SARVA SHIKSHA ABHIYAAN EXPLORATORY DATA ANALYSIS

Course Code: IT495
Semester 2
Course Instructor: Gopinath Panda

Group Number – 01

202218037
(Muskan Khare)

202218049
(Riya Kumari)

202218053
(Dhruv Solanki)

202218054
(Chinmaya Pandey)

202218061
(Jatan Sahu)

DECLARATION

We, [202218037, 202218049, 202218053, 202218054, 202218061] hereby declare that the EDA project work presented in this report is my original work and has not been submitted for any other academic degree. All the sources cited in this report have been appropriately referenced.

We acknowledge that the data used in this project is obtained from the Sarva Shiksha Abhiyan (SSA) dataset available at *openbudget site*. We also declare that we have adhered to the terms and conditions mentioned in the website for using the dataset.

We confirm that the dataset used in this project is true and accurate to the best of my knowledge. However, any errors or inaccuracies in the dataset are not the responsibility of the website or its administrators.

We acknowledge that we have received no external help or assistance in conducting the EDA project, except for the guidance provided by our Prof. Gopinath Panda Sir. We declare that there is no conflict of interest in conducting this EDA project.

We hereby sign the declaration statement and confirm the submission of this report on 2nd May, 2023.

Muskan Khare

202218037
(Muskan Khare)

Riya Kumari

202218049
(Riya Kumari)

202218053
(Dhruv Solanki)

Chinmay

202218054
(Chinmaya Pandey)

Jat

202218061
(Jatan Sahu)

CERTIFICATE

This is to certify that **Group 1** comprising Muskan Khare, Riya Kumari, Dhruv Solanki, Chinmaya Pandey, and Jatan Sahu has successfully completed an exploratory data analysis (EDA) project on the Sarva Shiksha Abhiyan (SSA) dataset, which was obtained from openbudget.

The EDA project presented by Group 1 is their original work and has been completed under the guidance of Prof. Gopinath Panda Sir, who has provided support and guidance throughout the project. The project is based on a thorough analysis of the SSA dataset, and the results presented in the report are based on the data obtained from the dataset.

I certify that the SSA dataset used in this project is true and accurate to the best of my knowledge. However, any errors or inaccuracies in the dataset are not the responsibility of the website or its administrators.

This certificate is issued to recognize the successful completion of the EDA project on the Sarva Shiksha Abhiyan dataset, which demonstrates the analytical skills and knowledge of the student/researcher in the field of data analysis.

Signed,
Gopinath Panda,
Faculty,
Dhirubhai Ambani Institute of Information
and Communication Technology,
2nd May 2023.

INDEX

INTRODUCTION

PROBLEM DESCRIPTION

Sarva Shiksha Abhiyan (SSA) is a flagship program of the Indian government aimed at providing free and compulsory education to all children in the age group of 6 to 14 years. The program was launched in 2001 and is implemented in partnership with state governments, local communities, and other stakeholders. In our project we work using various features available in the dataset to us, and analyse and predict results. EDA can be used to analyse learning outcomes; this can help in identifying factors that affect learning outcomes and designing interventions to improve learning outcomes.

There 5 steps in Exploratory Data Analysis which need to be followed, which are

1. Data Collection: This involves acquiring relevant data sets from various sources such as databases, files, surveys, or web scraping.

2. Data Understanding: It is a crucial step in EDA. The goal of data understanding is to gain insights into the structure, content, and quality of the data.

3. Data Cleaning: This step involves checking the quality of the collected data and identifying any issues that may affect the analysis. The process may include removing duplicates, filling in missing values, and correcting data format errors.

4. Data Visualization and Analysis: In this step, we use various statistical and visualization techniques to explore the data and identify patterns and trends. This includes examining the distribution of data, checking for outliers, and identifying correlations between variables.

5. Data Prediction: After exploring the data, we can build and train various models to predict or classify the target variable. This step involves selecting appropriate algorithms and evaluating their performance on the data.

1. DATA COLLECTION

The Sarva Shiksha Abhiyan (SSA) dataset is available on openbudgetsindia.org and provides information on various aspects of the SSA programme, which aims to provide free and compulsory education to all children between the ages of 6 and 14 in India. The dataset covers the period from 2015-16 to 2017-18 and contains information on budget allocation, funds released, expenditure incurred, and unspent balances at the state level.

