VARUVAN VADIVELAN INSTITUTE OF TECHNOLOGY

NAAN MUDHALVAN : IBM

PHASE – 5

TECHNOLOGY : DATA ANALYTICS

PROJECT TITLE : COVID 19 CASES ANALYSIS

PROBLEM STATEMENT :

To design a simple mobile webpage which would serve as COVID-19 help portal for the general public (Indian citizens) to help them track total & nearby COVID-19 cases, know about preventive measures and symptoms, gain access to general information on COVID-19 and other emergency services.

DESIGN PROCESS :

Information Gathering & Analysis → User Research → Information Architecture → Paper wireframes → High-fidelity wireframes.

INFORMATION GATHERING & ANALYSIS :

As per data available on the internet, below insights were gathered:

1. Total number of COVID-19 cases in India = 10.1M

2. Cases of fear & anxiety were reportedly increased in this pandemic as what could a new disease cause can be overwhelming and lead to strong emotions in people. Public health actions, such as social distancing, can make people feel isolated and lonely and can increase stress and anxiety.

3. Most frequently asked queries on COVID-19 are related to:

- symptoms
- preventions
- covid-19 hospitals
- news & updates
- vaccines

4. Total no. of languages spoken in India= 21.

5. % of English speaking citizens in India = 10%.

DATA PREPROCESSING :

✓ Some factors reduce the classification performance in artificial intelligence-based diagnostic systems realized by using biomedical datasets. For this reason, various data preprocessing techniques suitable for datasets are used to increase the classification performance.

✓ A data analysis directly depends on both the data preprocessing step and the techniques chosen for this purpose. 24 While the importance of data preprocessing is so evident, it is very important to find the most suitable data preprocessing techniques for the study to be carried out.

- ✓ In this study carried out in this direction, certain data preprocessing techniques on blood tests of infected and noninfected individuals were analyzed and the effect of these techniques on the diagnosis of COVID-19 was examined.

In this study, the dataset was applied to encode categorical values with one-hot encoding, min-max feature scaling in the range of [0–1], filling the missing data using KNN and MICE methods and data balancing with SMOTE method.

SOME STEMS IS THERE IN PREPROCESSING :

- Getting the dataset .
- Importing libraries .
- Importing datasets .
- Finding Missing Data .
- Encoding Categorical Data.
- Splitting dataset into training and test set
- Feature scaling

GET THE DATASET :

To create a Jupyter model , the first

thing required is a dataset as a jupyter model completely works on data. The Collected data for a particular problem in a
proper format is Known as the Dataset.

Data Preprocessing

```
import numpy as np
import pandas as pd#

# Data Analysis

import plotly.express as px
import missingno as msno

# Feature Selection

import scipy.stats as stats
from scipy.stats import chi2_contingency

# Data Modeling

from sklearn.model_selection import train_test_split
from imblearn.under_sampling import RandomUnderSampler
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier

# Model Evaluation & saving the model
```

```
from sklearn.metrics import classification_report, confusion_matrix, ConfusionMatrixDisplay, recall_score, accuracy_score, precision_score, f1_score

import pickle
```

Reading The Data

```
# Loading the Data


data = pd.read_csv("../input/covid19-dataset-for-year-2020/covid_data_2020-2021.csv")

data.head()
```

```
test_date	cough	fever	sore_throat	shortness_of_breath	head_ache	corona_result	age_60_and_above	gender	test_indication
```

0	2021-10-11	0	0	0	0	0	Negative	Yes	female	Other
1	2021-10-11	0	0	0	0	0	Negative	Yes	male	Other
2	2021-10-11	0	0	0	0	0	Negative	No	female	Other
3	2021-10-11	0	0	0	0	0	Negative	Yes	female	Other
4	2021-10-11	0	0	0	0	0	Negative	Yes	female	Other

```
data.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 5861480 entries, 0 to 5861479

Data columns (total 10 columns):

 #   Column              Dtype

---  ------              -----

 0   test_date           object

 1   cough               int64

 2   fever               int64

 3   sore_throat         int64
 4   shortness_of_breath  int64
 5   head_ache            int64
 6   corona_result        object
 7   age_60_and_above     object
 8   gender               object
 9   test_indication      object
dtypes: int64(5), object(5)
memory usage: 447.2+ MB
Dataset has 5861480 records and 10 features.
```

This is a Binary Classification Problem.

```
# Checking the levels for categorical features

def show(data):
  for i in data.columns[1:]:
    print("Feature: {} with {} Levels".format(i,data[i].unique()))

show(data)
Feature: cough with [0 1] Levels
Feature: fever with [0 1] Levels
Feature: sore_throat with [0 1] Levels
Feature: shortness   of  breath   with [0 1] Levels
Feature: head ache with [0 1] Levels
Feature: corona result with ['Negative' 'Positive'] Levels
Feature: age 60 and above with ['Yes' 'No'] Levels
Feature: gender with ['female' 'male'] Levels
Feature: test indication with ['Other' 'Contact with confirmed' 'Abroad']
Levels
```

Target Feature is Corona result.

