive/MyDrive/DSML/telecom_churn.csv")
titanic = pd.read_csv("/content/drive/MyDrive/DSML/test.csv")
```

Reading dataset files-

```
df.head(10)
```

	state	account length	area code	phone number	international plan	voice mail plan	number vmail messages	total day minutes	total day calls	total day charge	...	total eve calls	total eve charge	total night minutes	total night calls	total night charge	total intl minutes	total intl calls	total intl charge	customer service calls	churn
0	KS	128	415	382-4657	no	yes	25	265.1	110	45.07	...	99	16.78	244.7	91	11.01	10.0	3	2.70	1	False
1	OH	107	415	371-7191	no	yes	26	161.6	123	27.47	...	103	16.62	254.4	103	11.45	13.7	3	3.70	1	False
2	NJ	137	415	358-1921	no	no	0	243.4	114	41.38	...	110	10.30	162.6	104	7.32	12.2	5	3.29	0	False
3	OH	84	408	375-9999	yes	no	0	299.4	71	50.90	...	88	5.26	196.9	89	8.86	6.6	7	1.78	2	False
4	OK	75	415	330-6626	yes	no	0	166.7	113	28.34	...	122	12.61	186.9	121	8.41	10.1	3	2.73	3	False
5	AL	118	510	391-8027	yes	no	0	223.4	98	37.98	...	101	18.75	203.9	118	9.18	6.3	6	1.70	0	False
6	MA	121	510	355-9993	no	yes	24	218.2	88	37.09	...	108	29.62	212.6	118	9.57	7.5	7	2.03	3	False
7	MO	147	415	329-9001	yes	no	0	157.0	79	26.69	...	94	8.76	211.8	96	9.53	7.1	6	1.92	0	False
8	LA	117	408	335-4719	no	no	0	184.5	97	31.37	...	80	29.89	215.8	90	9.71	8.7	4	2.35	1	False
9	WV	141	415	330-8173	yes	yes	37	258.6	84	43.96	...	111	18.87	326.4	97	14.69	11.2	5	3.02	0	False

10 rows × 21 columns

```
titanic.tail()
```

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
413	1305	3	Spector, Mr. Woolf	male	NaN	0	0	A.5. 3236	8.0500	NaN	S
414	1306	1	Oliva y Ocana, Dona. Fermina	female	39.0	0	0	PC 17758	108.9000	C105	C
415	1307	3	Saether, Mr. Simon Sivertsen	male	38.5	0	0	SOTON/O.Q. 3101262	7.2500	NaN	S
416	1308	3	Ware, Mr. Frederick	male	NaN	0	0	359309	8.0500	NaN	S
417	1309	3	Peter, Master. Michael J	male	NaN	1	1	2668	22.3583	NaN	C

Selecting and Indexing data-

```
df[4:7]
```

	state	account length	area code	phone number	international plan	voice mail plan	number vmail messages	total day minutes	total day calls	total day charge	...	total eve calls	total eve charge	total night minutes	total night calls	total night charge	total intl minutes	total intl calls	total intl charge	customer service calls	churn
4	OK	75	415	330-6626	yes	no	0	166.7	113	28.34	...	122	12.61	186.9	121	8.41	10.1	3	2.73	3	False
5	AL	118	510	391-8027	yes	no	0	223.4	98	37.98	...	101	18.75	203.9	118	9.18	6.3	6	1.70	0	False
6	MA	121	510	355-9993	no	yes	24	218.2	88	37.09	...	108	29.62	212.6	118	9.57	7.5	7	2.03	3	False

3 rows × 21 columns

```
df['state']
```

```
0       KS
1       OH
2       NJ
3       OH
4       OK
       ..
3328    AZ
3329    WV
3330    RI
3331    CT
3332    TN
Name: state, Length: 3333, dtype: object
```

```
df.loc[2:6]
```

	state	account length	area code	phone number	international plan	voice mail plan	number vmail messages	total day minutes	total day calls	total day charge	...	total eve calls	total eve charge	total night minutes	total night calls	total night charge	total intl minutes	total intl calls	total intl charge	customer service calls	churn
2	NJ	137	415	358-1921	no	no	0	243.4	114	41.38	...	110	10.30	162.6	104	7.32	12.2	5	3.29	0	False
3	OH	84	408	375-9999	yes	no	0	299.4	71	50.90	...	88	5.26	196.9	89	8.86	6.6	7	1.78	2	False
4	OK	75	415	330-6626	yes	no	0	166.7	113	28.34	...	122	12.61	186.9	121	8.41	10.1	3	2.73	3	False
5	AL	118	510	391-8027	yes	no	0	223.4	98	37.98	...	101	18.75	203.9	118	9.18	6.3	6	1.70	0	False
6	MA	121	510	355-9993	no	yes	24	218.2	88	37.09	...	108	29.62	212.6	118	9.57	7.5	7	2.03	3	False

5 rows × 21 columns

```
df.loc[2]
```

```
state                     NJ
account length           137
area code                415
phone number        358-1921
international plan        no
voice mail plan           no
number vmail messages      0
total day minutes      243.4
total day calls          114
total day charge       41.38
total eve minutes      121.2
total eve calls          110
total eve charge        10.3
total night minutes    162.6
total night calls        104
total night charge      7.32
total intl minutes      12.2
total intl calls           5
total intl charge       3.29
customer service calls     0
churn                  False
Name: 2, dtype: object
```

```
df.loc[2:6, ["state", "account length"]]
```

	state	account length
2	NJ	137
3	OH	84
4	OK	75
5	AL	118
6	MA	121

```
df.loc[44:55, ["state", "account length", "churn"]]
```

	state	account length	churn
44	WI	64	False
45	OR	59	False
46	MI	65	False
47	DE	142	False
48	ID	119	True
49	WY	97	False
50	IA	52	False
51	IN	60	False
52	VA	10	False
53	UT	96	False
54	WY	87	True
55	IN	81	False

Describing data attributes-

```
df.describe()
```

	account length	area code	number vmail messages	total day minutes	total day calls	total day charge	total eve minutes	total eve calls	total eve charge	total night minutes	total night calls	total night charge	total intl minutes	total intl calls	total intl charge
count	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000
mean	101.064806	437.182418	8.099010	179.775098	100.435644	30.562307	200.980348	100.114311	17.083540	200.872037	100.107711	9.039325	10.237294	4.479448	2.764581
std	39.822106	42.371290	13.688365	54.467389	20.069084	9.259435	50.713844	19.922625	4.310668	50.573847	19.568609	2.275873	2.791840	2.461214	0.753773
min	1.000000	408.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	23.200000	33.000000	1.040000	0.000000	0.000000	0.000000
25%	74.000000	408.000000	0.000000	143.700000	87.000000	24.430000	166.600000	87.000000	14.160000	167.000000	87.000000	7.520000	8.500000	3.000000	2.300000
50%	101.000000	415.000000	0.000000	179.400000	101.000000	30.500000	201.400000	100.000000	17.120000	201.200000	100.000000	9.050000	10.300000	4.000000	2.780000
75%	127.000000	510.000000	20.000000	216.400000	114.000000	36.790000	235.300000	114.000000	20.000000	235.300000	113.000000	10.590000	12.100000	6.000000	3.270000
max	243.000000	510.000000	51.000000	350.800000	165.000000	59.640000	363.700000	170.000000	30.910000	395.000000	175.000000	17.770000	20.000000	20.000000	5.400000

```
df.describe
```

```
<bound method NDFrame.describe of      state  account length  area code phone number international plan  \
0       KS             128        415     382-4657                 no
1       OH             107        415     371-7191                 no
2       NJ             137        415     358-1921                 no
3       OH              84        408     375-9999                yes
4       OK              75        415     330-6626                yes
...    ...             ...        ...          ...                ...
3328    AZ             192        415     414-4276                 no
3329    WV              68        415     370-3271                 no
3330    RI              28        510     328-8230                 no
3331    CT             184        510     364-6381                yes
3332    TN              74        415     400-4344                 no

     voice mail plan  number vmail messages  total day minutes  \
0                yes                     25              265.1
1                yes                     26              161.6
2                 no                      0              243.4
3                 no                      0              299.4
4                 no                      0              166.7
...              ...                    ...                ...
3328             yes                     36              156.2
3329              no                      0              231.1
3330              no                      0              180.8
3331              no                      0              213.8
3332             yes                     25              234.4

      total day calls  total day charge  ...  total eve calls  \
0                 110             45.07  ...               99
1                 123             27.47  ...              103
2                 114             41.38  ...              110
3                  71             50.90  ...               88
4                 113             28.34  ...              122
...               ...               ...  ...              ...
3328               77             26.55  ...              126
3329               57             39.29  ...               55
3330              109             30.74  ...               58
3331              105             36.35  ...               84
3332              113             39.85  ...               82
```

