```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

Import the dataset

```
data = pd.read_csv("covid_19_india.csv")
```

```
data
```

	Sno	Date	Time	State/UnionTerritory	ConfirmedIndianNational	ConfirmedFor
0	1	30/01/20	6:00 PM	Kerala	1	
1	2	31/01/20	6:00 PM	Kerala	1	
2	3	01/02/20	6:00 PM	Kerala	2	
3	4	02/02/20	6:00 PM	Kerala	3	
4	5	03/02/20	6:00 PM	Kerala	3	
...	...	...	...	...	...	
7081	7082	07/10/20	8:00 AM	Telengana	-	
7082	7083	07/10/20	8:00 AM	Tripura	-	
7083	7084	07/10/20	8:00 AM	Uttarakhand	-	
7084	7085	07/10/20	8:00 AM	Uttar Pradesh	-	
7085	7086	07/10/20	8:00 AM	West Bengal	-	

7086 rows × 9 columns

Describe the dataset

```
data.describe()
```

	Sno	Cured	Deaths	Confirmed
count	7086.000000	7.086000e+03	7086.000000	7.086000e+03
mean	3543.500000	3.423502e+04	836.188682	4.534495e+04
std	2045.696336	9.913313e+04	2997.188810	1.263801e+05
min	1.000000	0.000000e+00	0.000000	0.000000e+00
25%	1772.250000	3.800000e+01	1.000000	1.660000e+02
50%	3543.500000	1.499500e+03	22.000000	2.774000e+03
75%	5314.750000	1.664325e+04	396.000000	2.573375e+04
max	7086.000000	1.179726e+06	38717.000000	1.465911e+06

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7086 entries, 0 to 7085
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype 
---  ------                    --------------  ----- 
 0   Sno                       7086 non-null   int64 
 1   Date                      7086 non-null   object
 2   Time                      7086 non-null   object
 3   State/UnionTerritory      7086 non-null   object
 4   ConfirmedIndianNational   7086 non-null   object
 5   ConfirmedForeignNational  7086 non-null   object
 6   Cured                     7086 non-null   int64 
 7   Deaths                    7086 non-null   int64 
 8   Confirmed                 7086 non-null   int64 
dtypes: int64(4), object(5)
memory usage: 498.4+ KB
```

```
data.tail()
```

	Sno	Date	Time	State/UnionTerritory	ConfirmedIndianNational	ConfirmedFor
			8:00			

```
data.columns
```

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

```
no_use_rows = data.loc[pd.to_numeric(data['ConfirmedIndianNational'], errors='coerce').isnul]
no_use_rows
```

	Sno	Date	Time	State/UnionTerritory	ConfirmedIndianNational	ConfirmedFor
446	447	29/03/20	7:30 PM	Andhra Pradesh	-	
447	448	29/03/20	7:30 PM	Andaman and Nicobar Islands	-	
448	449	29/03/20	7:30 PM	Bihar	-	
449	450	29/03/20	7:30 PM	Chandigarh	-	
450	451	29/03/20	7:30 PM	Chhattisgarh	-	
...	...	...	...	...	...	
7081	7082	07/10/20	8:00 AM	Telengana	-	
7082	7083	07/10/20	8:00 AM	Tripura	-	
7083	7084	07/10/20	8:00 AM	Uttarakhand	-	
7084	7085	07/10/20	8:00 AM	Uttar Pradesh	-	
7085	7086	07/10/20	8:00 AM	West Bengal	-	

6640 rows × 9 columns

```
del data['ConfirmedIndianNational']#deleting column
data.head()
```

	Sno	Date	Time	State/UnionTerritory	ConfirmedForeignNational	Cured	Deaths
0	1	30/01/20	6:00 PM	Kerala	0	0	0
1	2	31/01/20	6:00 PM	Kerala	0	0	0
2	3	01/02/20	6:00	Kerala	0	0	0

```
del data['ConfirmedForeignNational']#deleting column
data.head(2119)
```

	Sno	Date	Time	State/UnionTerritory	Cured	Deaths	Confirmed
0	1	30/01/20	6:00 PM	Kerala	0	0	1
1	2	31/01/20	6:00 PM	Kerala	0	0	1
2	3	01/02/20	6:00 PM	Kerala	0	0	2
3	4	02/02/20	6:00 PM	Kerala	0	0	3
4	5	03/02/20	6:00 PM	Kerala	0	0	3
...	...	...	...	...	...	...	...
2114	2115	20/05/20	8:00 AM	Jharkhand	127	3	231
2115	2116	20/05/20	8:00 AM	Karnataka	544	40	1397
2116	2117	20/05/20	8:00 AM	Kerala	497	4	642
2117	2118	20/05/20	8:00 AM	Ladakh	43	0	43
2118	2119	20/05/20	8:00 AM	Madhya Pradesh	2630	258	5465

2119 rows × 7 columns

```
data.rename(columns = {'State/UnionTerritory':'StatesUniounTerritories'}, inplace = True)
data
```

	Sno	Date	Time	StatesUniounTerritories	Cured	Deaths	Confirmed
0	1	30/01/20	6:00 PM	Kerala	0	0	1
1	2	31/01/20	6:00 PM	Kerala	0	0	1
2	3	01/02/20	6:00 PM	Kerala	0	0	2
3	4	02/02/20	6:00 PM	Kerala	0	0	3

```
data["StatesUniounTerritories"]= data["StatesUniounTerritories"].str.replace(r'\W',"")
data
```

	Sno	Date	Time	StatesUniounTerritories	Cured	Deaths	Confirmed
0	1	30/01/20	6:00 PM	Kerala	0	0	1
1	2	31/01/20	6:00 PM	Kerala	0	0	1
2	3	01/02/20	6:00 PM	Kerala	0	0	2
3	4	02/02/20	6:00 PM	Kerala	0	0	3
4	5	03/02/20	6:00 PM	Kerala	0	0	3
...	...	...	...	...	...	...	...
7081	7082	07/10/20	8:00 AM	Telengana	177008	1189	204748
7082	7083	07/10/20	8:00 AM	Tripura	22623	301	27545
7083	7084	07/10/20	8:00 AM	Uttarakhand	43238	677	52329
7084	7085	07/10/20	8:00 AM	UttarPradesh	370753	6153	420937
7085	7086	07/10/20	8:00 AM	WestBengal	243743	5318	277049

7086 rows × 7 columns

```
data["StatesUniounTerritories"]= data["StatesUniounTerritories"].str.replace('Telengana', "Te
data
```

	Sno	Date	Time	StatesUniounTerritories	Cured	Deaths	Confirmed
0	1	30/01/20	6:00 PM	Kerala	0	0	1
1	2	31/01/20	6:00 PM	Kerala	0	0	1
2	3	01/02/20	6:00 PM	Kerala	0	0	2
3	4	02/02/20	6:00 PM	Kerala	0	0	3

Merging two columns with slightly different name that will cause an error

```
data["StatesUniounTerritories"]= data["StatesUniounTerritories"].str.replace('DadarNagarHavel
data
```

	Sno	Date	Time	StatesUniounTerritories	Cured	Deaths	Confirmed
0	1	30/01/20	6:00 PM	Kerala	0	0	1
1	2	31/01/20	6:00 PM	Kerala	0	0	1
2	3	01/02/20	6:00 PM	Kerala	0	0	2
3	4	02/02/20	6:00 PM	Kerala	0	0	3
4	5	03/02/20	6:00 PM	Kerala	0	0	3
...	...	...	...	...	...	...	...
7081	7082	07/10/20	8:00 AM	Telangana	177008	1189	204748
7082	7083	07/10/20	8:00 AM	Tripura	22623	301	27545
7083	7084	07/10/20	8:00 AM	Uttarakhand	43238	677	52329
7084	7085	07/10/20	8:00 AM	UttarPradesh	370753	6153	420937
7085	7086	07/10/20	8:00 AM	WestBengal	243743	5318	277049

7086 rows × 7 columns

```
data["StatesUniounTerritories"]= data["StatesUniounTerritories"].str.replace('DamanDiu', "Dad
data
```