The SSA dataset can be a valuable resource for researchers, analysts, and policymakers interested in understanding the progress and challenges of the programme. By analysing the data and identifying trends and patterns, stakeholders can test hypotheses about the effectiveness of the programme and identify areas of need. Policymakers can use the dataset to target resources to areas with the greatest need and develop targeted interventions to address disparities in education outcomes.

Overall, the SSA dataset on openbudgetsindia.org can provide valuable insights into the progress and challenges of the SSA programme. By leveraging this data, stakeholders can work towards improving education outcomes for all children in India.

1.1. DATASET DESCRIPTION

The dataset covers the period from 2015-16 to 2017-18 and is based on data from the Government of India's District Information System for Education (DISE). The dataset contains 999 rows and 26 columns, including information on budget allocation, funds released, expenditure incurred, etc. The rows consist of state wise distribution of various features.

The dataset comprises following fields/features:

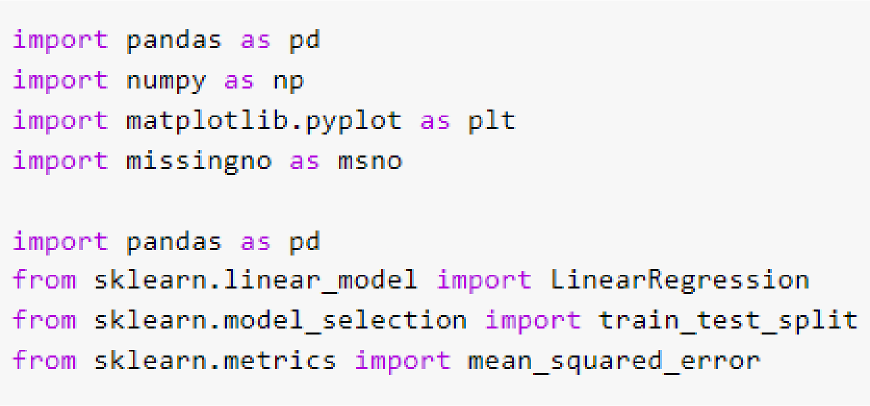
- State: A categorical variable indicating the state in India for which the data is reported.
- State UT Code: A numerical code for the state/UT (Union Territory).
- Financial Year: The year for which the budget and expenditure data are reported.
- Budget Approved: The budget approved for the SSA program in the given fiscal year.
- Funds Released by the Government of India: The funds released by the central government for the SSA program in the given financial year.
- Funds Released by the States/UTs: The funds released by the state/UT government for the SSA program in the given financial year.
- Total Funds Released (Government of India and States' Share): The total funds released for the SSA program in the given financial year.

- Expenditure Incurred by the States/UTs: The expenditure incurred by the state/UT government for the SSA program in the given financial year.
- Unspent Balance: The unspent balance of the funds released for the SSA program in the given financial year.
- Extent of Funds Released against Budget Approved (%): The extent to which the funds released for the SSA program match the budget approved for the given financial year.
- Extent of Funds Utilised against Budget Approved (%): The extent to which the funds utilized for the SSA program match the budget approved for the given financial year.

2. DATA UNDERSTANDING

2.1. IMPORTING LIBRARIES:

In programming, libraries are collections of pre-written code that can be used to perform specific tasks or functions. They are designed to save time and effort by providing ready-made solutions for common problems.



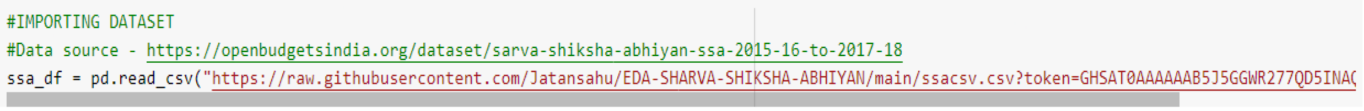
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import missingno as msno