```
      total eve charge  total night minutes  total night calls  \
0                16.78                244.7                 91
1                16.62                254.4                103
2                10.30                162.6                104
3                 5.26                196.9                 89
4                12.61                186.9                121
...                ...                  ...                ...
3328             18.32                279.1                 83
3329             13.04                191.3                123
3330             24.55                191.9                 91
3331             13.57                139.2                137
3332             22.60                241.4                 77

      total night charge  total intl minutes  total intl calls  \
0                  11.01                10.0                 3
1                  11.45                13.7                 3
2                   7.32                12.2                 5
3                   8.86                 6.6                 7
4                   8.41                10.1                 3
...                  ...                 ...               ...
3328               12.56                 9.9                 6
3329                8.61                 9.6                 4
3330                8.64                14.1                 6
3331                6.26                 5.0                10
3332               10.86                13.7                 4

      total intl charge  customer service calls  churn
0                  2.70                       1  False
1                  3.70                       1  False
2                  3.29                       0  False
3                  1.78                       2  False
4                  2.73                       3  False
...                 ...                     ...    ...
3328               2.67                       2  False
3329               2.59                       3  False
3330               3.81                       2  False
3331               1.35                       2  False
3332               3.70                       0  False

[3333 rows x 21 columns]>
```

Sorting data-

```
df.sort_values(["state", "churn"])
```

	state	account length	area code	phone number	international plan	voice mail plan	number vmail messages	total day minutes	total day calls	total day charge	...	total eve calls	total eve charge	total night minutes	total night calls	total night charge	total intl minutes	total intl calls	total intl charge	customer service calls	churn
36	AK	36	408	341-9764	no	yes	30	146.3	128	24.87	...	80	13.81	129.3	109	5.82	14.5	6	3.92	0	False
38	AK	136	415	402-1381	yes	yes	33	203.9	106	34.66	...	99	15.95	101.7	107	4.58	10.5	6	2.84	3	False
95	AK	104	408	366-4467	no	no	0	278.4	106	47.33	...	113	6.89	163.2	137	7.34	9.8	5	2.65	1	False
138	AK	127	510	345-8237	no	yes	36	183.2	117	31.14	...	76	10.78	263.3	71	11.85	11.2	8	3.02	1	False
282	AK	48	415	389-7073	no	yes	37	211.7	115	35.99	...	84	13.59	144.1	80	6.48	12.2	1	3.29	1	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1337	WY	97	510	346-1629	yes	no	0	236.9	107	40.27	...	105	13.40	241.0	120	10.85	7.3	2	1.97	0	True
1533	WY	127	510	400-2181	yes	no	0	242.2	102	41.17	...	80	19.22	252.0	96	11.34	13.9	5	3.75	2	True
2819	WY	159	415	391-2159	no	no	0	167.4	68	28.46	...	74	12.22	140.1	111	6.30	10.3	3	2.78	0	True
2874	WY	134	510	366-1084	no	no	0	296.0	93	50.32	...	117	19.24	246.8	98	11.11	12.3	10	3.32	0	True
2952	WY	123	415	387-1116	no	no	0	242.2	87	41.17	...	101	19.22	268.6	121	12.09	8.2	3	2.21	5	True

3333 rows × 21 columns

```
df.sort_values("area code")
```

	state	account length	area code	phone number	international plan	voice mail plan	number vmail messages	total day minutes	total day calls	total day charge	...	total eve calls	total eve charge	total night minutes	total night calls	total night charge	total intl minutes	total intl calls	total intl charge	customer service calls	churn
2536	CT	119	408	344-5181	no	no	0	294.2	100	50.01	...	53	19.76	195.0	64	8.78	9.0	1	2.43	0	True
887	IA	128	408	335-8146	no	no	0	158.8	75	27.00	...	91	22.51	270.0	77	12.15	7.6	7	2.05	1	False
2486	MS	76	408	368-8972	no	no	0	173.2	93	29.44	...	80	11.15	170.9	104	7.69	5.4	3	1.46	0	False
2482	MT	157	408	417-3257	no	no	0	240.2	67	40.83	...	98	13.01	249.0	72	11.21	10.2	6	2.75	2	False
2476	WV	84	408	369-1220	no	no	0	146.8	133	24.96	...	73	14.59	234.5	69	10.55	9.9	3	2.67	1	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
983	MN	92	510	344-7470	no	no	0	212.4	105	36.11	...	118	19.09	221.3	105	9.96	9.0	4	2.43	1	False
2390	NY	122	510	397-3943	no	no	0	145.6	102	24.75	...	111	24.20	228.2	91	10.27	12.2	5	3.29	0	False
2388	SC	161	510	343-2592	no	no	0	297.9	141	50.64	...	107	20.24	240.5	93	10.82	8.9	5	2.40	1	True
2396	WY	127	510	356-4706	yes	no	0	247.5	99	42.08	...	118	9.22	232.0	72	10.44	10.6	3	2.86	2	False
869	NE	127	510	348-5567	yes	no	0	180.9	114	30.75	...	118	17.81	249.9	105	11.25	7.4	4	2.00	2	False

3333 rows × 21 columns

Checking data types of each column-

```
df.dtypes
```

```
state                   object
account length           int64
area code                int64
phone number            object
international plan      object
voice mail plan         object
number vmail messages    int64
total day minutes       float64
total day calls          int64
total day charge        float64
total eve minutes       float64
total eve calls          int64
total eve charge        float64
total night minutes     float64
total night calls        int64
total night charge      float64
total intl minutes      float64
total intl calls         int64
total intl charge       float64
customer service calls   int64
churn                     bool
dtype: object
```

```
df["area code"].dtypes
```

```
dtype('int64')
```

Counting unique values of data-

```
df.count()
```

```
state                   3333
account length          3333
area code               3333
phone number            3333
international plan      3333
voice mail plan         3333
number vmail messages   3333
total day minutes       3333
total day calls         3333
total day charge        3333
total eve minutes       3333
total eve calls         3333
total eve charge        3333
total night minutes     3333
total night calls       3333
total night charge      3333
total intl minutes      3333
total intl calls        3333
total intl charge       3333
customer service calls  3333
churn                   3333
dtype: int64
```

```
df["area code"].count()
```

```
3333
```

```
df["area code"].count()
```

```
array([415, 408, 510])
```

```
len(df["area code"].unique())
```

```
3
```

Converting variable data type-

```
df["area code"] = df["area code"].astype(str)
df.dtypes
```

```
state                    object
account length            int64
area code                object
phone number             object
international plan       object
voice mail plan          object
number vmail messages     int64
total day minutes       float64
total day calls           int64
total day charge        float64
total eve minutes       float64
total eve calls           int64
total eve charge        float64
total night minutes     float64
total night calls         int64
total night charge      float64
total intl minutes      float64
total intl calls          int64
total intl charge       float64
customer service calls    int64
churn                      bool
dtype: object
```

Checking formats of column-

```
df.columns
```

```
Index(['state', 'account length', 'area code', 'phone number',
       'international plan', 'voice mail plan', 'number vmail messages',
       'total day minutes', 'total day calls', 'total day charge',
       'total eve minutes', 'total eve calls', 'total eve charge',
       'total night minutes', 'total night calls', 'total night charge',
       'total intl minutes', 'total intl calls', 'total intl charge',
       'customer service calls', 'churn'],
      dtype='object')
```

```
df.shape
```

```
(3333, 21)
```

Identifying the missing values-

```
titanic.isnull()
```

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	False	False	False	False	False	False	False	False	False	True	False
1	False	False	False	False	False	False	False	False	False	True	False
2	False	False	False	False	False	False	False	False	False	True	False
3	False	False	False	False	False	False	False	False	False	True	False
4	False	False	False	False	False	False	False	False	False	True	False
...	...	...	...	...	...	...	...	...	...	...	...
413	False	False	False	False	True	False	False	False	False	True	False
414	False	False	False	False	False	False	False	False	False	False	False
415	False	False	False	False	False	False	False	False	False	True	False
416	False	False	False	False	True	False	False	False	False	True	False
417	False	False	False	False	True	False	False	False	False	True	False

418 rows × 11 columns

```
titanic.isna()
```

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	False	False	False	False	False	False	False	False	False	True	False
1	False	False	False	False	False	False	False	False	False	True	False
2	False	False	False	False	False	False	False	False	False	True	False
3	False	False	False	False	False	False	False	False	False	True	False
4	False	False	False	False	False	False	False	False	False	True	False
...	...	...	...	...	...	...	...	...	...	...	...
413	False	False	False	False	True	False	False	False	False	True	False
414	False	False	False	False	False	False	False	False	False	False	False
415	False	False	False	False	False	False	False	False	False	True	False
416	False	False	False	False	True	False	False	False	False	True	False
417	False	False	False	False	True	False	False	False	False	True	False

418 rows × 11 columns

```
titanic.isnull().sum()
```

```
PassengerId      0
Pclass           0
Name             0
Sex              0
Age             86
SibSp            0
Parch            0
Ticket           0
Fare             1
Cabin          327
Embarked         0
dtype: int64
```

Filling in the missing the values-