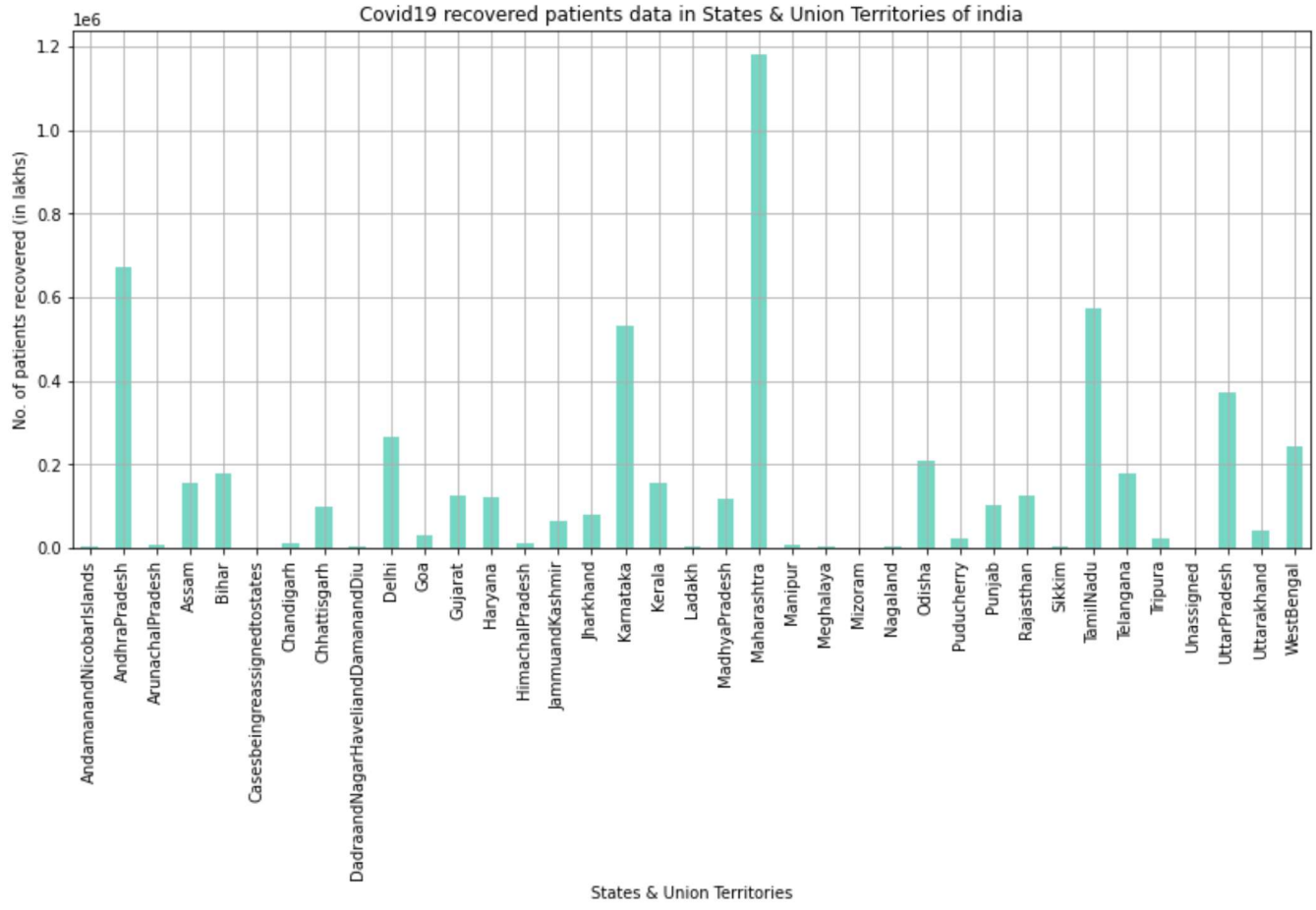
	Sno	Date	Time	StatesUniounTerritories	Cured	Deaths	Confirmed
0	1	30/01/20	6:00 PM	Kerala	0	0	1
1	2	31/01/20	6:00 PM	Kerala	0	0	1
2	3	01/02/20	6:00 PM	Kerala	0	0	2
3	4	02/02/20	6:00 PM	Kerala	0	0	3
4	5	03/02/20	6:00 PM	Kerala	0	0	3

```
cured = data.groupby('StatesUniounTerritories').Cured.max()
cured
```

```
StatesUniounTerritories
AndamanandNicobarIslands               3678
AndhraPradesh                        672479
ArunachalPradesh                       7965
Assam                                155077
Bihar                                178395
Casesbeingreassignedtostates              0
Chandigarh                            11035
Chhattisgarh                         100551
DadraandNagarHaveliandDamanandDiu      3000
Delhi                                266935
Goa                                   31050
Gujarat                              125111
Haryana                              123286
HimachalPradesh                       12918
JammuandKashmir                       65496
Jharkhand                             78089
Karnataka                            533074
Kerala                               154092
Ladakh                                 3464
MadhyaPradesh                        118039
Maharashtra                         1179726
Manipur                                9482
Meghalaya                              4606
Mizoram                                1887
Nagaland                               5460
Odisha                               210217
Puducherry                            24614
Punjab                               104355
Rajasthan                            125448
Sikkim                                 2587
TamilNadu                            575212
Telangana                            177008
Tripura                               22623
Unassigned                                0
UttarPradesh                         370753
Uttarakhand                           43238
WestBengal                           243743
Name: Cured, dtype: int64
```

```
cured.plot(kind='bar', figsize=(14, 6),color='#76D7C4')
```

```
plt.ylabel('No. of patients recovered (in lakhs)')
plt.xlabel('States & Union Territories')
plt.title('Covid19 recovered patients data in States & Union Territories of india')
plt.savefig('cured patients data in bar graph.png', bbox_inches = 'tight')
plt.grid()
plt.show()
```

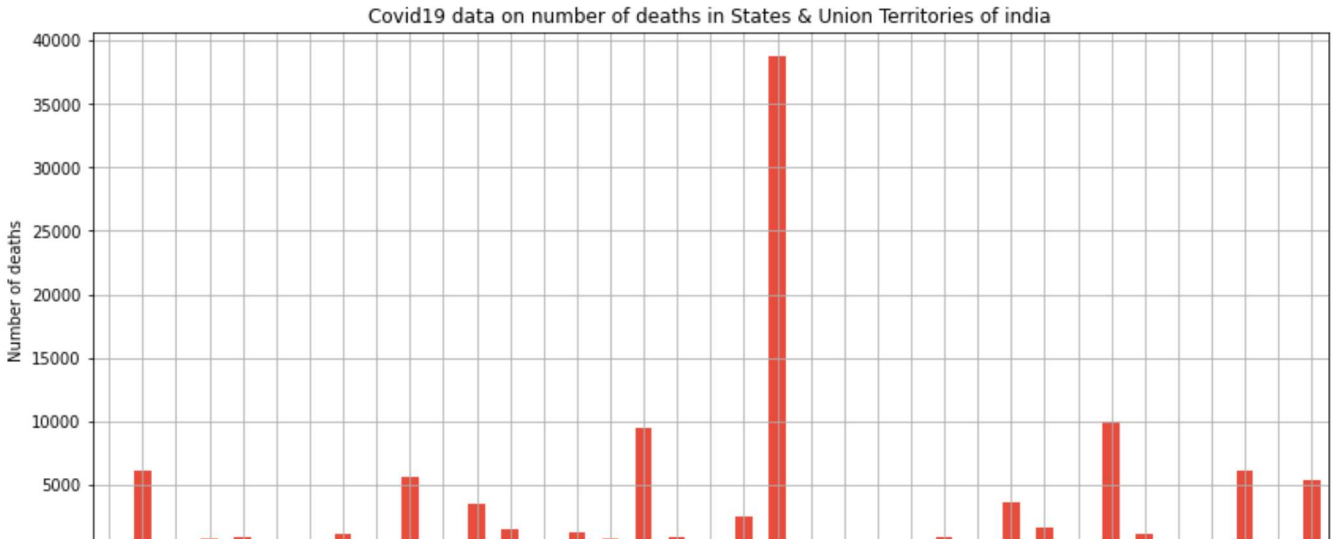



```
max_deaths = data.groupby('StatesUniounTerritories').Deaths.max()
max_deaths
```

```
StatesUniounTerritories
AndamanandNicobarIslands                54
AndhraPradesh                         6052
ArunachalPradesh                        20
Assam                                  778
Bihar                                  925
Casesbeingreassignedtostates             0
Chandigarh                             180
Chhattisgarh                          1104
DadraandNagarHaveliandDamanandDiu        2
```

```
Delhi                          5581
Goa                             468
Gujarat                        3519
Haryana                        1509
HimachalPradesh                 229
JammuandKashmir                1268
Jharkhand                       757
Karnataka                      9461
Kerala                          884
Ladakh                           61
MadhyaPradesh                  2488
Maharashtra                   38717
Manipur                          78
Meghalaya                        60
Mizoram                           0
Nagaland                         17
Odisha                          940
Puducherry                      546
Punjab                         3679
Rajasthan                      1574
Sikkim                           49
TamilNadu                      9917
Telangana                      1189
Tripura                         301
Unassigned                        0
UttarPradesh                   6153
Uttarakhand                     677
WestBengal                     5318
Name: Deaths, dtype: int64
```

```
max_deaths.plot(kind='bar', figsize=(14, 6),color='#E74C3C')
plt.ylabel('Number of deaths')
plt.xlabel('States & Union Territories')
plt.title('Covid19 data on number of deaths in States & Union Territories of india')
plt.grid()
plt.savefig('death cases data bar graph2.png', bbox_inches = 'tight')
```




```
max_cases = data.groupby('StatesUniounTerritories').Confirmed.max()
max_cases
```