import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
```

Once the library is imported, you can use its functions and classes in your code by referencing them with the library name and a dot notation.

LIBRARIES IMPORTED:

1. Pandas: Pandas is an open-source library for data manipulation and analysis in Python. It provides data structures like DataFrame, Series, and Panel for working with structured data in a fast, efficient, and easy way.
2. NumPy: NumPy is a fundamental package for scientific computing in Python. It provides support for large, multi-dimensional arrays and matrices, along with a large collection of mathematical functions to operate on these arrays.
3. Matplotlib: Matplotlib is a plotting library for Python that provides a variety of 2D and 3D plots for visualization. It is widely used for creating static, animated, and interactive visualizations in Python.
4. Missingno: Missingno is a Python library for visualizing missing data in a dataset. It provides a variety of tools for identifying and visualizing missing data patterns in a dataset.
5. LinearRegression: LinearRegression is a class in the scikit-learn library that provides linear regression models for machine learning applications.
6. train_test_split: train_test_split is a function in the scikit-learn library that splits a dataset into training and testing subsets.
7. mean_squared_error: mean_squared_error is a function in the scikit-learn library that calculates the mean squared error (MSE) between the actual and predicted values of a regression problem. It is a popular metric for evaluating the performance of regression models. A lower MSE value indicates a better model performance.

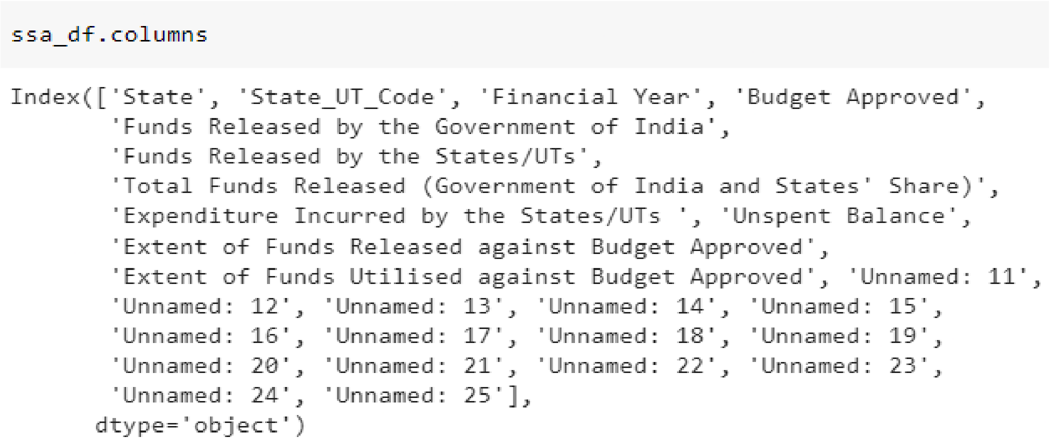
2.2. IMPORTING DATASET FROM GIT



```
#IMPORTING DATASET
#Data source - https://openbudgetsindia.org/dataset/sarva-shiksha-abhiyan-ssa-2015-16-to-2017-18
ssa_df = pd.read_csv("https://raw.githubusercontent.com/Jatansahu/EDA-SHARVA-SHIKSHA-ABHIYAN/main/ssacsv.csv?token=GHSAT0AAAAAAB5J5GGWR277QD5INAC
```

The code imports a dataset name "ssacsv.csv" from a GitHub repository and creates a Pandas DataFrame named "ssa_df". The "read_csv ()" function is used to read the CSV file into a DataFrame object. The URL of the raw CSV file is provided as an argument to the function.

2.3. READING COLUMNS IN THE DATASETS

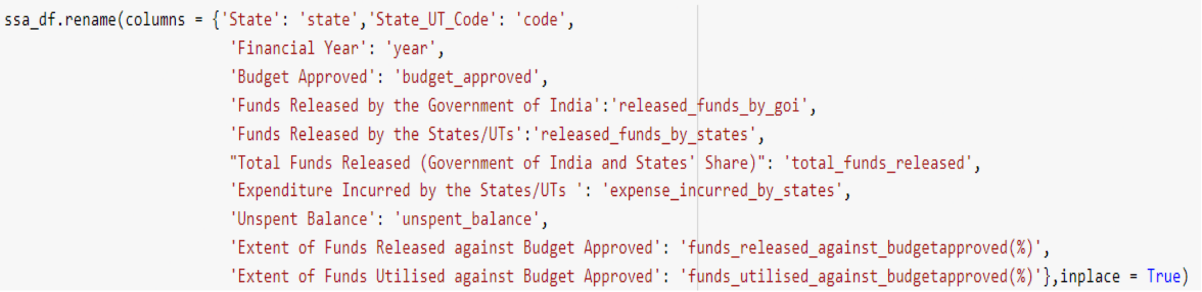