```
titanic["Age"].fillna(titanic["Age"].mean(), inplace=True)
titanic
```

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	892	3	Kelly, Mr. James	male	34.50000	0	0	330911	7.8292	NaN	Q
1	893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.00000	1	0	363272	7.0000	NaN	S
2	894	2	Myles, Mr. Thomas Francis	male	62.00000	0	0	240276	9.6875	NaN	Q
3	895	3	Wirz, Mr. Albert	male	27.00000	0	0	315154	8.6625	NaN	S
4	896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.00000	1	1	3101298	12.2875	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...
413	1305	3	Spector, Mr. Woolf	male	30.27259	0	0	A.5. 3236	8.0500	NaN	S
414	1306	1	Oliva y Ocana, Dona. Fermina	female	39.00000	0	0	PC 17758	108.9000	C105	C
415	1307	3	Saether, Mr. Simon Sivertsen	male	38.50000	0	0	SOTON/O.Q. 3101262	7.2500	NaN	S
416	1308	3	Ware, Mr. Frederick	male	30.27259	0	0	359309	8.0500	NaN	S
417	1309	3	Peter, Master. Michael J	male	30.27259	1	1	2668	22.3583	NaN	C

418 rows × 11 columns

```
titanic["Fare"].fillna(titanic["Fare"].median())
titanic
```

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q
1	893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S
2	894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q
3	895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S
4	896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...
413	1305	3	Spector, Mr. Woolf	male	NaN	0	0	A.5. 3236	8.0500	NaN	S
414	1306	1	Oliva y Ocana, Dona. Fermina	female	39.0	0	0	PC 17758	108.9000	C105	C
415	1307	3	Saether, Mr. Simon Sivertsen	male	38.5	0	0	SOTON/O.Q. 3101262	7.2500	NaN	S
416	1308	3	Ware, Mr. Frederick	male	NaN	0	0	359309	8.0500	NaN	S
417	1309	3	Peter, Master. Michael J	male	NaN	1	1	2668	22.3583	NaN	C

418 rows × 11 columns

Conclusion:

Thus, we can conclude that we have successfully implemented the basic steps, using Python , that we should follow whenever we are working on a dataset.

Assignment No.: 2

Problem Statement:

Perform the following operations using R/Python on the data sets Compute and display summary statistics for each feature available in the dataset. (e.g. minimum value, maximum value, mean, range, standard deviation, variance and percentiles. Data Visualization-Create a histogram for each feature in the dataset to illustrate the feature distributions, Data cleaning, Data integration, Data transformation, Data model building (e.g, Classification)

Objective:

The goal is to make it easier to identify patterns, trends and outliers in large data sets.

Theory:

1) **Data Visualization:** Data visualization is the graphical representation of information and data. By using visual elements like charts, graph and maps, data visualization tools provide an accessible way to see and understand trends, outliers and patterns in data.

2) **Data Cleaning:** Data cleaning is the process of fixing or removing incorrect, corrupted, incorrectly formatted, duplicate, or incomplete data within a dataset. When combining multiple data sources, there are many opportunities for data to be duplicated or mislabeled. So, for that purpose data cleaning is required.

3) **Data Transformation:** Data transformation is the process of changing the format, structure, or values of data. For data analytics projects, data may be transformed at two stages of the data pipeline. Organizations that use on-premises data warehouses generally use an ETL(extract, transform, load) process, in which data transformation is the middle process.

4) **Data Integration:** Data integration is the process of combining data from different sources into a single, unified view. Integration begins with the ingestion process, and includes steps such as cleansing, ETL mapping, and transformation. Data integration ultimately enables analytics tools to produce effective, actionable business intelligence.

5) **Data Model Building:** Data modeling is the process of conceptualizing and visualizing how data will be captured, stored, and used by an organization. The ultimate aim of data modeling is to establish clear data standards for your entire organization.

Sample Input:

Covid 19 India dataset (covid_19_india.csv)

Output:

Importing required libraries-

```
import numpy as nm
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from google.colab import drive
drive.mount('/content/drive')
df = pd.read_csv("/content/drive/MyDrive/DSML/covid_19_india.csv")
```

```
df.columns
```

```
Index(['Sno', 'Date', 'Time', 'State/UnionTerritory',
       'ConfirmedIndianNational', 'ConfirmedForeignNational', 'Cured',
       'Deaths', 'Confirmed'],
      dtype='object')
```

Mean, Median and Mode-

```
df.Deaths.count()
```

```
18110
```

```
df.Time.min()
```

```
'10:00 AM'
```

```
df.Time.min()
```

```
'9:30 PM'
```

```
df.Deaths.mean()
```

```
4052.402263942573
```

```
df.Deaths.median()
```

```
588.0
```

```
df.Confirmed.mode()
```

```
0    1
dtype: int64
```

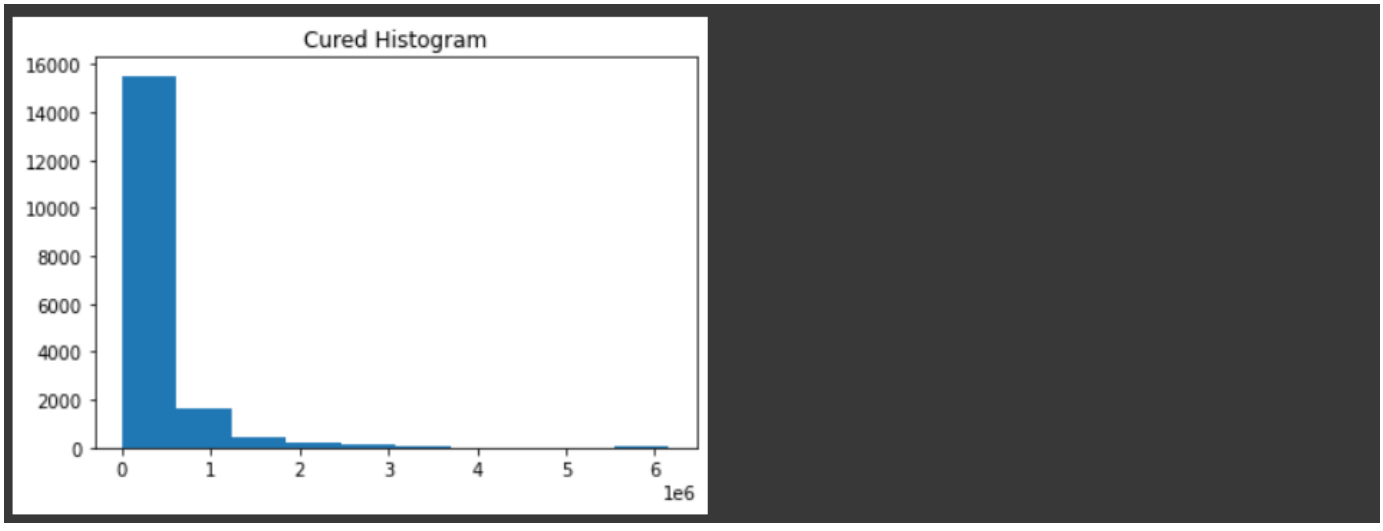
Calculating variance-

```
df.var()
```

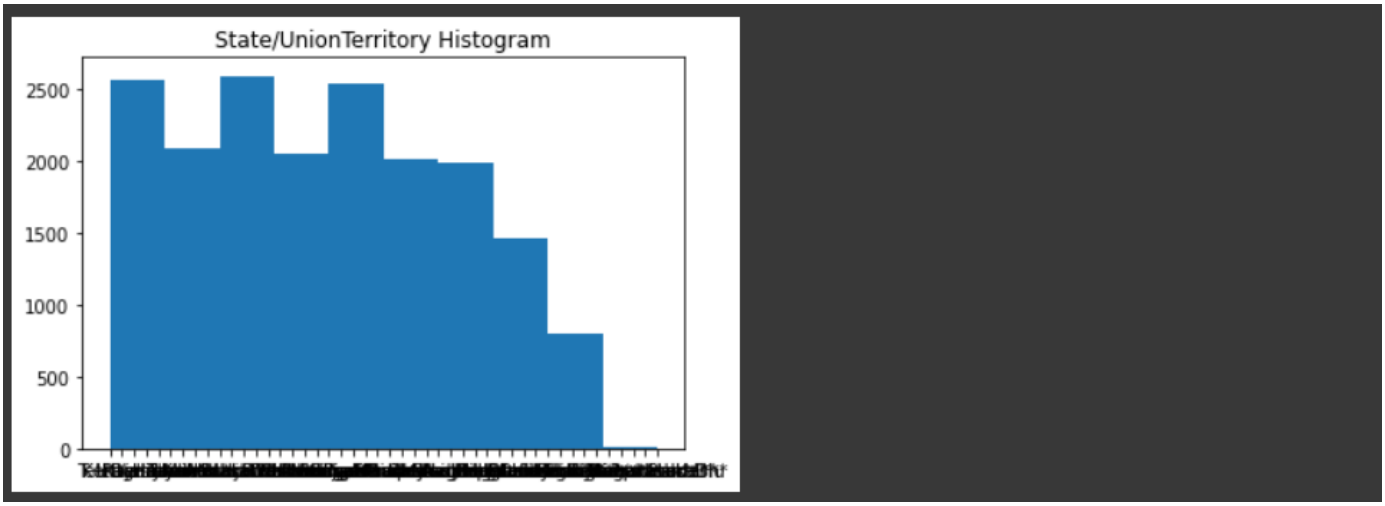
```
Sno          2.733252e+07
Cured        3.780908e+11
Deaths       1.192262e+08
Confirmed    4.305313e+11
dtype: float64
```

Displaying Histogram-