Data is completely Categorical except test date feature.

OUTPUT :

```
       Classification Report for Train Data

          precision   recall  f1-score   support

        0      1.00      0.96      0.98    188938
        1      0.94      1.00      0.97    113501

   accuracy                        0.98    302439
  macro avg      0.97      0.98      0.98    302439
weighted avg      0.98      0.98      0.98    302439

--------------------------------------------------------------------
Recall on Train Data:  0.9994
Specificity on Train Data:  0.9641
Accuracy on Train Data:  0.9773
Precision on Train Data:  0.9435
F1 Score on Train Data:  0.9707
--------------------------------------------------------------------

Classification Report for Test Data

          precision   recall  f1-score   support

        0      1.00      0.96      0.98     81097
        1      0.94      1.00      0.97     48520
```

```
    accuracy                           0.98    129617
   macro avg       0.97      0.98      0.98    129617
weighted avg       0.98      0.98      0.98    129617
```

IMPORTING LIBRARIES :

In orders to perform data preprocessing using python, we need to import some predefined python libraries . These libraries are used to perform some specific jobs. These are three specific libraries that we will use for data preprocessing.

NUMPY :

Numpy python library is usesd for including any type of mathematical operation in the code. It is the fundamental package for scientific calculation in python. It also supports to add large, multidimensional arrays and matrices. So, in python, we can import it as:

```
import numpy as np
```

Here we have used nm, which is a short name for Numpy, and it will be used in the whole program

Matplotlib:

The second library is matplotlib, which is a python 2D plotting library, and with this library, we need to import a sub- library Pyplot. This library is used to plot any type of chats in python for the code .

```
Import matplotlib pyplot as mtp
```

Pandas :

The last library is the Pandas library , which is one the most famous Python libraries and used for importing and managing and the Dataset .

It's an open-sources data manipulation and analysics library .

It will be imported as below

```
Import pandas as pd
```

Handling Missing data :

The next step of data preprocessing is to Handle missing data in the datasets . If our dataset contains Some missing data, then it may create a huge problem for our jupyter model . Hence it's necessary to handle missing values present in the dataset .

Ways to handle missing data :

There are mainly two ways to handle missing data ,which are:

By deleting the particular row :

The first way is used to commonly deal with null values in the ways , we just delete the specific row or column which consist null values . But this way is not so efficient and removing data may lead to loss of information which will not give accurate output .

By calculating the mean :

In this way,we will calculate the mean of that column or row which contains any missing value and will put it on the placeof missing value.This strategy is useful for the features which have numeric data such as age,salary,year,etc. Here we will use this approach.

To handle missing values,we will use Scikit-let :

Encoding Categorial data :

Categorial data is which has some categories sucn as , in our dataset;

There are two categorical varible, Country and Purchased .

Since Jupyter model completely works on mathematics and numbers,but if our dataset would have a categorical varible,then it may create trouble while building the model.So it is necessary to encode these categorical variables into numbers.

For Country variable :

Firstly,we will convert the country variables into categorical data. So to do this,we will use **LabelEncoder()**class from **preprocessing** library **learn** library in our code,which contains various laibraries for building Jupyter models. Here we will use **Imputer** class of **sklearn.preprocessing** library.

```
# categorical data
 # for Country Variable
 From sklearn.preprocessing import
 LabelEncoder
 Label_encoder_x= LabelEncoder()
 X[:,0]= label_encoder_x.fit_transform(x[:,0])
```

Splitting the Dataset into the Tranining set and Test set :

In Jupyter model data preprocessing we divide our dataset into a training set and test set. This is one of the crucial steps of data preprocessing as by doing this, we can enhance the performance of our Jupyter model.

Suppose , if we have given training to our Jupyter model by a data set and we test it by a completely different data set. Then , it will create difficult for our model to understand the correlations between the models.

If we train our model vary well and its training accuracy is also very high , but we provide a new data set to it , then it will decrease the performance.

So we always try to may a Jupyter model which performances well with the training set and also with the dataset . Here, we can define these dataset as :

Training set:

A subset of dataset to train the Jupter model , and we already know the output.

Test set :

A subset of dataset to test the Jupter model, and by using the test set, model predicts the output.

For splitting the dataset, we will use the below lines of the code.