```
ssa_df.columns
```

```
Index(['State', 'State_UT_Code', 'Financial Year', 'Budget Approved',
       'Funds Released by the Government of India',
       'Funds Released by the States/UTs',
       'Total Funds Released (Government of India and States' Share)',
       'Expenditure Incurred by the States/UTs ', 'Unspent Balance',
       'Extent of Funds Released against Budget Approved',
       'Extent of Funds Utilised against Budget Approved', 'Unnamed: 11',
       'Unnamed: 12', 'Unnamed: 13', 'Unnamed: 14', 'Unnamed: 15',
       'Unnamed: 16', 'Unnamed: 17', 'Unnamed: 18', 'Unnamed: 19',
       'Unnamed: 20', 'Unnamed: 21', 'Unnamed: 22', 'Unnamed: 23',
       'Unnamed: 24', 'Unnamed: 25'],
      dtype='object')
```

When we call ssa_df.columns , it returns a Pandas Index object containing the names of all the columns in the ssa_df DataFrame. This tells us the names of all the columns in the DataFrame, and we can use these names to access specific columns or perform operations on the entire DataFrame.

2.4. RENAMING ALL COLUMN NAMES

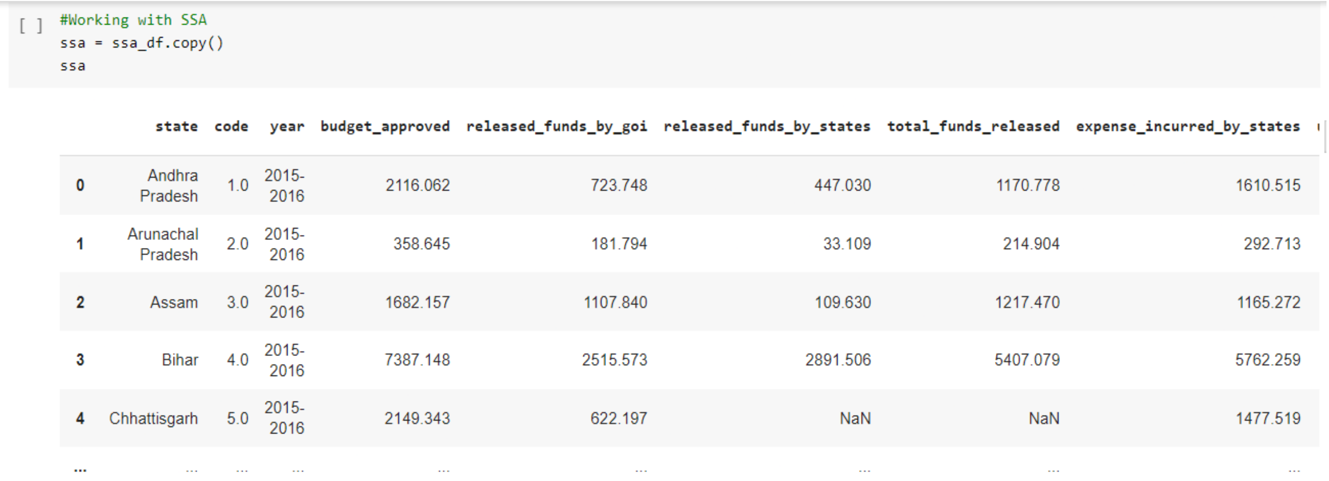