```
plt.hist(df['Cured'])
plt.title('Cured Histogram')
plt.show()
```




```
plt.hist(df['State/UnionTerritory'])
plt.title('State/UnionTerritory Histogram')
plt.show()
```



Data Transformation-

```
df.dtypes
```

```
Sno                       int64
Date                     object
Time                     object
State/UnionTerritory     object
ConfirmedIndianNational  object
ConfirmedForeignNational object
Cured                     int64
Deaths                    int64
Confirmed                 int64
dtype: object
```

```
data_types_dict = {'Deaths': float}
df = df.astype(data_types_dict)
df.dtypes
```

```
Sno                       int64
Date                     object
Time                     object
State/UnionTerritory     object
ConfirmedIndianNational  object
ConfirmedForeignNational object
Cured                     int64
Deaths                  float64
Confirmed                 int64
dtype: object
```

Data Cleaning-

```
df.isnull()
```

	Sno	Date	Time	State/UnionTerritory	ConfirmedIndianNational	ConfirmedForeignNational	Cured	Deaths	Confirmed
0	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...
18105	False	False	False	False	False	False	False	False	False
18106	False	False	False	False	False	False	False	False	False
18107	False	False	False	False	False	False	False	False	False
18108	False	False	False	False	False	False	False	False	False
18109	False	False	False	False	False	False	False	False	False

18110 rows × 9 columns

```
df.isnull().sum()
```

```
Sno                        0
Date                       0
Time                       0
State/UnionTerritory       0
ConfirmedIndianNational    0
ConfirmedForeignNational   0
Cured                      0
Deaths                     0
Confirmed                  0
dtype: int64
```

Since there are 0 null values in every attribute of the dataset there is no need of data cleaning.

Conclusion:

In this assignment we learned the use of pandas and matplotlib to get the count, min, max, mean etc. Also, how to display the histogram using matplotlib also data cleaning and transformation.

Assignment No.: 3

Problem Statement:

Write a program to do: A dataset collected in a cosmetics shop showing details of customers and whether or not they responded to a special offer to buy a new lip-stick is shown in the table below. Use this dataset to build a decision tree, with Buys as the target variable, to help in buying lipsticks in the future. Find the root node of the decision tree. According to the decision tree you have made from the previous training data set, what is the decision for the test data: [Age < 21, Income = Low, Gender = Female, Marital Status = Married]?

Objective:

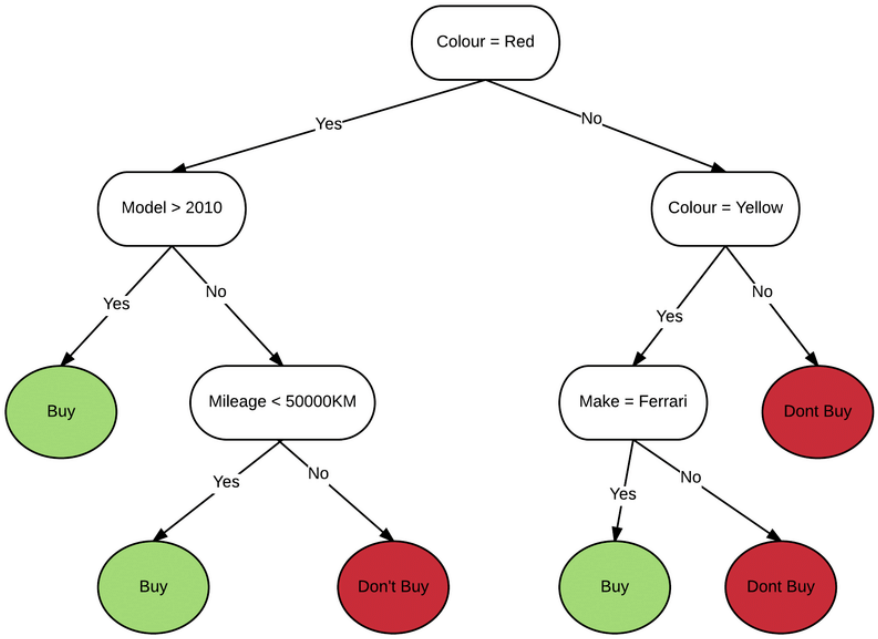
Objective of the Decision Tree is to create a training model that can be used to predict the class or value of the target variable by learning simple decision rules inferred from prior data(training data).

Theory:

Classification is a two-step process, learning step and prediction step, in machine learning. Decision Tree is one of the easiest and popular classification algorithms to understand and interpret.

1) What is a Decision Tree?

Decision Tree is the most powerful and popular tool for classification and prediction. A Decision tree is a flowchart-like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label.



2) Decision Tree representation:

Decision trees classify instances by sorting them down the tree from the root to some leaf node, which provides the classification of the instance. An instance is classified by starting at the root node of the tree, testing the attribute specified by this node, then moving down the tree branch corresponding to the value of the attribute as we can see in the diagram above.

3) What is Gini Index:

Gini Index is a score that evaluates how accurate a split is among the classified groups. Gini index evaluates a score in the range between 0 and 1, where 0 is when all observations belong to one class, and 1 is a random distribution of the elements within classes. In this case, we want to have a Gini index score as low as possible. Gini Index is the evaluation metrics we shall use to evaluate our Decision Tree Model.

Algorithm:

In a decision tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree. This algorithm compares the values of the root attribute with the record (basically real dataset) attribute and, based on the comparison, follows the branch and jumps to the next node.

Step-1: Begin the tree with the root node, says S, which contains the complete dataset.
Step-2: Find the best attribute in the dataset using Attribute Selection Measure (ASM).
Step-3: Divide the S into subsets that contains possible values for the best attributes.
Step-4: Generate the decision tree node, which contains the best attribute.
Step-5: Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node

Input:

Sales data (sales.csv)

Output:

Importing required libraries

```
import numpy as np
import pandas as pd
from google.colab import drive
from sklearn.preprocessing import LabelEncoder
```

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import export_graphviz
from IPython.display import Image
drive.mount('/content/drive')
data = pd.read_csv("/content/drive/MyDrive/DSML/sales.csv")
data
```

	Age	Income	Gender	MartialStatus	Buys
0	<21	High	Male	Single	No
1	<21	High	Male	Married	No
2	21-35	High	Male	Single	Yes
3	>35	Medium	Male	Single	Yes
4	>35	Low	Female	Single	Yes
5	>35	Low	Female	Married	No
6	21-35	Low	Female	Married	Yes
7	<21	Medium	Male	Single	No
8	<21	Low	Female	Married	Yes
9	>35	Medium	Female	Single	Yes
10	<21	Medium	Female	Married	Yes
11	21-35	Medium	Male	Married	Yes
12	21-35	High	Female	Single	Yes
13	>35	Medium	Male	Married	No

```
data.describe()
```

	Age	Income	Gender	MartialStatus	Buys
count	14	14	14	14	14
unique	3	3	2	2	2
top	<21	Medium	Male	Single	Yes
freq	5	6	7	7	9

```
data['Buys'].value_counts()
```

```
Yes    9
No     5
Name: Buys, dtype: int64
```

```
le=LabelEncoder();
x=data.iloc[:,:-1]

x=x.apply(le.fit_transform)
print("Age with encoded value: ", list( zip(data.iloc[:,0], x.iloc[:,0])))
print("\nIncome with encoded value: ", list( zip(data.iloc[:,1], x.iloc[:,1])))
print("\nGender with encoded value: ", list( zip(data.iloc[:,2], x.iloc[:,2])))
print("\nmaritialStatus with encoded value :", list( zip(data.iloc[:,3], x.iloc[:
,3])))
```

```
Age with encoded value: [('<21', 1), ('<21', 1), ('21-35', 0), ('>35', 2), ('>35', 2),
('>35', 2), ('21-35', 0), ('<21', 1), ('<21', 1), ('>35', 2), ('<21', 1), ('21-35', 0),
('21-35', 0), ('>35', 2)]

Income with encoded value: [('High', 0), ('High', 0), ('High', 0), ('Medium', 2),
('Low', 1), ('Low', 1), ('Low', 1), ('Medium', 2), ('Low', 1), ('Medium', 2), ('Medium',
2), ('Medium', 2), ('High', 0), ('Medium', 2)]

Gender with encoded value: [('Male', 1), ('Male', 1), ('Male', 1), ('Male', 1),
('Female', 0), ('Female', 0), ('Female', 0), ('Male', 1), ('Female', 0), ('Female',
0), ('Female', 0), ('Male', 1), ('Female', 0), ('Male', 1)]

maritialStatus with encoded value: [('Single', 1), ('Married', 0), ('Single', 1),
('Single', 1), ('Single', 1), ('Married', 0), ('Married', 0), ('Single', 1),
('Married', 0), ('Single', 1), ('Married', 0), ('Married', 0), ('Single', 1),
('Married', 0)]
```