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,
test_size=0,.2,rabdom_state=0)
```

Feature Scaling:

Feature scaling is the final step of data preprocessing in jupyter . It is a technique to standardize the independent variables of the dataset in a specific range . In feature scaling , we put our variables in the same range and in the same scale so that no any variable dominate the other variable .

```
import pandas as pd
import matplotlib.pyplot as plt
import plotly.express as px
import numpy as np
import plotly
import plotly.graph_objects as go from plotly.subplots import
make_subplots
```

Gathering Data :

For a clean and perfect data analysis, the foremost important element is collecting quality Data. For this analysis, I’ve collected many data from various sources for better accuracy.

Workflow :

- Import libraries
- Load dataset
- Look for the missing values
- Perform Data visualization

Import all libraries

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

Loading data :

```
 data = pd.read_csv
data
```

output :

	State/UTS	Total cases	Active	Discharged	Deaths	Active Ratio	Discharged Ratio	Death Ratio	Population
0	Andaman and Nicobar	10747	0	10618	129	0.00	98.80	1.20	100896618
1	Andhra Pradesh	2339078	7	2324338	14733	0.00	99.37	0.63	128500364
2	Arunachal Pradesh	66891	0	66595	296	0.00	99.56	0.44	658019
3	Assam	746100	0	738065	8035	0.00	98.92	1.08	290492
4	Bihar	851404	1	839100	12303	0.00	98.55	1.45	40100376
5	Chandigarh	99358	3	98174	1181	0.00	98.81	1.19	30501026
6	Chhattisgarh	1177768	8	1163614	14146	0.00	98.80	1.20	28900667
7	Dadra and Nagar Haveli and Daman and Diu	11591	0	11587	4	0.00	99.97	0.03	231502578
8	Delhi	2007313	10	1980781	26522	0.00	98.68	1.32	773997
9	Goa	259110	15	255082	4013	0.01	98.45	1.55	3772103

10	Gujarat	1277615	11	1266561	11043	0.00	99.13	0.86	70400153
11	Haryana	1056655	38	1045903	10714	0.00	98.98	1.01	7503010
12	Himachal Pradesh	312692	14	308465	4213	0.00	98.65	1.35	3436948
13	Jammu and Kashmir	479444	10	474649	4785	0.00	99.00	1.00	66001
14	Jharkhand	442574	0	437243	5331	0.00	98.80	1.20	124904071
15	Karnataka	4072536	123	4032105	40308	0.00	99.01	0.99	1711947
16	Kerala	6829249	1300	6756379	71570	0.02	98.93	1.05	91702478
17	Ladakh	29417	1	29185	231	0.00	99.21	0.79	4184959
18	Lakshadweep	11415	0	11363	52	0.00	99.54	0.46	11700099
19	Madhya Pradesh	1054934	2	1044155	10777	0.00	98.98	1.02	14999397
20	Maharashtra	8136945	134	7988392	148419	0.00	98.17	1.82	399001
21	Manipur	139924	0	137775	2149	0.00	98.46	1.54	47099270

22	Meghalaya	96786	1	95161	1624	0.00	98.32	1.68	79502477
23	Mizoram	238964	0	238238	726	0.00	99.70	0.30	1308967
24	Nagaland	35986	0	35204	782	0.00	97.83	2.17	38157311
25	Odisha	1336595	84	1327306	9205	0.01	99.31	0.69	19301096
26	Puducherry	175636	73	173588	1975	0.04	98.83	1.12	2073074
27	Punjab	784282	29	764964	19289	0.00	97.54	2.46	34698876
28	Rajasthan	1315564	5	1305906	9653	0.00	99.27	0.73	1521992
29	Sikkim	44321	2	43820	499	0.00	98.87	1.13	83697770
30	Tamil Nadu	3594573	58	3556466	38049	0.00	98.94	1.06	35998752
31	Telengana	841453	27	837315	4111	0.00	99.51	0.49	69599762
32	Tripura	108034	0	107094	940	0.00	99.13	0.87	1646050
33	Uttar Pradesh	2128154	18	2104502	23634	0.00	98.89	1.11	1158040

34	Uttarakhand	449429	11	441665	7753	0.00	98.27	1.73	85002417
35	West Bengal	2118696	50	2097114	21532	0.00	98.98	1.02	32199722

Shape of Data :

```
data.shape # for show number of rows and columns
```

output :

```
(36, 9)
```

Top 5 rows :