```
ssa_df.rename(columns = {'State': 'state','State_UT_Code': 'code',
                         'Financial Year': 'year',
                         'Budget Approved': 'budget_approved',
                         'Funds Released by the Government of India':'released_funds_by_goi',
                         'Funds Released by the States/UTs':'released_funds_by_states',
                         "Total Funds Released (Government of India and States' Share)": 'total_funds_released',
                         'Expenditure Incurred by the States/UTs ': 'expense_incurred_by_states',
                         'Unspent Balance': 'unspent_balance',
                         'Extent of Funds Released against Budget Approved': 'funds_released_against_budgetapproved(%)',
                         'Extent of Funds Utilised against Budget Approved': 'funds_utilised_against_budgetapproved(%)'},inplace = True)
```

Renaming all column names in a Pandas DataFrame is important because it helps to make the column names more descriptive and easier to understand.

This code renames the columns of the `ssa_df` DataFrame using the `rename()` method with the `inplace=True` parameter to modify the DataFrame in place. These new column names are more descriptive and easier to understand than the original names, which can improve the readability and interpretability of the data.

2.5. COPYING DATA TO ssa DATAFRAME FROM ssa_df

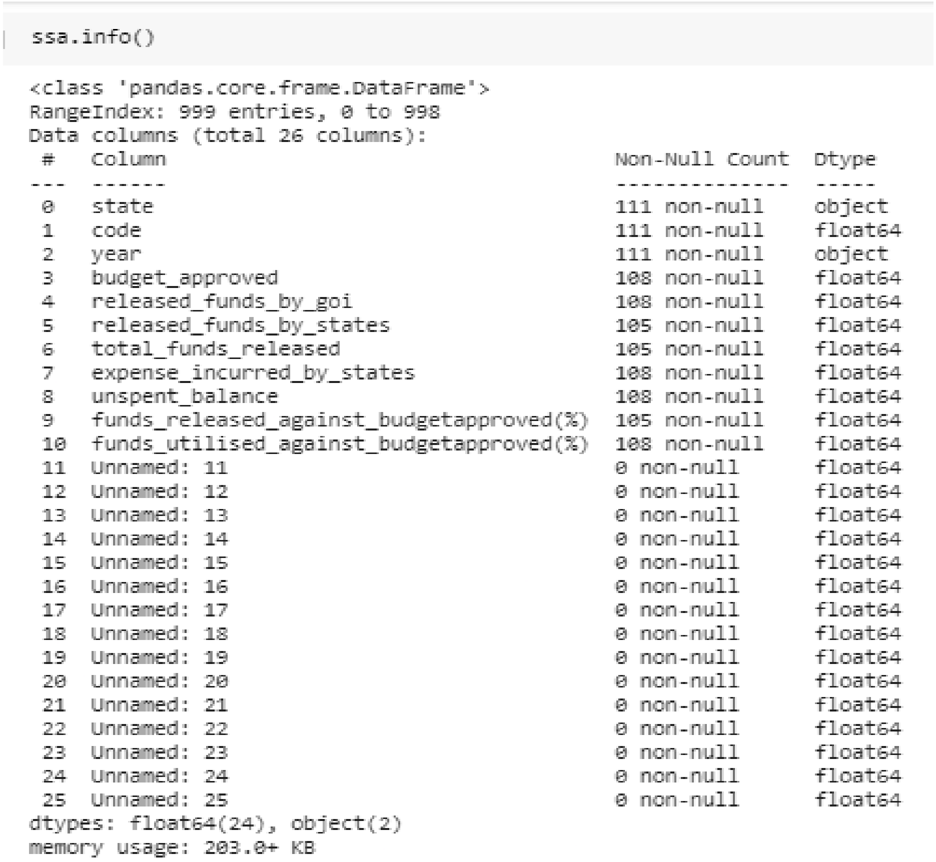


```
[ ] #Working with SSA
    ssa = ssa_df.copy()
    ssa
```

	state	code	year	budget_approved	released_funds_by_goi	released_funds_by_states	total_funds_released	expense_incurred_by_states
0	Andhra Pradesh	1.0	2015-2016	2116.062	723.748	447.030	1170.778	1610.515
1	Arunachal Pradesh	2.0	2015-2016	358.645	181.794	33.109	214.904	292.713
2	Assam	3.0	2015-2016	1682.157	1107.840	109.630	1217.470	1165.272
3	Bihar	4.0	2015-2016	7387.148	2515.573	2891.506	5407.079	5762.259
4	Chhattisgarh	5.0	2015-2016	2149.343	622.197	NaN	NaN	1477.519
...	...	...	...	...	...	...	...	...

ssa_df.copy() creates a new copy of the `ssa_df` DataFrame and assigns it to a new variable called `ssa`. This is done because we want to make changes to the DataFrame but want to preserve the original data in case you need to revert back to it later.

2.6. DATA INFORMATION

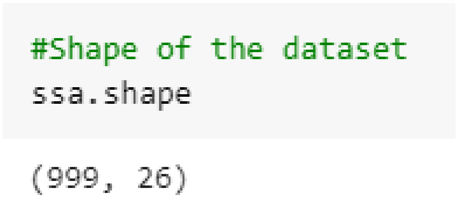