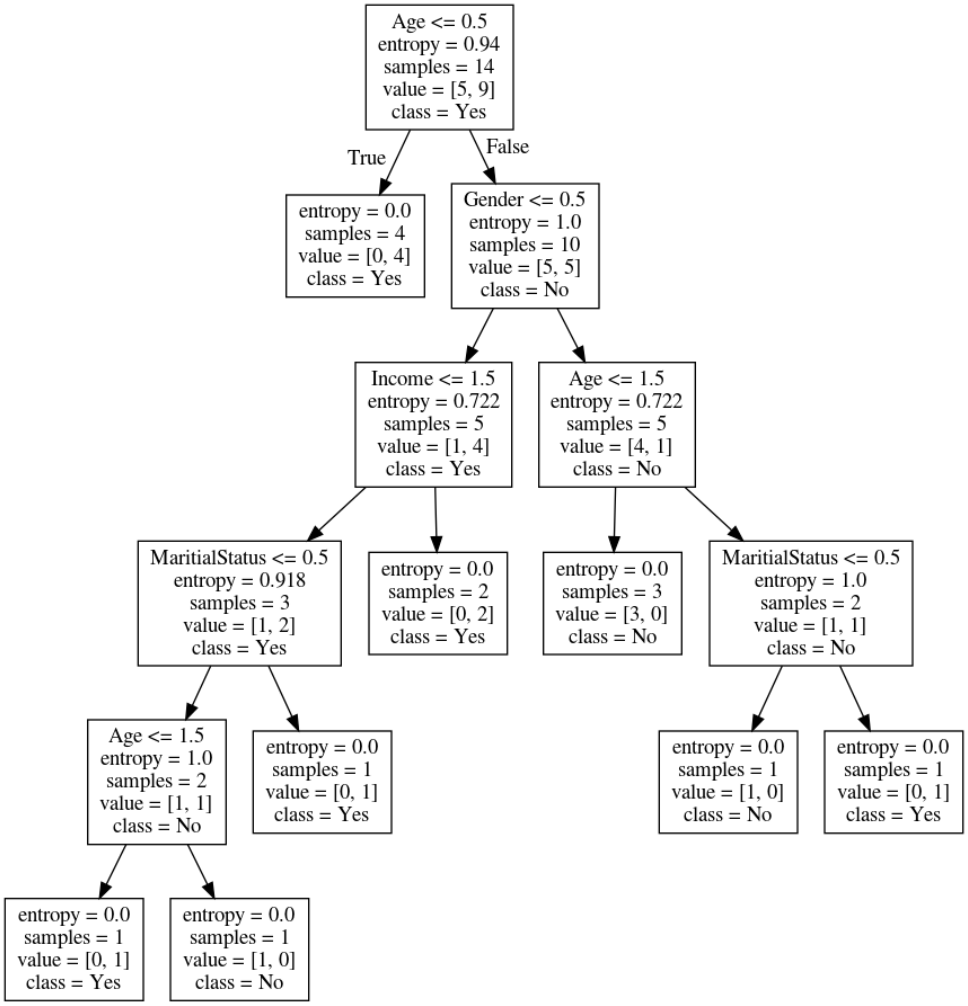
```
y=data.iloc[:,-1]
classifier=DecisionTreeClassifier(criterion='entropy')
classifier.fit(x,y)
```

```
DecisionTreeClassifier(criterion='entropy')
```

```
test_x=np.array([1,1,0,0])
pred_y=classifier.predict([test_x])
print("Predicted class for input [Age < 21, Income = Low,Gender = Female,Marital
Status = Married]\n", test_x," is ",pred_y[0])
```

```
Predicted class for input [Age < 21, Income = Low,Gender = Female,Marital Status = Married]
 [1 1 0 0]  is  Yes
```

```
export_graphviz(classifier,out_file="data.dot",feature_names=x.columns,class_name
s=["No","Yes"])
```



Conclusion:

Hence, we got to learn modules like numpy, pandas and sklearn to get decision tree as output as we can see above.

Assignment No.: 4

Problem Statement:

Write a program to do following: We have given a collection of 8 points. P1=[0.1,0.6] P2=[0.15,0.71] P3=[0.08,0.9] P4=[0.16,0.85] P5=[0.2,0.3] P6=[0.25,0.5] P7=[0.24,0.1] P8=[0.3,0.2]. Perform the k-mean clustering with initial centroids as m1 = P1 = Cluster#1 = C1 and m2 = P8 = cluster#2 = C2. Answer the following

1] Which cluster does P6 belongs to?

2] What is the population of cluster around m2?

3] What is updated value of m1 and m2?

Objective:

To Perform K-mean clustering with given initial centroids on the given dataset of 8 points using Python language.

Theory:

What is K-means Clustering?

K-means (Macqueen, 1967) is one of the simplest unsupervised learning algorithms that solve the well-known clustering problem. K-means clustering is a method of vector quantization, originally from signal processing, that is popular for cluster analysis in data mining.

K-means Clustering Method:

If k is given, the K-means algorithm can be executed in the following steps:

1) Partition of objects into k non-empty subsets
2) Identifying the cluster centroids (mean point) of the current partition.
3) Assigning each point to a specific cluster
4) Compute the distances from each point and allot points to the cluster where the distance from the centroid is minimum.
5) After re-allocating the points, find the centroid of the new cluster formed.

Pandas:

Pandas is a Python library for data analysis. Started by Wes McKinney in 2008 out of a need for a powerful and flexible quantitative analysis tool, pandas have grown into one of the most popular Python libraries. It has an extremely active community of contributors**.**

NumPy:

NumPy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, Fourier transform, and matrices. NumPy was

created in 2005 by Travis Oliphant. It is an open-source project and you can use it freely.

Matplotlib:

Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack.

Algorithm:

1) Import the Required Package
2) Create dataset using DataFrame
3) Find centroid points
4) plot the given points
5) for i in centroids():
6) plot given elements with centroid elements
7) import KMeans class and create object of it
8) using labels find population around centroid
9) Find new centroid.

Input:

Sample Input-

```
{
        'Points' : ['P1','P2','P3','P4','P5','P6','P7','P8',],
        'x_coordinate' : [0.1,0.15,0.08,0.16,0.2,0.25,0.24,0.3],
        'y_coordinate' : [0.6,0.71,0.9,0.85,0.3,0.5,0.1,0.2]
}
```

Output:

Importing required libraries and storing dataset-

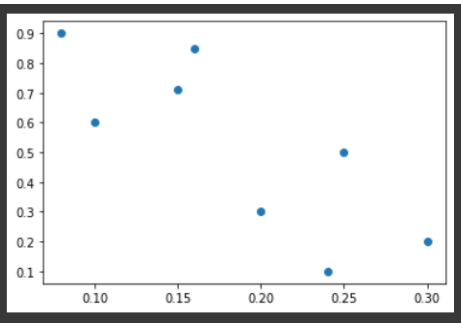
```
import numpy as np
import pandas as pd
import math
from matplotlib import pyplot as plt
%matplotlib inline
```

```
x = np.array([0.1,0.15,0.08,0.16,0.2,0.25,0.24,0.3])
y = np.array([0.6,0.71,0.9,0.85,0.3,0.5,0.1,0.2])
```

	Points	x_coordinate	y_coordinate
0	P1	0.10	0.60
1	P2	0.15	0.71
2	P3	0.08	0.90
3	P4	0.16	0.85
4	P5	0.20	0.30
5	P6	0.25	0.50
6	P7	0.24	0.10
7	P8	0.30	0.20

Plotting points-

```
plt.plot(x,y,"o")
plt.show()
```




```
def eucledian_distance(x1,y1,x2,y2):
    return math.sqrt((x1-x2)**2+(y1-y2)**2)

def manhattan_distance(x1,y1,x2,y2):
    return math.fabs(x1-x2)+math.fabs(y1-y2)
```

```
def returnCluster(m1,m2,x_co,y_co):
    distance1=manhattan_distance(m1[0],m1[1],x_co,y_co)

    distance2=manhattan_distance(m2[0],m2[1],x_co,y_co)

    if(distance1<distance2):
        return 1;
    else:
        return 2;
```