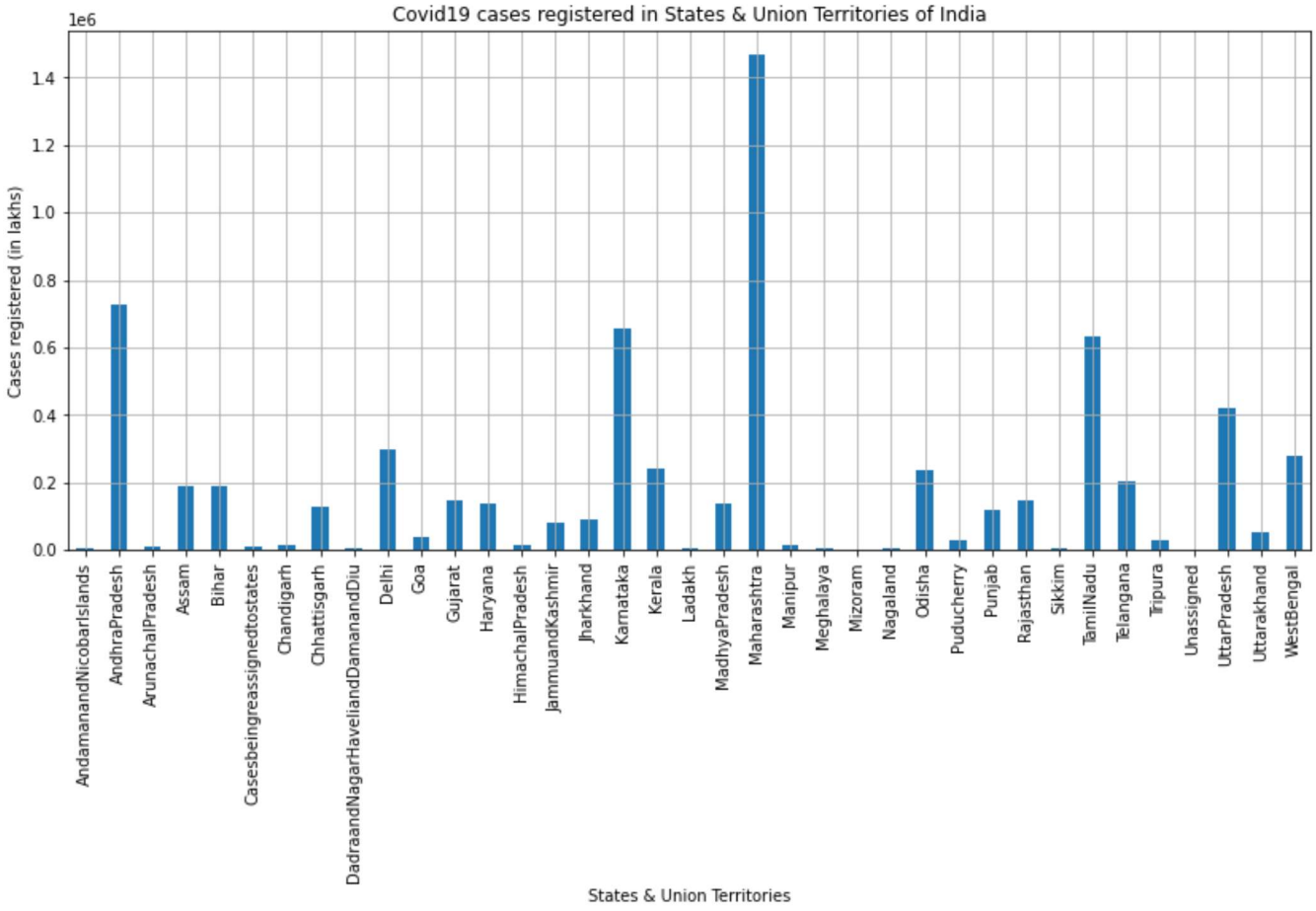
```
StatesUniounTerritories
AndamanandNicobarIslands               3912
AndhraPradesh                        729307
ArunachalPradesh                      11007
Assam                                188902
Bihar                                190740
Casesbeingreassignedtostates           9265
Chandigarh                            12707
Chhattisgarh                         128893
DadraandNagarHaveliandDamanandDiu      3103
Delhi                                295236
Goa                                   36238
Gujarat                              145200
Haryana                              136115
HimachalPradesh                       16283
JammuandKashmir                       80476
Jharkhand                             88873
Karnataka                            657705
Kerala                               242799
Ladakh                                 4720
MadhyaPradesh                        138668
Maharashtra                         1465911
Manipur                               12240
Meghalaya                              7037
Mizoram                                2148
Nagaland                               6662
Odisha                               238003
Puducherry                            29682
Punjab                               120016
Rajasthan                            148316
Sikkim                                 3216
TamilNadu                            630408
Telangana                            204748
Tripura                               27545
Unassigned                               77
UttarPradesh                         420937
Uttarakhand                           52329
```

```
WestBengal                          277049
Name: Confirmed, dtype: int64
```

```
max_cases.sum()
```

```
6766473
```

```
max_cases.plot(kind='bar', figsize=(14, 6))
plt.ylabel('Cases registered (in lakhs)')
plt.xlabel('States & Union Territories')
plt.grid()
plt.title('Covid19 cases registered in States & Union Territories of India')
plt.savefig("cases registered in states2.png", bbox_inches = 'tight')
```




```
#how many states affected
data['StatesUniounTerritories'].unique()
```

```
array(['Kerala', 'Telangana', 'Delhi', 'Rajasthan', 'UttarPradesh',
       'Haryana', 'Ladakh', 'TamilNadu', 'Karnataka', 'Maharashtra',
       'Punjab', 'JammuandKashmir', 'AndhraPradesh', 'Uttarakhand',
       'Odisha', 'Puducherry', 'WestBengal', 'Chhattisgarh', 'Chandigarh',
       'Gujarat', 'HimachalPradesh', 'MadhyaPradesh', 'Bihar', 'Manipur',
       'Mizoram', 'AndamanandNicobarIslands', 'Goa', 'Unassigned',
       'Assam', 'Jharkhand', 'ArunachalPradesh', 'Tripura', 'Nagaland',
       'Meghalaya', 'DadraandNagarHaveliandDamanandDiu',
       'Casesbeingreassignedtostates', 'Sikkim'], dtype=object)
```

```
len(data['StatesUniounTerritories'].unique())
```

```
37
```

```
data['Month'] = data['Date'].str[3:5]
data['Month'] = data['Month'].astype('int64')
data.head(5)
```

	Sno	Date	Time	StatesUniounTerritories	Cured	Deaths	Confirmed	Month
0	1	30/01/20	6:00 PM	Kerala	0	0	1	1
1	2	31/01/20	6:00 PM	Kerala	0	0	1	1
2	3	01/02/20	6:00 PM	Kerala	0	0	2	2
3	4	02/02/20	6:00 PM	Kerala	0	0	3	2
4	5	03/02/20	6:00 PM	Kerala	0	0	3	2

```
data.groupby(['Month']).sum()
```

Month	Sno	Cured	Deaths	Confirmed
1	3	0	0	2
2	493	0	0	86
3	139689	808	202	9687
4	952796	75443	13270	422442
5	2088522	1133341	89834	2938234
6	3311103	5668946	319690	10558374
7	4582965	19980130	793511	31726501
8	5695165	58580895	1553468	80749620
9	6632325	118592934	2443374	149113758
10	1706180	38556837	711884	45795597

```
results = data.groupby(['Month'])[["Confirmed", "Cured", "Deaths"]].max()
results
```