```
linkcode
data.head() # top 5 rows
```

Output :

	State/UTS	Total Cases	Active	Discharged	Death	Active Ratio	Discharged Ratio	Death Ratio	Population
31	Telengana	841453	27	837315	4111	0.0	99.51	0.49	69599762
32	Tripura	108034	0	107094	940	0.0	99.13	0.87	1646050

33	Uttar Pradesh	2128154	18	2104502	23634	0.0	98.89	1.11	1158040
34	Uttarakhand	449429	11	441665	7753	0.0	98.27	1.73	85002417
35	West Bengal	2118696	50	2097114	21532	0.0	98.98	1.02	32199722

```
data.tail() # lasr 5 rows
```

Output :

	State/UTS	Total Cases	Active	Discharged	Death	Active Ratio	Discharged Ratio	Death Ratio	Population
31	Telengana	841453	27	837315	4111	0.0	99.51	0.49	69599762
32	Tripura	108034	0	107094	940	0.0	99.13	0.87	1646050
33	Uttar Pradesh	2128154	18	2104502	23634	0.0	98.89	1.11	1158040
34	Uttarakhand	449429	11	441665	7753	0.0	98.27	1.73	85002417
35	West Bengal	2118696	50	2097114	21532	0.0	98.98	1.02	32199722

Looking for summary :

```
data.columns
```

output :

```
Index(['State/UTs', 'Total Cases', 'Active', 'Discharged', 'Deaths
'Active Rat io', 'Discharge Ratio', 'Death Ratio', 'Population'],
dtype='object')
```

Looking for missing values :

In [11]:

linkcode

```
data.isnull() #for cheking null values
```

output :

	State/UTS	Total Cases	Active	Dischrged	Death	Active Ratio	Discharged Ratio	Death Ratio	Population
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
5	False	False	False	False	False	False	False	False	False

6	False	False	False	False	False	False	False	False	False
7	False	False	False	False	False	False	False	False	False
8	False	False	False	False	False	False	False	False	False
9	False	False	False	False	False	False	False	False	False
10	False	False	False	False	False	False	False	False	False

```
data.isnull().sum()  # for number of null values
```

output :

```
State/UTs        0
Total Cases      0
 Active          0
Discharged       0
 Deaths          0
 Active Ratio    0
Discharge Rati   0
Death Ratio      0
Population       0
dtype: int64
```

Data visualization :

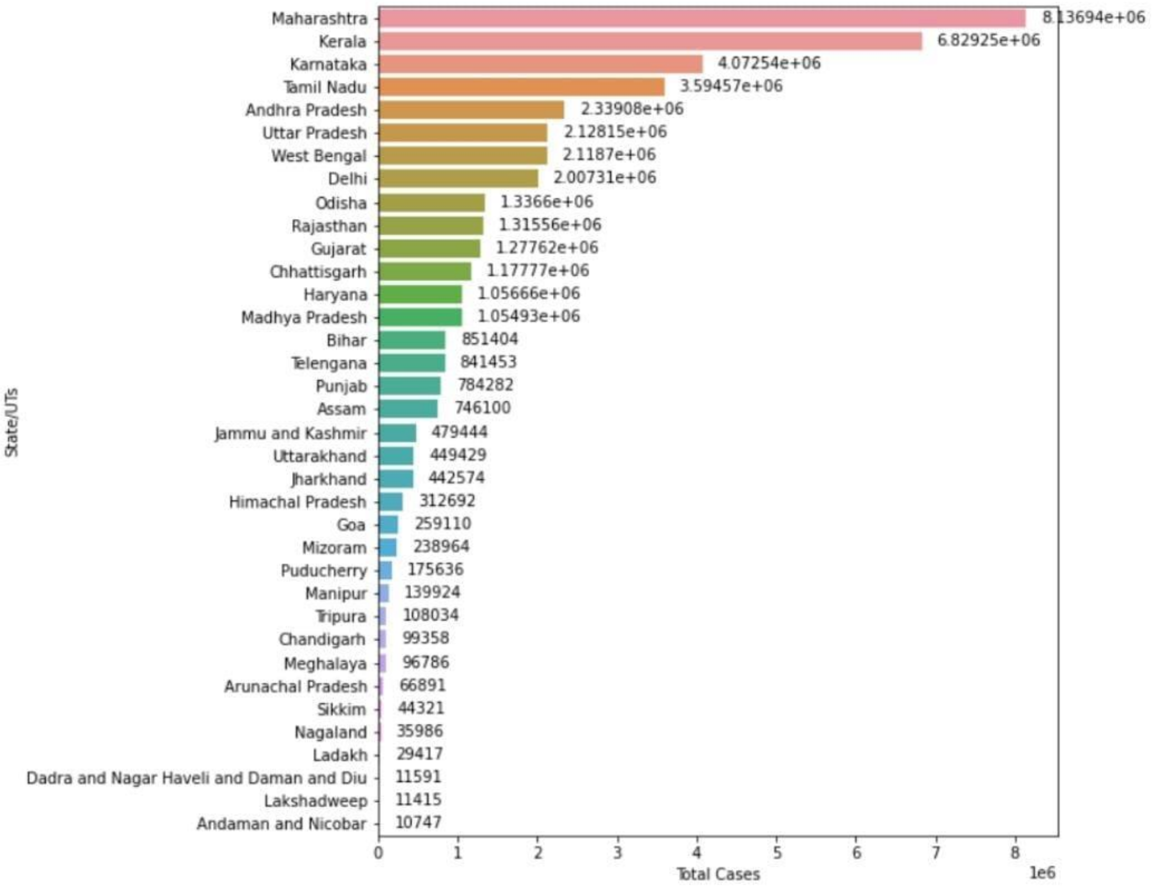
In [13]:

```
Total_cases = data[['State/UTs','Total Cases']].sort_values s(by=['Total
Cases'],ascen ding=False).reset_index(drop=True) In[14]:
```

linkcode

```
# Draw barplot
plt.figure(figsize=(8,10)ax=sns.barplot(x='Total
Cases',y='State/UTs',data=Total_cases)
ax.bar_label(ax.containers[0],padding=10,fmt='%g'); plt.show()
```

Output :



Complete Code:

```
 import numpy as np
 import pandas as pd
   import matplotlib.pyplot as plt
  data = pd.read_csv('case_time_series.csv')
  Y = data.iloc[61:,1].values
  R = data.iloc[61:,3].values
  D = data.iloc[61:,5].values

   X = data.iloc[61:,0]
   plt.figure(figsize=(25,8))
   ax = plt.axes()
  .grid(linewidth=0.4, color='#8f8f8f')
  ax.set_facecolor("black")
  ax.set_xlabel('\nDate',size=25,color='#4bb4f2')
  ax.set_ylabel('Number of Confirmed Cases\n',
size=25,color='#4bb4f2')
  ax.plot (X,Y, colour ='#1F77B4', marker='o', linewidth=4,
marker size=15, marker edge colour='#035E9B') .
```

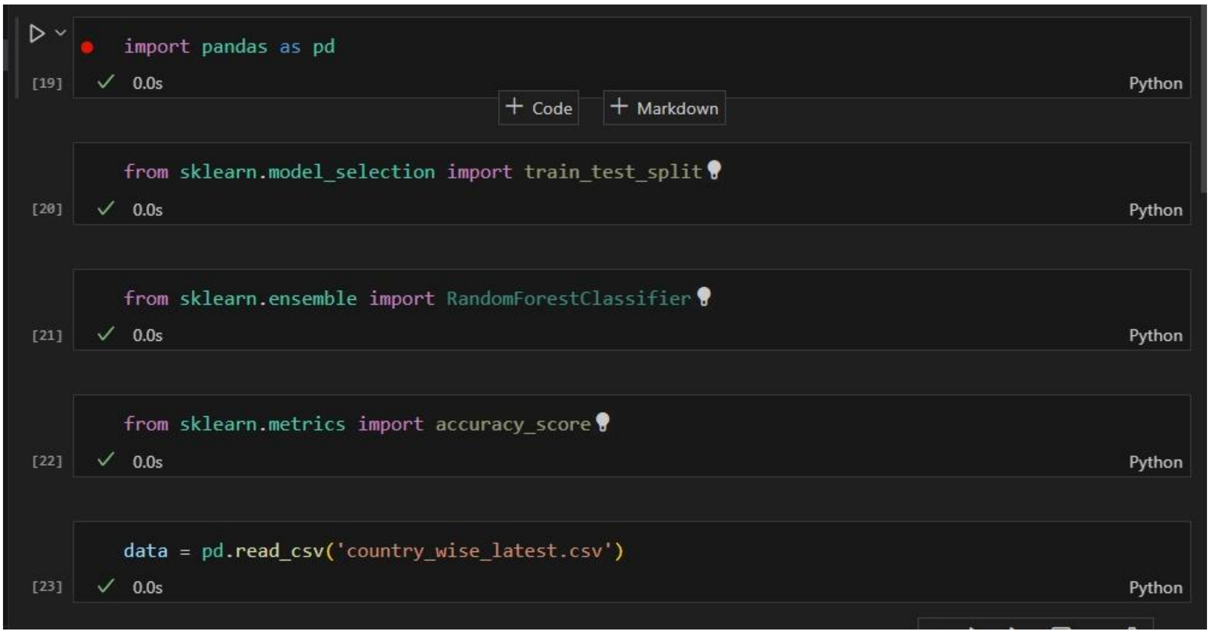
program :