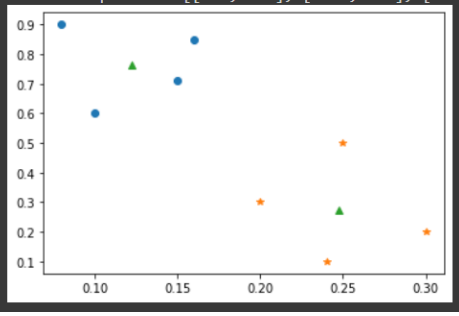
Forming the clusters and its visualization-

```
m1=[0.1,0.6]
m2=[0.3,0.2]
#difference and iteration is for controlling iteration
difference = math.inf
threshold=0.02
iteration=0;
while difference>threshold: #use any one condition #iteration one is easy
    print("Iteration ",iteration, " : m1=",m1, " m2=",m2)
    cluster1=[];
    cluster2=[];

    #step1 assign all points to nearest cluster
    for i in range(0,np.size(x)):
        clusterNumber=returnCluster(m1,m2,x[i],y[i])
        point=[x[i],y[i]]
        if clusterNumber==1:
            cluster1.append(point);
        else:
            cluster2.append(point)

    print("cluster 1", cluster1,"\nCLuster 2: ", cluster2)

    #step 2: Calculating new centriod for cluster1
    m1_old=m1;
    m1=[]
    m1=np.mean(cluster1, axis=0) #axis=0 means columnwise

    #calculating centroid for cluster2
    m2_old=m2;
    m2=[];
    m2=np.mean(cluster2,axis=0)
    print("m1 = ",m1," m2=",m2)

    #adjusting diffrences of adjustment between m1 nd m1_old
    xAvg=0.0;
    yAvg=0.0;
    xAvg=math.fabs(m1[0]-m1_old[0])+math.fabs(m2[0]-m2_old[0])
    xAvg=xAvg/2;

    yAvg=math.fabs(m1[1]-m1_old[1])+math.fabs(m2[1]-m2_old[1])
    yAvg=yAvg/2;

    if(xAvg>yAvg):
        difference=xAvg;
    else:
        difference=yAvg;

    print("Difference  : ", difference)
    iteration+=1;
    print("")
```

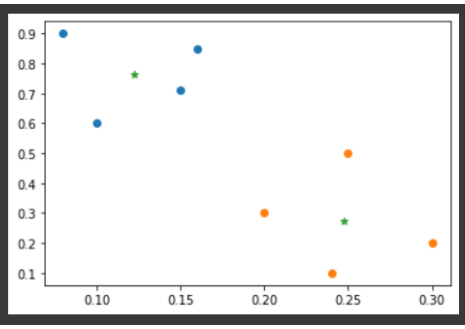
```
Iteration  0  : m1= [0.1, 0.6]  m2= [0.3, 0.2]
cluster 1 [[0.1, 0.6], [0.15, 0.71], [0.08, 0.9], [0.16, 0.85], [0.25, 0.5]]
CLuster 2:  [[0.2, 0.3], [0.24, 0.1], [0.3, 0.2]]
m1 =  [0.148 0.712]  m2= [0.24666667 0.2       ]
Difference  :  0.05600000000000001

Iteration  1  : m1= [0.148 0.712]  m2= [0.24666667 0.2       ]
cluster 1 [[0.1, 0.6], [0.15, 0.71], [0.08, 0.9], [0.16, 0.85]]
CLuster 2:  [[0.2, 0.3], [0.25, 0.5], [0.24, 0.1], [0.3, 0.2]]
m1 =  [0.1225 0.765 ]  m2= [0.2475 0.275 ]
Difference  :  0.06400000000000002

Iteration  2  : m1= [0.1225 0.765 ]  m2= [0.2475 0.275 ]
cluster 1 [[0.1, 0.6], [0.15, 0.71], [0.08, 0.9], [0.16, 0.85]]
CLuster 2:  [[0.2, 0.3], [0.25, 0.5], [0.24, 0.1], [0.3, 0.2]]
m1 =  [0.1225 0.765 ]  m2= [0.2475 0.275 ]
Difference  :  0.0
```

```
print("Cluster 1 centroid : m1 = ",m1)
print("CLuster 1 points: ", cluster1)
print("Cluster 2 centroid : m2 = ",m2)
print("CLuster 2 points: ", cluster2)

clust1=np.array(cluster1)
clust2=np.array(cluster2)

#cluster 1 points
plt.plot(clust1[:,0],clust1[:,1],"o")

#cluster2 points
plt.plot(clust2[:,0], clust2[:,1],"*")

#centroids
plt.plot([m1[0],m2[0]],[m1[1],m2[1]],"^")
plt.show()
```

```
Cluster 1 centroid : m1 =  [0.1225 0.765 ]
CLuster 1 points:  [[0.1, 0.6], [0.15, 0.71], [0.08, 0.9], [0.16, 0.85]]
Cluster 2 centroid : m2 =  [0.2475 0.275 ]
CLuster 2 points:  [[0.2, 0.3], [0.25, 0.5], [0.24, 0.1], [0.3, 0.2]]
```




```
plt.scatter(clust1[:,0],clust1[:,1])
plt.scatter(clust2[:,0],clust2[:,1])
plt.scatter([m1[0],m2[0]],[m1[1],m2[1]],marker="*")
plt.show()
```



Answer the following

1] Which cluster does P6 belongs to?

=> 2nd Cluster

2] What is the population of cluster around m2?

=> 4

3] What is updated value of m1 and m2?

=> [0.247 0.275]

Conclusion:

Thus, we have successfully implemented K-means clustering algorithm using the given initial centroids to find clusters of different points.

Assignment No.: 5

Problem Statement:

Visualize the data using R/Python by plotting the graphs for assignment no. 1 and 2. Use Scatter plot, bar plot, Box plot and Histogram OR Perform the data visualization operations using Tableau for the given dataset. Consider a suitable data set.

Objective:

Data visualization is the practice of translating information into a visual context, such as a map or graph, to make data easier for the human brain to understand and pull insights from. The main goal of data visualization is to make it easier to identify patterns, trends and outliers in large data sets.

Theory:

The process of finding trends and correlations in our data by representing it pictorially is called Data Visualization. To perform data visualization in python, we can use various python data visualization modules such as Matplotlib, Seaborn, Plotly, etc. in this assignment two data visualization modules has been used which are as following:

Matplotlib:

Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack.

Seaborn:

Seaborn is a library that uses Matplotlib underneath to plot graphs. It will be used to visualize random distributions.

Input:

Tips dataset (tips.csv)

Output:

Importing required libraries-

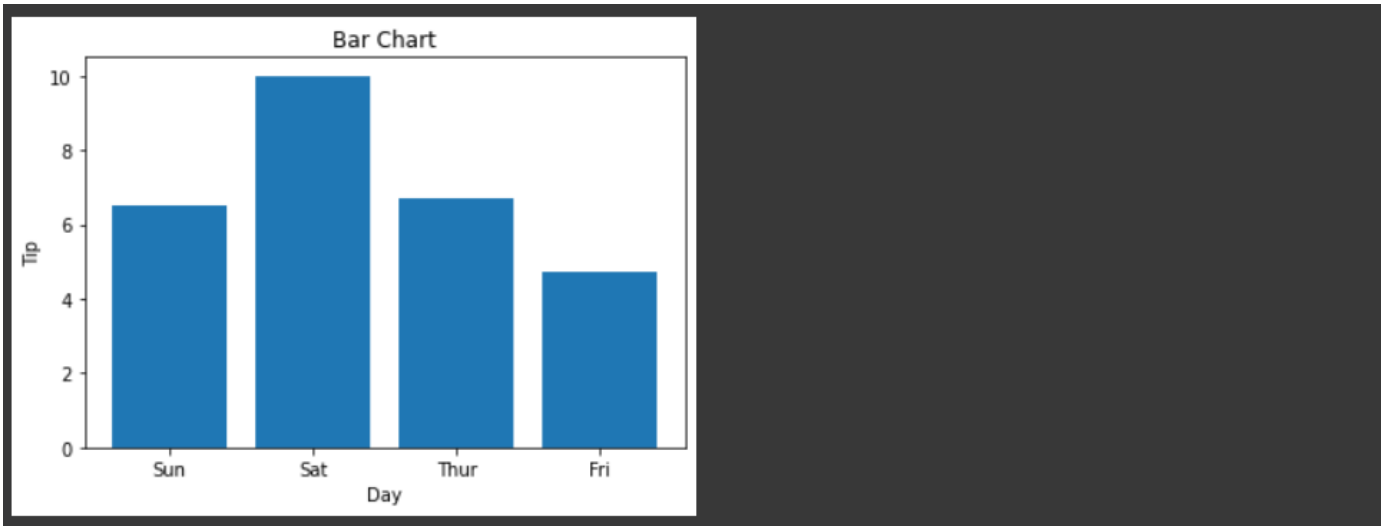
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from google.colab import drive
drive.mount("/content/drive")
df = pd.read_csv("/content/drive/MyDrive/DSML/tips.csv")
df
```

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4
...	...	...	...	...	...	...	...
239	29.03	5.92	Male	No	Sat	Dinner	3
240	27.18	2.00	Female	Yes	Sat	Dinner	2
241	22.67	2.00	Male	Yes	Sat	Dinner	2
242	17.82	1.75	Male	No	Sat	Dinner	2
243	18.78	3.00	Female	No	Thur	Dinner	2

244 rows × 7 columns

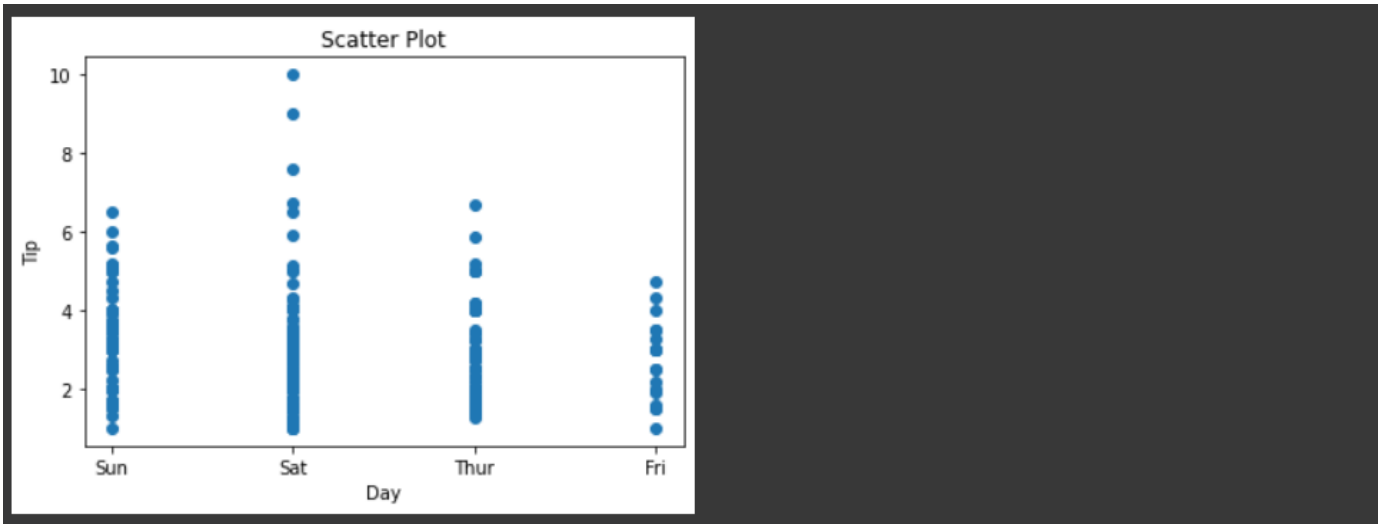
Displaying bar chart-

```
plt.bar(df['day'], df['tip'])
plt.title('Bar Chart')
plt.xlabel('Day')
plt.ylabel('Tip')
plt.show()
```



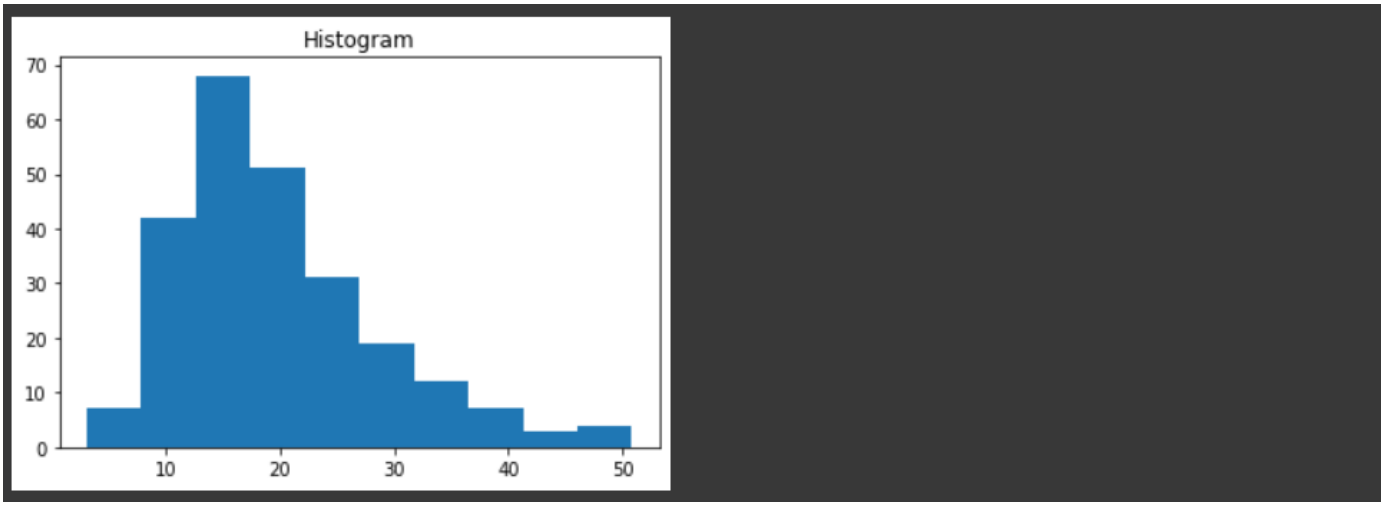
Displaying scatter plot-

```
plt.scatter(df['day'], df['tip'])
plt.title('Scatter Plot')
plt.xlabel('Day')
plt.ylabel('Tip')
plt.show()
```



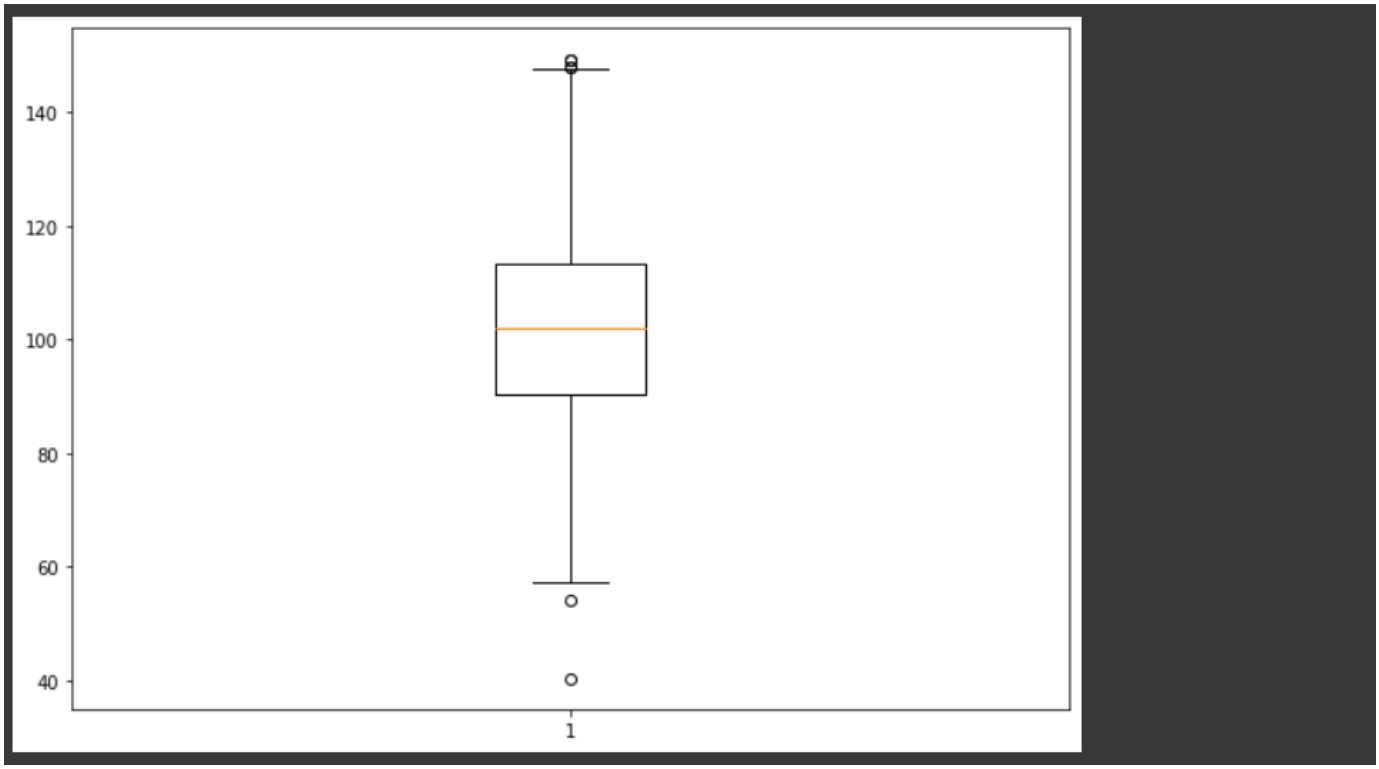
Displaying histogram-