Month	Confirmed	Cured	Deaths
1	1	0	0
2	3	0	0
3	234	39	9
4	9915	1593	432
5	65168	28081	2197
6	169883	88960	7610
7	411798	248615	14729
8	780689	562401	24399
9	1366129	1069159	36181
10	1465911	1179726	38717

```
data.dtypes
```

```
Sno                          int64
Date                        object
Time                        object
StatesUniounTerritories     object
Cured                        int64
Deaths                       int64
Confirmed                    int64
Month                        int64
dtype: object
```

```
data['Time']= pd.to_datetime(data.Time)
data
```

	Sno	Date	Time	StatesUniounTerritories	Cured	Deaths	Confirmed	Mont
0	1	30/01/20	2020-11-24 18:00:00	Kerala	0	0	1	
1	2	31/01/20	2020-11-24 18:00:00	Kerala	0	0	1	
			2020-					

```
del data['Time']
data
```

	Sno	Date	StatesUniounTerritories	Cured	Deaths	Confirmed	Month
0	1	30/01/20	Kerala	0	0	1	1
1	2	31/01/20	Kerala	0	0	1	1
2	3	01/02/20	Kerala	0	0	2	2
3	4	02/02/20	Kerala	0	0	3	2
4	5	03/02/20	Kerala	0	0	3	2
...	...	...	...	...	...	...	...
7081	7082	07/10/20	Telangana	177008	1189	204748	10
7082	7083	07/10/20	Tripura	22623	301	27545	10
7083	7084	07/10/20	Uttarakhand	43238	677	52329	10
7084	7085	07/10/20	UttarPradesh	370753	6153	420937	10
7085	7086	07/10/20	WestBengal	243743	5318	277049	10

7086 rows × 7 columns

```
all_cured = cured.value_counts
all_cured
```

```
<bound method IndexOpsMixin.value_counts of StatesUniounTerritories
AndamanandNicobarIslands                3678
AndhraPradesh                         672479
ArunachalPradesh                        7965
Assam                                 155077
Bihar                                 178395
Casesbeingreassignedtostates               0
Chandigarh                             11035
Chhattisgarh                          100551
DadraandNagarHaveliandDamanandDiu       3000
Delhi                                 266935
Goa                                    31050
Gujarat                               125111
```

```
Haryana                          123286
HimachalPradesh                   12918
JammuandKashmir                   65496
Jharkhand                         78089
Karnataka                        533074
Kerala                           154092
Ladakh                             3464
MadhyaPradesh                    118039
Maharashtra                     1179726
Manipur                            9482
Meghalaya                          4606
Mizoram                            1887
Nagaland                           5460
Odisha                           210217
Puducherry                        24614
Punjab                           104355
Rajasthan                        125448
Sikkim                             2587
TamilNadu                        575212
Telangana                        177008
Tripura                           22623
Unassigned                            0
UttarPradesh                     370753
Uttarakhand                       43238
WestBengal                       243743
Name: Cured, dtype: int64>
```

```
all_cases = max_cases.value_counts
all_cases
```

```
<bound method IndexOpsMixin.value_counts of StatesUniounTerritories
AndamanandNicobarIslands               3912
AndhraPradesh                        729307
ArunachalPradesh                      11007
Assam                                188902
Bihar                                190740
Casesbeingreassignedtostates           9265
Chandigarh                            12707
Chhattisgarh                         128893
DadraandNagarHaveliandDamanandDiu      3103
Delhi                                295236
Goa                                   36238
Gujarat                              145200
Haryana                              136115
HimachalPradesh                       16283
JammuandKashmir                       80476
Jharkhand                             88873
Karnataka                            657705
Kerala                               242799
Ladakh                                 4720
MadhyaPradesh                        138668
Maharashtra                         1465911
Manipur                               12240
Meghalaya                              7037
Mizoram                                2148
Nagaland                               6662
```

```
Odisha                              238003
Puducherry                           29682
Punjab                              120016
Rajasthan                           148316
Sikkim                                3216
TamilNadu                           630408
Telangana                           204748
Tripura                              27545
Unassigned                              77
UttarPradesh                        420937
Uttarakhand                          52329
WestBengal                          277049
Name: Confirmed, dtype: int64>
```

```
all_deaths = max_deaths.value_counts()
all_deaths
```

```
0        3
5581     1
229      1
677      1
20       1
17       1
1104     1
78       1
38717    1
925      1
778      1
6153     1
5318     1
884      1
2        1
940      1
1189     1
3519     1
61       1
1268     1
60       1
9917     1
2488     1
54       1
757      1
180      1
49       1
546      1
468      1
301      1
9461     1
1574     1
1509     1
6052     1
3679     1
Name: Deaths, dtype: int64
```

```
all_states = data.StatesUniounTerritories
all_states
```

```
all_states
```

```
0             Kerala
1             Kerala
2             Kerala
3             Kerala
4             Kerala
            ...
7081       Telangana
7082         Tripura
7083     Uttarakhand
7084    UttarPradesh
7085      WestBengal
Name: StatesUniounTerritories, Length: 7086, dtype: object
```

Collecting the data only related to Telangana State

```
telangana_data = data[data.StatesUniounTerritories.str.contains('Telangana')]
telangana_data
```

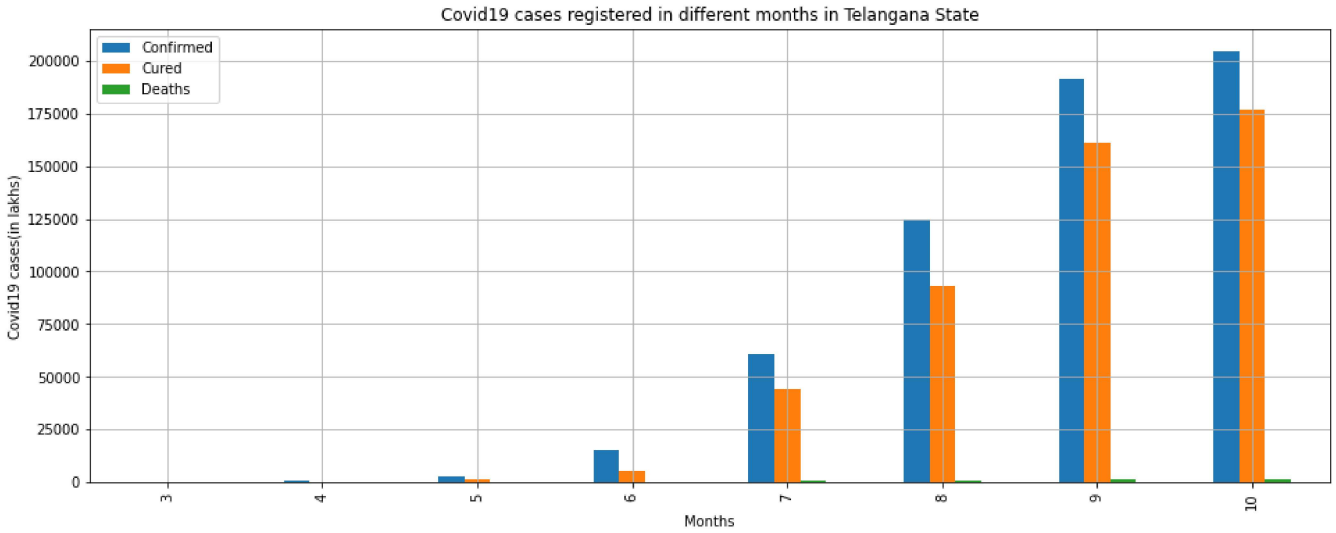
	Sno	Date	StatesUniounTerritories	Cured	Deaths	Confirmed	Month
32	33	02/03/20	Telangana	0	0	1	3
35	36	03/03/20	Telangana	0	0	1	3
43	44	04/03/20	Telangana	0	0	1	3
49	50	05/03/20	Telangana	0	0	1	3
56	57	06/03/20	Telangana	0	0	1	3
...	...	...	...	...	...	...	...
6941	6942	03/10/20	Telangana	167846	1153	197327	10
6976	6977	04/10/20	Telangana	170212	1163	199276	10
7011	7012	05/10/20	Telangana	172388	1171	200611	10
7046	7047	06/10/20	Telangana	174769	1181	202594	10
7081	7082	07/10/20	Telangana	177008	1189	204748	10

220 rows × 7 columns

```
month_telangana = telangana_data.groupby(['Month'])[['Confirmed', 'Cured', 'Deaths']].max()
month_telangana
```