```
# Import necessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

# Load and preprocess the data
```

```
data = pd.read_csv('vaccine_data.csv')
X = data.drop('target_variable', axis=1)
y = data['target_variable']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize and train a random forest classifier
clf = RandomForestClassifier()
clf.fit(X_train, y_train)

# Make predictions
y_pred = clf.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy}')
```

output :

```
import necessary libraries
```



```
import pandas as pd
[19] ✓ 0.0s
```

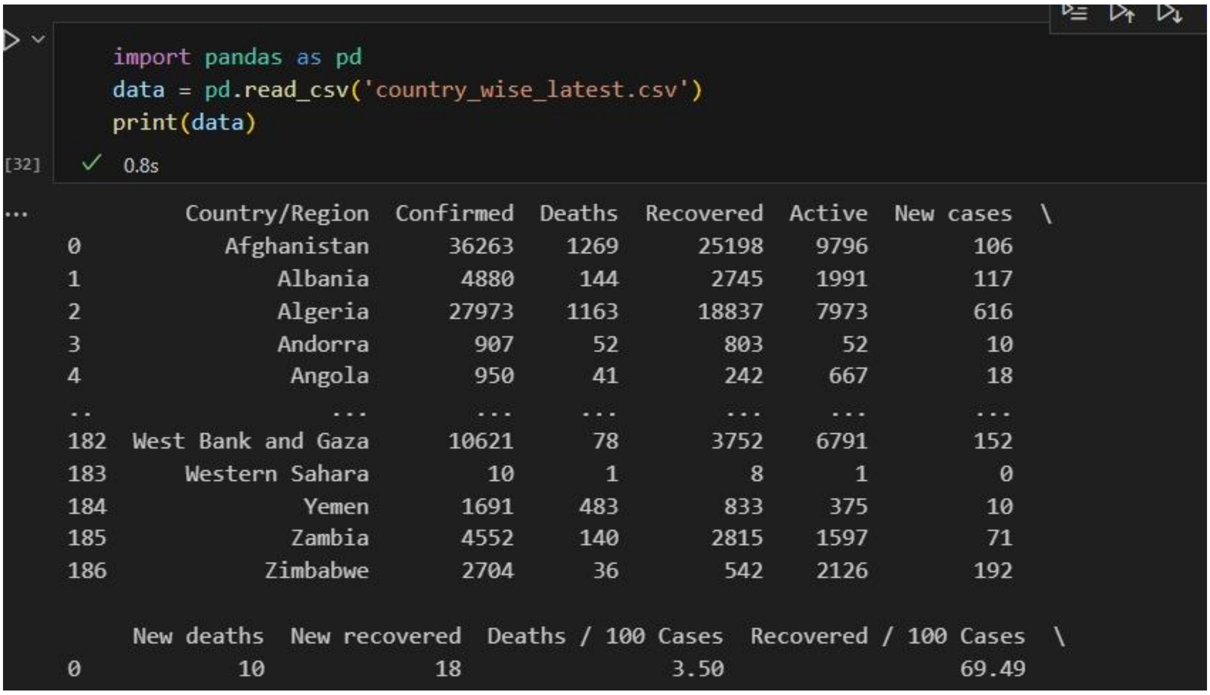
```
from sklearn.model_selection import train_test_split
[20] ✓ 0.0s
```