```
ssa.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 999 entries, 0 to 998
Data columns (total 26 columns):
 #   Column                                 Non-Null Count  Dtype  
---  ------                                 --------------  -----  
 0   state                                  111 non-null    object 
 1   code                                   111 non-null    float64
 2   year                                   111 non-null    object 
 3   budget_approved                        108 non-null    float64
 4   released_funds_by_goi                  108 non-null    float64
 5   released_funds_by_states               105 non-null    float64
 6   total_funds_released                   105 non-null    float64
 7   expense_incurred_by_states             108 non-null    float64
 8   unspent_balance                        108 non-null    float64
 9   funds_released_against_budgetapproved(%)  105 non-null    float64
 10  funds_utilised_against_budgetapproved(%)  108 non-null    float64
 11  Unnamed: 11                            0 non-null      float64
 12  Unnamed: 12                            0 non-null      float64
 13  Unnamed: 13                            0 non-null      float64
 14  Unnamed: 14                            0 non-null      float64
 15  Unnamed: 15                            0 non-null      float64
 16  Unnamed: 16                            0 non-null      float64
 17  Unnamed: 17                            0 non-null      float64
 18  Unnamed: 18                            0 non-null      float64
 19  Unnamed: 19                            0 non-null      float64
 20  Unnamed: 20                            0 non-null      float64
 21  Unnamed: 21                            0 non-null      float64
 22  Unnamed: 22                            0 non-null      float64
 23  Unnamed: 23                            0 non-null      float64
 24  Unnamed: 24                            0 non-null      float64
 25  Unnamed: 25                            0 non-null      float64
dtypes: float64(24), object(2)
memory usage: 203.0+ KB
```

The ssa_df.info() method provides information about the DataFrame(rows=999, columns=26) , including the number of non-null values, the data type of each column, and the memory usage of the DataFrame.

2.7. DATASET SHAPE

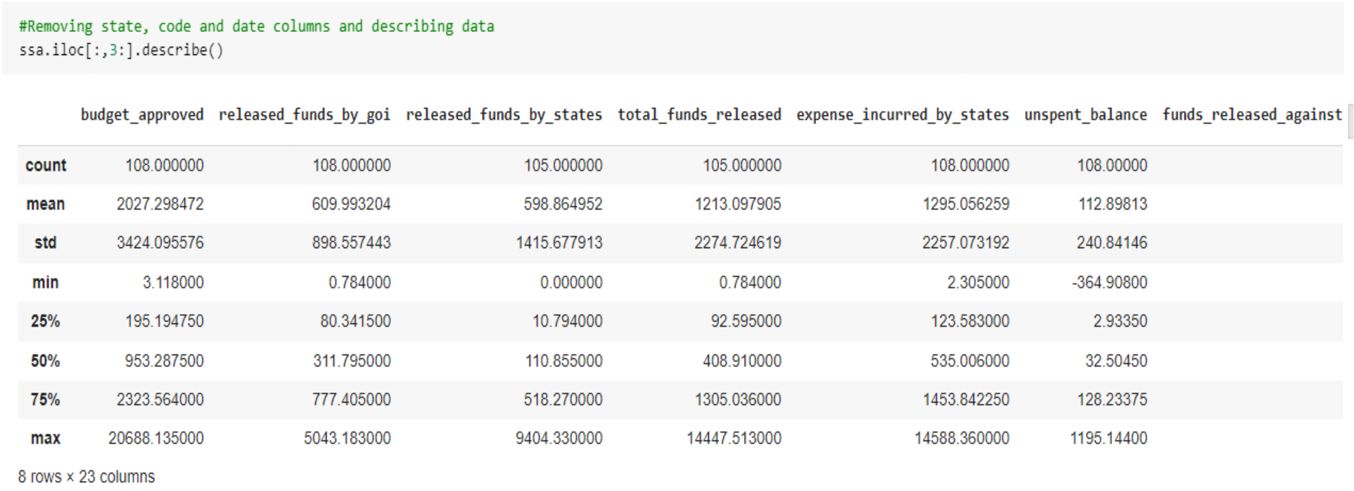