Month	Confirmed	Cured	Deaths
3	79	1	1
4	1012	367	26
5	2499	1412	77
6	15394	5582	253
7	60717	44572	505

```
month_telangana.plot(kind='bar', figsize=(16, 6))
plt.grid()
plt.ylabel('Covid19 cases(in lakhs)')
plt.xlabel('Months')
plt.title('Covid19 cases registered in different months in Telangana State')
plt.savefig("cases registered in telangana.png", bbox_inches = 'tight')
```




```
telangana_data.max()
```

```
Sno                           7082
Date                       31/08/20
StatesUniounTerritories    Telangana
Cured                         177008
Deaths                          1189
Confirmed                     204748
Month                             10
dtype: object
```

Collecting the data only related to Andaman and Nicobar islands

```
AndamanandNicobarIslands_data = data[data.StatesUniounTerritories.str.contains('AndamanandNic
AndamanandNicobarIslands_data
```

	Sno	Date	StatesUniounTerritories	Cured	Deaths	Confirmed	Month
365	366	26/03/20	AndamanandNicobarIslands	0	0	1	3
392	393	27/03/20	AndamanandNicobarIslands	0	0	1	3
420	421	28/03/20	AndamanandNicobarIslands	0	0	6	3
447	448	29/03/20	AndamanandNicobarIslands	0	0	9	3
474	475	30/03/20	AndamanandNicobarIslands	0	0	9	3
...	...	...	...	...	...	...	...
6911	6912	03/10/20	AndamanandNicobarIslands	3631	53	3858	10
6946	6947	04/10/20	AndamanandNicobarIslands	3642	53	3868	10
6981	6982	05/10/20	AndamanandNicobarIslands	3649	53	3884	10
7016	7017	06/10/20	AndamanandNicobarIslands	3659	54	3899	10
7051	7052	07/10/20	AndamanandNicobarIslands	3678	54	3912	10

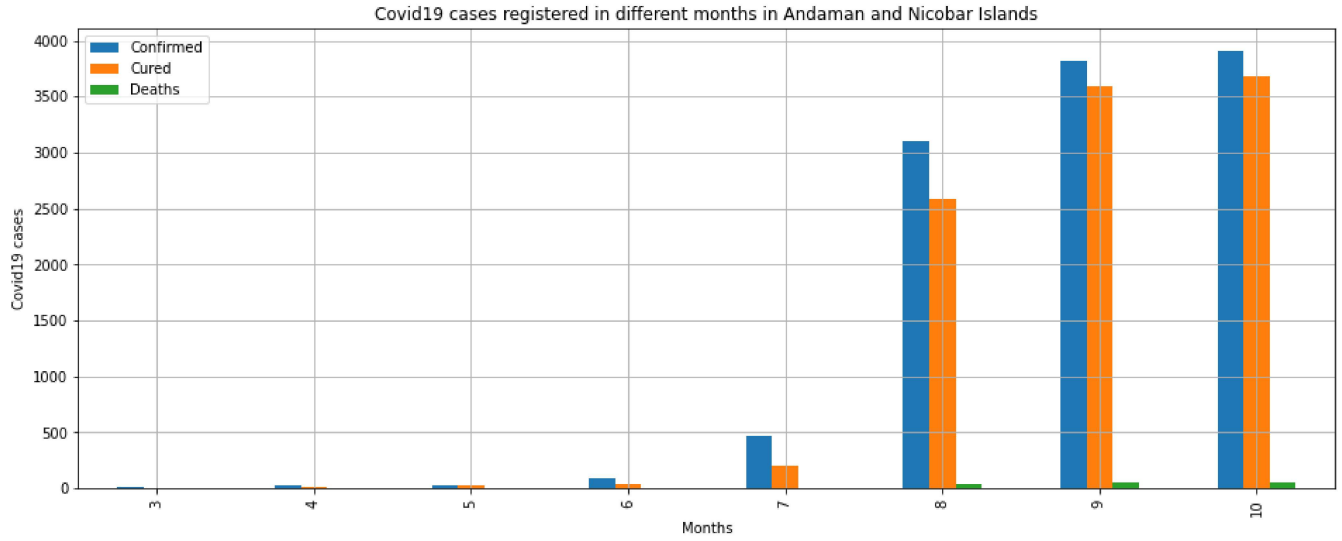
196 rows × 7 columns

```
month_AndamanandNicobarIslands = AndamanandNicobarIslands_data.groupby(['Month'])[['Confirmed
month_AndamanandNicobarIslands
```

Month	Confirmed	Cured	Deaths
3	10	0	0
4	33	15	0
5	33	33	0
6	90	46	0
7	471	201	4
8	3104	2586	45
9	3821	3587	53
10	3912	3678	54

```
month_AndamanandNicobarIslands.plot(kind='bar', figsize=(16, 6))
plt.grid()
plt.ylabel('Covid19 cases')
plt.xlabel('Months')
plt.title('Covid19 cases registered in different months in Andaman and Nicobar Islands')
```

```
plt.title('Covid19 cases registered in different months in Andaman and Nicobar Islands')
plt.savefig("cases registered in AndamanandNicobarIslands.png", bbox_inches = 'tight')
```




```
AndamanandNicobarIslands_data.max()
```

```
Sno                                    7052
Date                                31/08/20
StatesUniounTerritories    AndamanandNicobarIslands
Cured                                  3678
Deaths                                   54
Confirmed                              3912
Month                                    10
dtype: object
```

Capturing data of only Andhra Pradesh

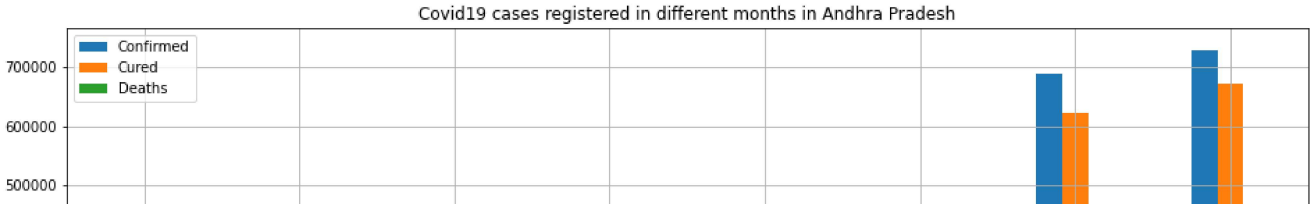
```
AndhraPradesh_data = data[data.StatesUniounTerritories.str.contains('AndhraPradesh')]
AndhraPradesh_data
```

	Sno	Date	StatesUniounTerritories	Cured	Deaths	Confirmed	Month
121	122	12/03/20	AndhraPradesh	0	0	1	3
134	135	13/03/20	AndhraPradesh	0	0	1	3
147	148	14/03/20	AndhraPradesh	0	0	1	3
148	149	15/03/20	AndhraPradesh	0	0	1	3
162	163	16/03/20	AndhraPradesh	0	0	1	3

```
month_AndhraPradesh = AndhraPradesh_data.groupby(['Month'])[['Confirmed', 'Cured', 'Deaths']]
month_AndhraPradesh
```

Month	Confirmed	Cured	Deaths
3	40	1	0
4	1403	321	31
5	3569	2289	60
6	13891	6232	180
7	130557	60024	1281
8	424767	321754	3884
9	687351	622136	5780
10	729307	672479	6052

```
month_AndhraPradesh.plot(kind='bar', figsize=(16, 6))
plt.grid()
plt.ylabel('Covid19 cases')
plt.xlabel('Months')
plt.title('Covid19 cases registered in different months in Andhra Pradesh')
plt.savefig("cases registered in AndhraPradesh.png", bbox_inches = 'tight')
```




```
AndhraPradesh_data.max()
```

```
Sno                                7053
Date                           31/08/20
StatesUniounTerritories    AndhraPradesh
Cured                            672479
Deaths                             6052
Confirmed                        729307
Month                                10
dtype: object
```

maximum cases recorded in Indian states

```
results = data.groupby(['Month'])[["Confirmed", "Cured", "Deaths"]].max()
results
```

Month	Confirmed	Cured	Deaths
1	1	0	0
2	3	0	0
3	234	39	9
4	9915	1593	432
5	65168	28081	2197
6	169883	88960	7610
7	411798	248615	14729
8	780689	562401	24399
9	1366129	1069159	36181
10	1465911	1179726	38717