```
plt.hist(df['total_bill'])
plt.title('Histogram')
plt.show()
```



Displaying Box plot-

```
np.random.seed(10)
data = np.random.normal(100, 20, 200)
fig = plt.figure(figsize =(10, 7))
plt.boxplot(data)
plt.show()
```



Conclusion:

Hence, we got to learn matplotlib as well as seaborn python modules that are used for the data visualization as well as we got to learn how data should be effectively visualized.

Assignment No.: 6

Problem Statement:

Identify problem statement. Use Semi or unstructured data set. Define 3 to 4 objectives. Perform 1. Data Interpretation, 2. Data preprocessing, 3. Data Modeling (perform classification, Prescriptive Analysis (if required and fits for the data set)), and 4.data visualization. (Mini project is to be performed in a group of 3 to 4 students).

Objective:

The primary purpose of a machine learning project is to discover patterns in the user data and then make predictions based on these and intricate patterns for answering business questions and solving business problems. Machine learning helps in analyzing the data as well as identifying trends.

Theory:

A) Data interpretation:

Data interpretation is the process of reviewing data and arriving at relevant conclusions using various analytical methods. Data analysis assists researchers in categorizing, manipulating, and summarizing data to answer critical questions.

Data Interpretation Steps:

1) **Gather the data:** The very first step in data interpretation is gathering all relevant data. You can do this by first visualizing it in a bar, graph, or pie chart. This step aims to analyze the data accurately and without bias. Now is the time to recall how you conducted your research.

2) **Develop your discoveries:** This is a summary of your findings. Here, you thoroughly examine the data to identify trends, patterns, or behavior. If you are researching a group of people using a sample population, this is the section where you examine behavioral patterns. You can compare these deductions to previous data sets, similar data sets, or general hypotheses in your industry. This step's goal is to compare these deductions before drawing any conclusions.

3) **Draw Conclusions:** After you've developed your findings from your data sets, you can draw conclusions based on your discovered trends. Your findings should address the questions that prompted your research. If they

do not respond, inquire about why; it may produce additional research or questions.

4) **Give recommendations:** Recommendations are a summary of your findings and conclusions, they should be brief. There are only two options for recommendations; you can either recommend a course of action or suggest additional research.

Example: Let's say your users fall into four age groups. So, a company can see which age group likes their content or product. Based on bar charts or pie charts, they can develop a marketing strategy to reach uninvolved groups or an outreach strategy to grow their core user base.

B) Data-Preprocessing:

Data preprocessing is a data mining technique which is used to transform the raw data in a useful and efficient format. The data can have many irrelevant and missing parts. To handle this part, data cleaning is done. It involves handling of missing data, noisy data etc.

1) **Missing Data:** This situation arises when some data is missing in the data. It can be handled in various ways. Some of them are:
   a) **Ignore the tuples:** This approach is suitable only when the dataset we have is quite large and multiple values are missing within a tuple.
   b) **Fill the Missing values:** There are various ways to do this task. You can choose to fill the missing values manually, by attribute mean or the most probable value.

2) **Noisy Data:** Noisy data is a meaningless data that can't be interpreted by machines. It can be generated due to faulty data collection, data entry errors etc. It can be handled in following ways:
   a) **Binning Method:** This method works on sorted data in order to smooth it. The whole data is divided into segments of equal size and then various methods are performed to complete the task. Each segmented is handled separately. One can replace all data in a segment by its mean or boundary values can be used to complete the task.
   b) **Regression:** Here data can be made smooth by fitting it to a regression function. The regression used may be linear (having one

independent variable) or multiple (having multiple independent variables).

c) **Clustering:** This approach groups the similar data in a cluster. The outliers may be undetected or it will fall outside the clusters.

C) Data-Preprocessing:

This step is taken in order to transform the data in appropriate forms suitable for mining process. This involves following ways:

1) **Normalization:** It is done in order to scale the data values in a specified range (-1.0 to 1.0 or 0.0 to 1.0)
2) **Attribute Selection:** In this strategy, new attributes are constructed from the given set of attributes to help the mining process.
3) **Discretization:** This is done to replace the raw values of numeric attribute by interval levels or conceptual levels.
4) **Concept Hierarchy Generation:** Here attributes are converted from lower level to higher level in hierarchy. For Example-The attribute "city" can be converted to "country".

D) Data Reduction:

Since data mining is a technique that is used to handle huge amounts of data. While working with a huge volume of data, analysis became harder in such cases. In order to get rid of this, we use data reduction techniques. It aims to increase the storage efficiency and reduce data storage and analysis costs.

The various steps to data reduction are:

1) **Data Cube Aggregation:** Aggregation operation is applied to data for the construction of the data cube.
2) **Attribute Subset Selection:** The highly relevant attributes should be used, rest all can be discarded. For performing attribute selection, one can use level of significance and p- value of the attribute. The attributes having p-value greater than significance level can be discarded.
3) **Numerosity Reduction:** This enables us to store the model of data instead of whole data, for example: Regression Models.
4) **Dimensionality Reduction:** This reduces the size of data by encoding mechanisms. It can be lossy or lossless. If after reconstruction from compressed data, original data can be retrieved, such reduction is called lossless reduction, else it is called lossy reduction. The two effective methods of dimensionality reduction are: Wavelet transforms and PCA (Principal Component Analysis).

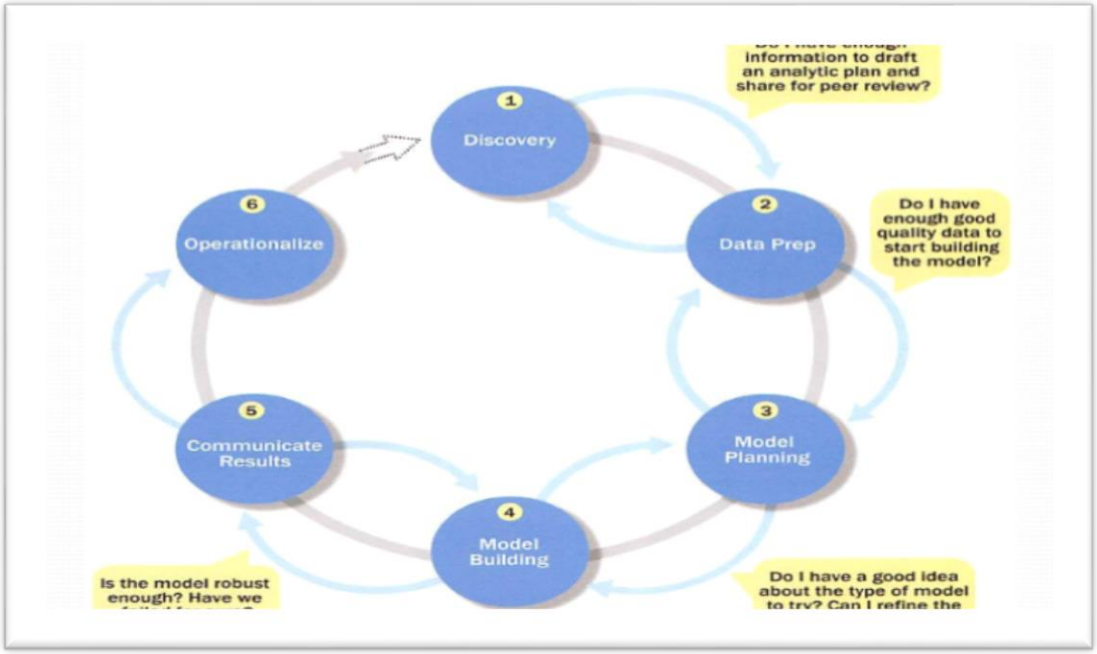
E) Data Modeling:

Data modeling is the process of creating a visual representation of either a whole information system or parts of it to communicate connections between data points and structures. The goal is to illustrate the types of data used and stored within the system, the relationships among these data types, the ways the data can be grouped and organized and its formats and attributes.

F) Data Visualization:

Data visualization is a way to represent information graphically, highlighting patterns and trends in data and helping the reader to achieve quick insights.

Data Analytics Lifecycle:



Few Suggested Mini Project Topics:

1) Movie Ticket Pricing System
2) Big-Mart Sales Prediction ML Project
3) House Price Prediction
4) Fake Or Real News Prediction
5) Share price prediction

Copy No.	**Copy Holder**
1	TnL Coordinator (Soft Copy)
2	Practical In-charge (Hard Copy)
3	Lab Assistant (Soft Copy)

Title: Distribution List Section: Annexure