```
from sklearn.ensemble import RandomForestClassifier
[21] ✓ 0.0s
```

```
from sklearn.metrics import accuracy_score
[22] ✓ 0.0s
```

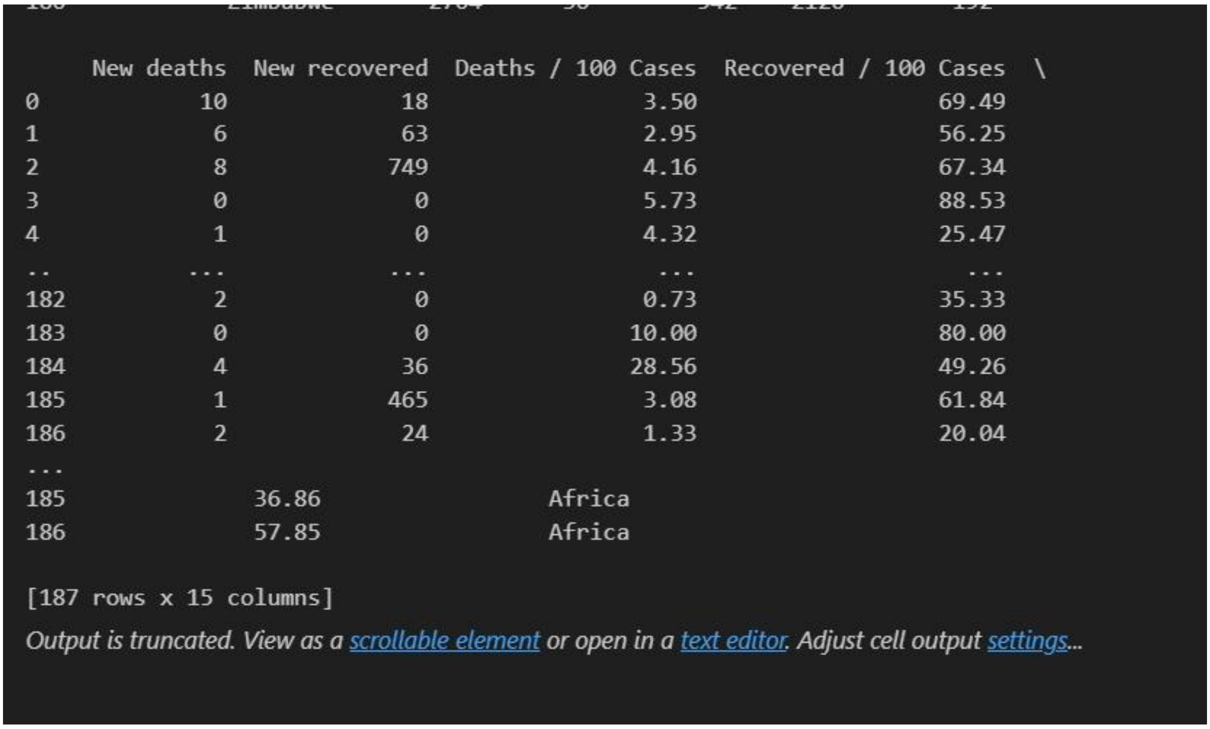
```
data = pd.read_csv('country_wise_latest.csv')
[23] ✓ 0.0s
```

Load and preprocess the data



```
import pandas as pd
data = pd.read_csv('country_wise_latest.csv')
print(data)
[32] ✓ 0.8s
```

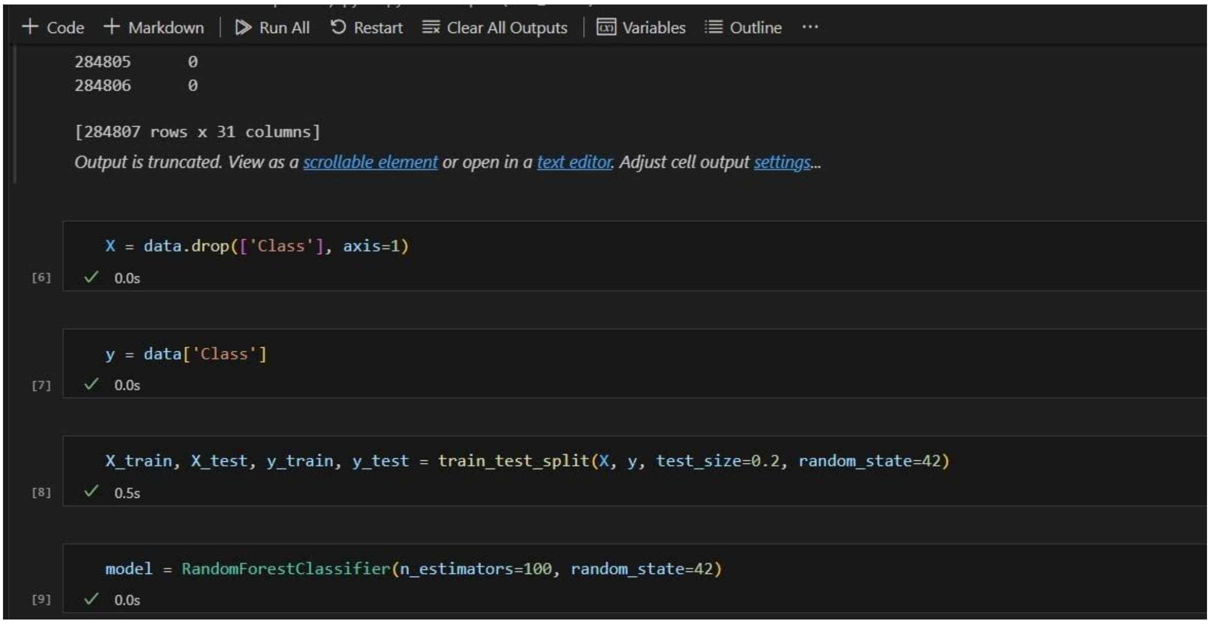
```
...       Country/Region  Confirmed  Deaths  Recovered  Active  New cases  \
    0          Afghanistan      36263    1269      25198    9796        106
    1              Albania       4880     144       2745    1991        117
    2              Algeria      27973    1163      18837    7973        616
    3              Andorra        907      52        803      52         10
    4               Angola        950      41        242     667         18
    ..                 ...        ...     ...        ...     ...        ...
    182  West Bank and Gaza      10621      78       3752    6791        152
    183      Western Sahara         10       1          8       1          0
    184               Yemen       1691     483        833     375         10
    185              Zambia       4552     140       2815    1597         71
    186            Zimbabwe       2704      36        542    2126        192