Check for the null values sum in the dataset

```
data.isnull().sum()
```

```
Sno                        0
Date                       0
```

```
StatesUniounTerritories    0
Cured                      0
Deaths                     0
Confirmed                  0
Month                      0
dtype: int64
```

```
data.columns
```

```
Index(['Sno', 'Date', 'StatesUniounTerritories', 'Cured', 'Deaths',
       'Confirmed', 'Month'],
      dtype='object')
```

```
data
```

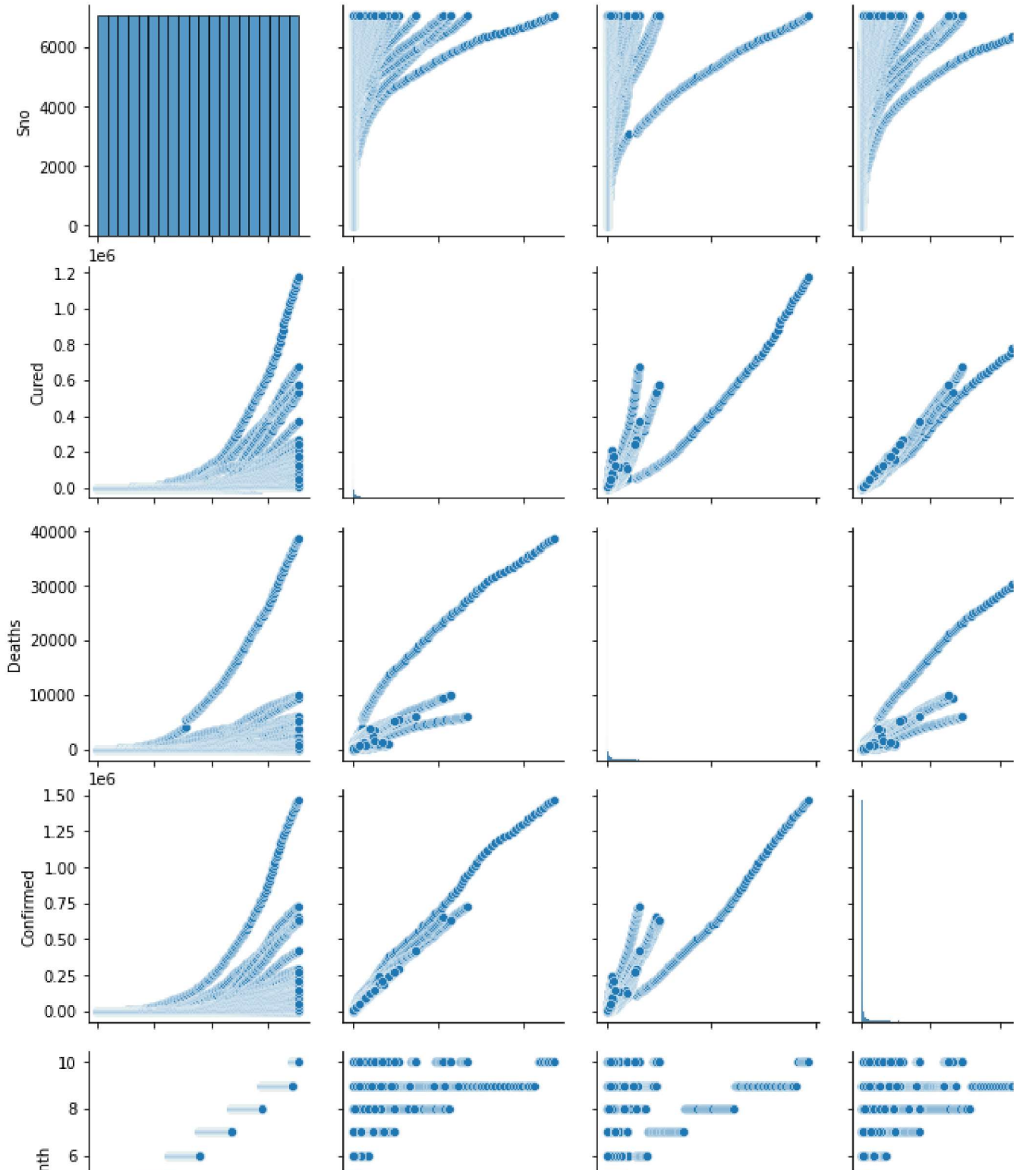
	Sno	Date	StatesUniounTerritories	Cured	Deaths	Confirmed	Month
0	1	30/01/20	Kerala	0	0	1	1
1	2	31/01/20	Kerala	0	0	1	1
2	3	01/02/20	Kerala	0	0	2	2
3	4	02/02/20	Kerala	0	0	3	2
4	5	03/02/20	Kerala	0	0	3	2
...	...	...	...	...	...	...	...
7081	7082	07/10/20	Telangana	177008	1189	204748	10
7082	7083	07/10/20	Tripura	22623	301	27545	10
7083	7084	07/10/20	Uttarakhand	43238	677	52329	10
7084	7085	07/10/20	UttarPradesh	370753	6153	420937	10
7085	7086	07/10/20	WestBengal	243743	5318	277049	10

7086 rows × 7 columns

Pairplot for the dataset using Seaborn library

```
import seaborn as sns
sns.pairplot(data)
```

```
<seaborn.axisgrid.PairGrid at 0x7f91170802b0>
```



State and map the unique values in the column StatesUniounTerritories

```
data.StatesUniounTerritories.unique()
```

```
array(['Kerala', 'Telangana', 'Delhi', 'Rajasthan', 'UttarPradesh',
       'Haryana', 'Ladakh', 'TamilNadu', 'Karnataka', 'Maharashtra',
       'Punjab', 'JammuandKashmir', 'AndhraPradesh', 'Uttarakhand',
       'Odisha', 'Puducherry', 'WestBengal', 'Chhattisgarh', 'Chandigarh',
```

```
       'Gujarat', 'HimachalPradesh', 'MadhyaPradesh', 'Bihar', 'Manipur',
       'Mizoram', 'AndamanandNicobarIslands', 'Goa', 'Unassigned',
       'Assam', 'Jharkhand', 'ArunachalPradesh', 'Tripura', 'Nagaland',
       'Meghalaya', 'DadraandNagarHaveliandDamanandDiu',
       'Casesbeingreassignedtostates', 'Sikkim'], dtype=object)
```

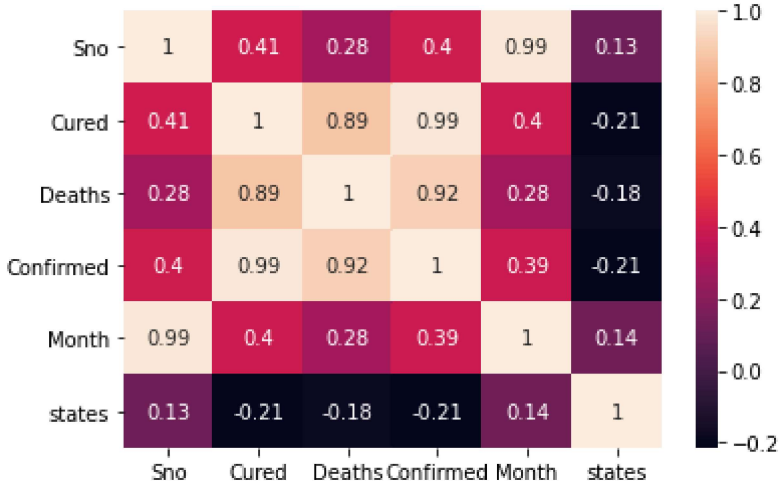
```
Value_Mapping = {'Kerala' : 0, 'Telangana' :1,'Delhi':3,'Rajasthan':4,'UttarPradesh':5,'Harya
                 'AndhraPradesh':13,'Uttarakhand':14,'Odisha':15,'Puducherry':16,'WestBengal
                 'Manipur':24,'Mizoram':25,'AndamanandNicobarIslands':26,'Goa':27,'Unassigne
                 'Casesbeingreassignedtostates':36,'Sikkim':37}
data['states'] = data['StatesUniounTerritories'].map(Value_Mapping)
data.head(5)
```

	Sno	Date	StatesUniounTerritories	Cured	Deaths	Confirmed	Month	states
0	1	30/01/20	Kerala	0	0	1	1	0
1	2	31/01/20	Kerala	0	0	1	1	0
2	3	01/02/20	Kerala	0	0	2	2	0
3	4	02/02/20	Kerala	0	0	3	2	0
4	5	03/02/20	Kerala	0	0	3	2	0