         New deaths  New recovered  Deaths / 100 Cases  Recovered / 100 Cases  \
    0            10             18                3.50                  69.49
```



```
     New deaths  New recovered  Deaths / 100 Cases  Recovered / 100 Cases  \
0            10             18                3.50                  69.49
1             6             63                2.95                  56.25
2             8            749                4.16                  67.34
3             0              0                5.73                  88.53
4             1              0                4.32                  25.47
..          ...            ...                 ...                    ...
182           2              0                0.73                  35.33
183           0              0               10.00                  80.00
184           4             36               28.56                  49.26
185           1            465                3.08                  61.84
186           2             24                1.33                  20.04
...
185            36.86               Africa
186            57.85               Africa

[187 rows x 15 columns]
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

Split the data into training and testing sets



```
284805      0
284806      0

[284807 rows x 31 columns]
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

```
X = data.drop(['Class'], axis=1)
```

[6] ✓ 0.0s

```
y = data['Class']
```

[7] ✓ 0.0s

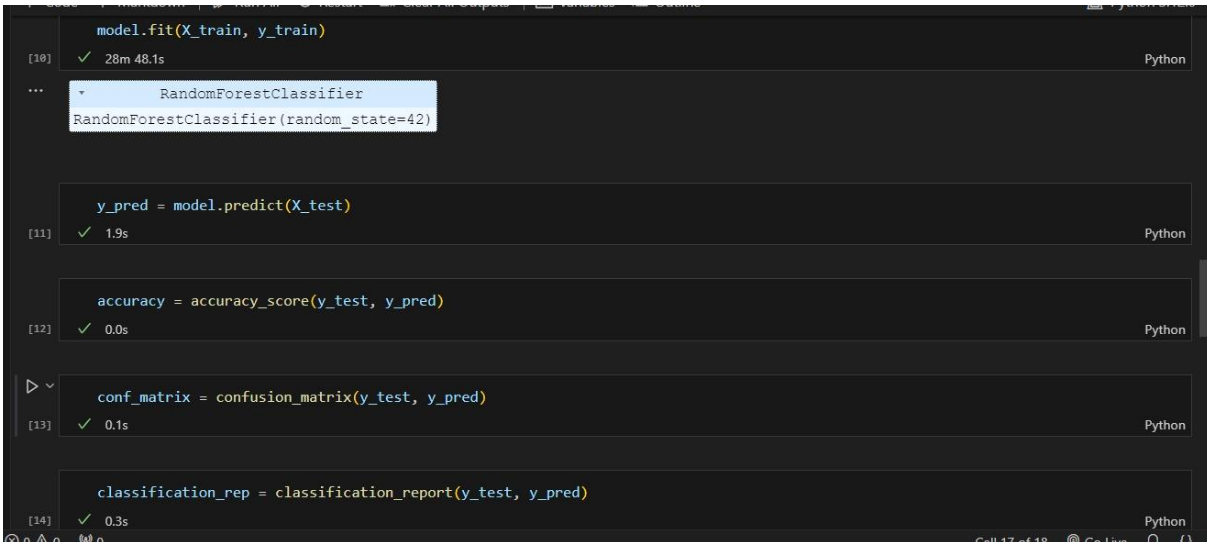
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

[8] ✓ 0.5s

```
model = RandomForestClassifier(n_estimators=100, random_state=42)
```

[9] ✓ 0.0s

Initialize and train a random forest classfier



```
model.fit(X_train, y_train)
[10]  ✓ 28m 48.1s                                                    Python

RandomForestClassifier
RandomForestClassifier(random_state=42)

y_pred = model.predict(X_test)
[11]  ✓ 1.9s                                                         Python

accuracy = accuracy_score(y_test, y_pred)
[12]  ✓ 0.0s                                                         Python

conf_matrix = confusion_matrix(y_test, y_pred)
[13]  ✓ 0.1s                                                         Python

classification_rep = classification_report(y_test, y_pred)
[14]  ✓ 0.3s                                                         Python
```

Conclusion :

The COVID-19 pandemic has led to questions about many aspects in India—the quality of health care, the response of governments and institutions, and issues related to law and order.