Heatmap reprasenting the co-relation in the dataset

```
correlation = data.corr()
sns.heatmap(correlation,xticklabels = correlation.columns,yticklabels = correlation.columns,a
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9106d90e10>
```



Drop the unwanted columns

```
df = data.drop(['Date','StatesUniounTerritories','Sno'],axis =1)
```

Separate the Features and Labels

```
y = df["states"]
X = df.drop("states",axis=1)
```

Split the dataset into Train and Test values

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,random_state = 42)
```

Preprocessing data using Standard Scaler technique

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

Ensemble model trained with Random Forest Regressor

```
from sklearn.ensemble import RandomForestRegressor
forest=RandomForestRegressor()
forest.fit(X_train,y_train)
```

```
RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                      max_depth=None, max_features='auto', max_leaf_nodes=None,
                      max_samples=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=1,
                      min_samples_split=2, min_weight_fraction_leaf=0.0,
                      n_estimators=100, n_jobs=None, oob_score=False,
                      random_state=None, verbose=0, warm_start=False)
```

```
forest.score(X_test,y_test)
```

```
0.8642687254725648
```

Tree model trained with Decision tree Regressor

```
from sklearn.tree import DecisionTreeRegressor
regressor = DecisionTreeRegressor()
regressor.fit(X_train, y_train)
```

```
DecisionTreeRegressor(ccp_alpha=0.0, criterion='mse', max_depth=None,
                      max_features=None, max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
```

```
                            min_samples_leaf=1, min_samples_split=2,
                            min_weight_fraction_leaf=0.0, presort='deprecated',
                            random_state=None, splitter='best')
```

```
regressor.score(X_test,y_test)
```

```
     0.7700194251543991
```

```
from sklearn.ensemble import VotingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

estimator = []
estimator.append(('LR',
                  LogisticRegression(solver ='lbfgs',
                                     multi_class ='multinomial',
                                     max_iter = 200)))
estimator.append(('SVC', SVC(gamma ='auto', probability = True)))
estimator.append(('DTC', DecisionTreeClassifier()))

vot_hard = VotingClassifier(estimators = estimator, voting ='hard')
vot_hard.fit(X_train, y_train)
y_pred = vot_hard.predict(X_test)

score = accuracy_score(y_test, y_pred)
print("Hard Voting Score % d" % score)

vot_soft = VotingClassifier(estimators = estimator, voting ='soft')
vot_soft.fit(X_train, y_train)
y_pred = vot_soft.predict(X_test)

score = accuracy_score(y_test, y_pred)
print("Soft Voting Score % d" % score)
```

```
     Hard Voting Score  0
     Soft Voting Score  0
```

XG Boost model trained with XGBoost Classifier

```
import xgboost as xgb
my_model = xgb.XGBClassifier()
my_model.fit(X_train, y_train)
```

```
     XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
                   colsample_bynode=1, colsample_bytree=1, gamma=0,
                   learning_rate=0.1, max_delta_step=0, max_depth=3,
```

```
                    min_child_weight=1, missing=None, n_estimators=100, n_jobs=1,
                    nthread=None, objective='multi:softprob', random_state=0,
                    reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
                    silent=None, subsample=1, verbosity=1)
```

```
my_model.score(X_test,y_test)
```

```
0.5609480812641083
```

Confusion Matrix

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
cm
```

```
array([[ 0,  0,  0, ...,  2,  0,  0],
       [ 0,  6,  5, ...,  2,  0,  0],
       [ 0,  0, 13, ...,  6,  0,  0],
       ...,
       [ 0,  0,  0, ..., 18,  8,  0],
       [ 0,  0,  0, ...,  0,  9,  0],
       [ 0,  0,  0, ..., 11,  7,  4]])
```

Naive Bayes model trained with Gaussian Naive Bayes algorithm

```
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
gnb.fit(X_train, y_train)

# making predictions on the testing set
y_pred = gnb.predict(X_test)
from sklearn import metrics
print("Gaussian Naive Bayes model accuracy(in %):", metrics.accuracy_score(y_test, y_pred)*10
```

```
Gaussian Naive Bayes model accuracy(in %): 10.66591422121896
```

model trained with support vector classifier

```
from sklearn.svm import SVC
clf = SVC(kernel='linear')
clf.fit(X_train, y_train)
clf.score(X_test,y_test)
```

```
0.17494356659142213
```

Check for Accuracy Score using metrics

```
from sklearn.metrics import accuracy_score
score_1 = accuracy_score(y_test,y_pred)
score_1
```

```
0.10665914221218961
```

Tree model trained with Decision Tree Classifier

```
from sklearn.tree import DecisionTreeClassifier
DTC = DecisionTreeClassifier()
DTC.fit(X_train, y_train)
```

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
```

```
DTC.score(X_test,y_test)
```

```
0.7420993227990971
```

Neighbors model trained with K Nearest Neighbour Classifier

```
from sklearn.neighbors import KNeighborsClassifier
KNNC = KNeighborsClassifier()
KNNC.fit(X_train, y_train)
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                     weights='uniform')
```

```
KNNC.score(X_test,y_test)
```

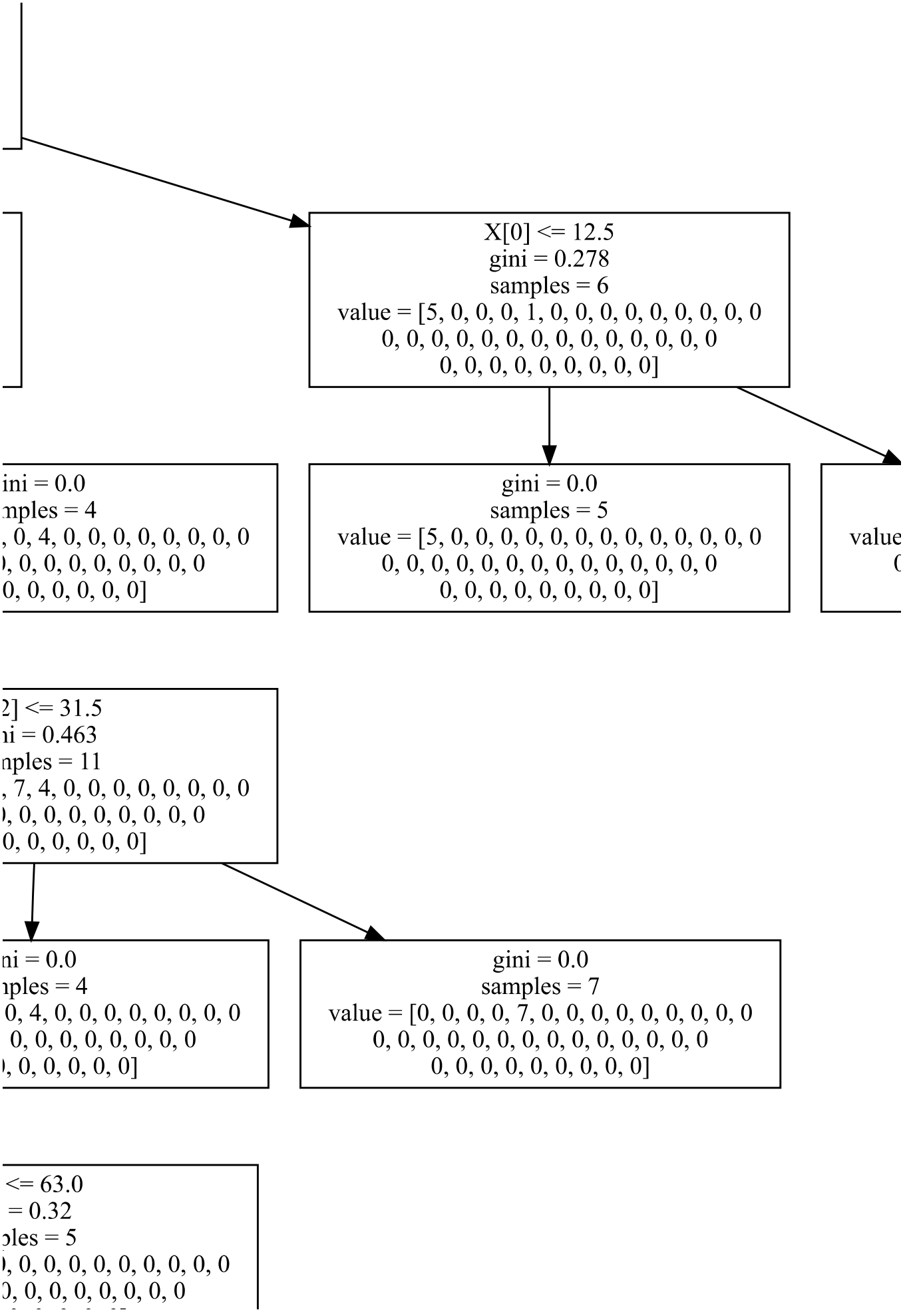
```
0.7268623024830699
```

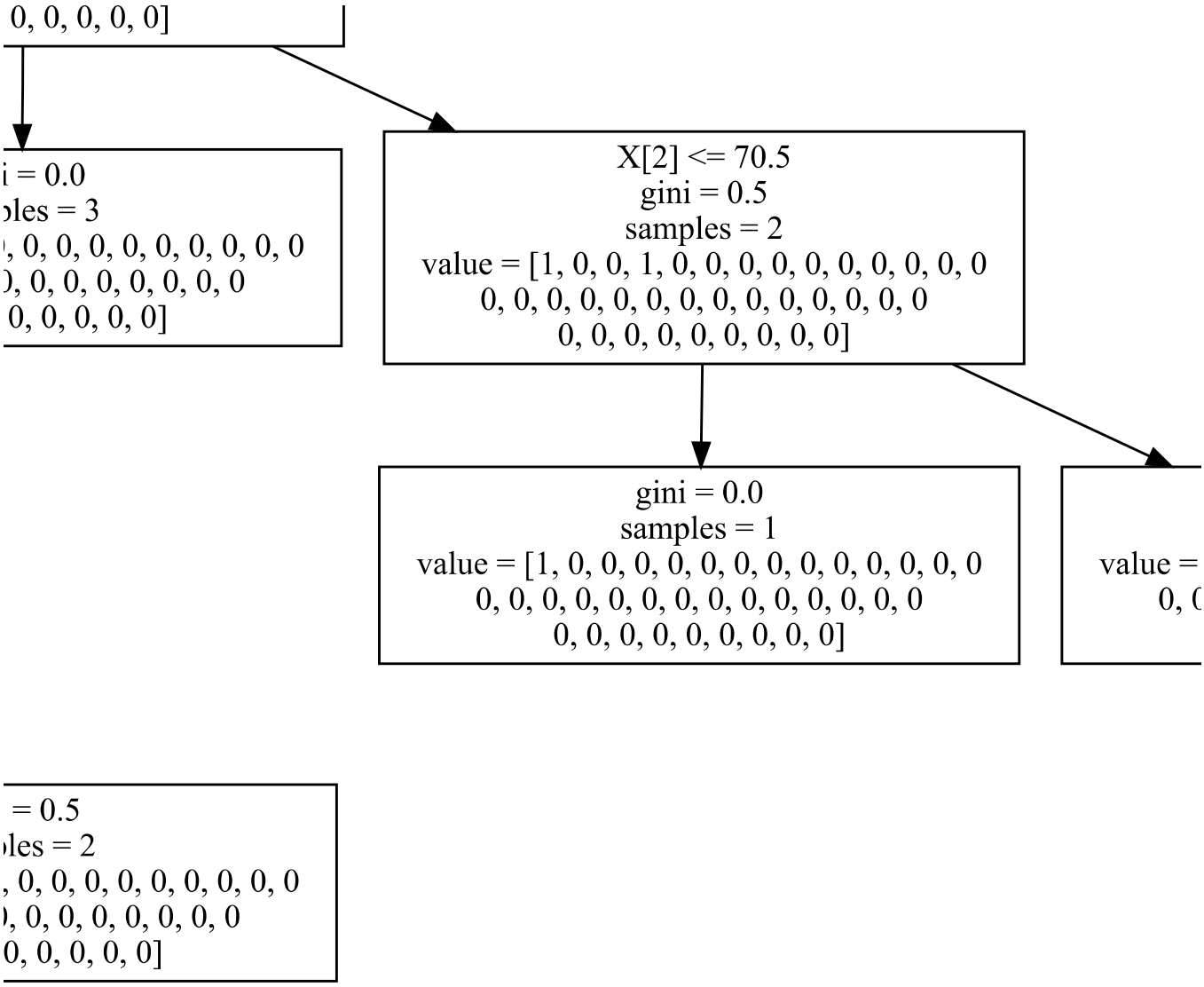
```
Reprasentation of Decision Tree using graphviz
```

```
import graphviz
dot_data = tree.export_graphviz(clf, out_file=None)
graph = graphviz.Source(dot_data)
graph.render("states")
graphviz
```

```
<module 'graphviz' from '/usr/local/lib/python3.6/dist-packages/graphviz/__init__.py'>
```

```
graph
```


X[0] <= 12.5
gini = 0.278
samples = 6
value = [5, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0]
gini = 0.0
samples = 5
value = [5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0]
gini = 0.463
samples = 11
value = [0, 7, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0]
gini = 0.0
samples = 7
value = [0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0]



Ensemble model trained with Random Forest Classifier

```
from sklearn.ensemble import RandomForestClassifier

RF = RandomForestClassifier(n_estimators=100, max_depth=2, random_state=0)
RF.fit(X, y)
RF.predict(X.iloc[460:,:])
round(RF.score(X,y), 4)
```

```
0.105
```

Neural Network model trained with MLP Classifier

```
from sklearn.neural_network import MLPClassifier

NN = MLPClassifier()
NN.fit(X_train, y_train)
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/neural_network/_multilayer_perceptron.py
  % self.max_iter, ConvergenceWarning)
MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
              beta_2=0.999, early_stopping=False, epsilon=1e-08,
              hidden_layer_sizes=(100,), learning_rate='constant',
              learning_rate_init=0.001, max_fun=15000, max_iter=200,
              momentum=0.9, n_iter_no_change=10, nesterovs_momentum=True,
              power_t=0.5, random_state=None, shuffle=True, solver='adam',
              tol=0.0001, validation_fraction=0.1, verbose=False,
              warm_start=False)
```

```
NN.score(X,y,sample_weight=None)
```

```
0.030200395145357044
```

Cluster model trained with KMeans

```
from sklearn.cluster import KMeans
k_means = KMeans()
k_means.fit(X_train,y_train)
```

```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
       n_clusters=8, n_init=10, n_jobs=None, precompute_distances='auto',
       random_state=None, tol=0.0001, verbose=0)
```

```
k_means.score(X,y)
```

```
-205721117784624.34
```

Ensemble model trained with Gradient Boosting Classifier

```
from sklearn.ensemble import GradientBoostingClassifier
GBC = GradientBoostingClassifier()
GBC.fit(X,y)
```

```
GradientBoostingClassifier(ccp_alpha=0.0, criterion='friedman_mse', init=None,
                           learning_rate=0.1, loss='deviance', max_depth=3,
                           max_features=None, max_leaf_nodes=None,
```

```
                                min_impurity_decrease=0.0, min_impurity_split=None,
                                min_samples_leaf=1, min_samples_split=2,
                                min_weight_fraction_leaf=0.0, n_estimators=100,
                                n_iter_no_change=None, presort='deprecated',
                                random_state=None, subsample=1.0, tol=0.0001,
                                validation_fraction=0.1, verbose=0,
                                warm_start=False)
```

```
GBC.score(X,y)
```

```
0.8842788597233983
```

Model trained with decomposition Factor analysis

```
from sklearn import decomposition
afa = decomposition.FactorAnalysis()
afa.fit(X,y)
```

```
FactorAnalysis(copy=True, iterated_power=3, max_iter=1000, n_components=None,
               noise_variance_init=None, random_state=0,
               svd_method='randomized', tol=0.01)
```

```
afa.score(X,y)
```

```
-34.18866520674265
```

Linear model trained with Logistic Regression

```
from sklearn.linear_model import LogisticRegression
LR = LogisticRegression()
LR.fit(X,y)
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/linear_model/_logistic.py:940: Convergen
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

```
LR.score(X,y)
```

```
0.056872706745695736
```

Linearn model training with L1 prior as Regularizer

```
from sklearn.linear_model import Lasso
L = Lasso()
L.fit(X,y)
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/linear_model/_coordinate_descent.py:476:
  positive)
Lasso(alpha=1.0, copy_X=True, fit_intercept=True, max_iter=1000,
      normalize=False, positive=False, precompute=False, random_state=None,
      selection='cyclic', tol=0.0001, warm_start=False)
```

```
L.score(X,y)
```

```
0.12116148052609176
```