```
#Shape of the dataset
ssa.shape
```

```
(999, 26)
```

We can note the shape of the dataset if 999 rows and 26 columns. Knowing the shape of the dataset is important because it gives an idea about the size of the dataset and the amount of data it contains.

2.8. DESCRIPTIVE ANALYSIS



```
#Removing state, code and date columns and describing data
ssa.iloc[:,3:].describe()
```

	budget_approved	released_funds_by_goi	released_funds_by_states	total_funds_released	expense_incurred_by_states	unspent_balance	funds_released_against
count	108.000000	108.000000	105.000000	105.000000	108.000000	108.00000	
mean	2027.298472	609.993204	598.864952	1213.097905	1295.056259	112.89813	
std	3424.095576	898.557443	1415.677913	2274.724619	2257.073192	240.84146	
min	3.118000	0.784000	0.000000	0.784000	2.305000	-364.90800	
25%	195.194750	80.341500	10.794000	92.595000	123.583000	2.93350	
50%	953.287500	311.795000	110.855000	408.910000	535.006000	32.50450	
75%	2323.564000	777.405000	518.270000	1305.036000	1453.842250	128.23375	
max	20688.135000	5043.183000	9404.330000	14447.513000	14588.360000	1195.14400	

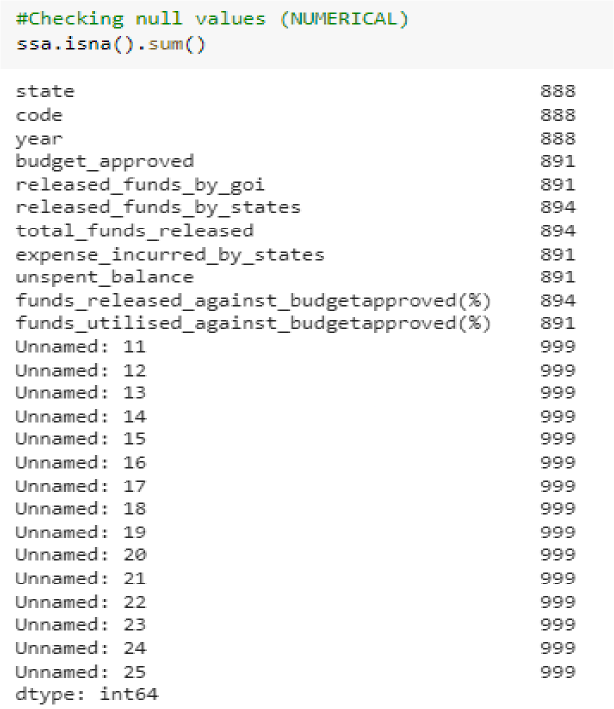
8 rows × 23 columns

This code uses pandas 'iloc' method to select all rows and columns from the 4th column to the end of the DataFrame. Then, it applies the describe method to only the selected columns to generate summary statistics such as count, mean, standard deviation, minimum, and maximum values for each column.

3. DATA CLEANING

PART A-MISSING VALUES(INTERPRETING NUMERICALLY)

3.1. CHECKING FOR MISSING VALUES

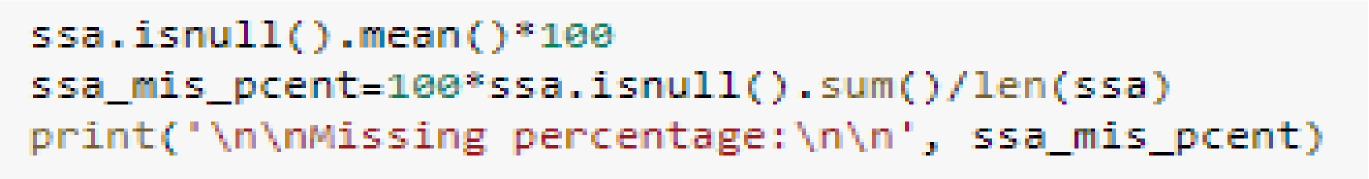


```
#Checking null values (NUMERICAL)
ssa.isna().sum()
```

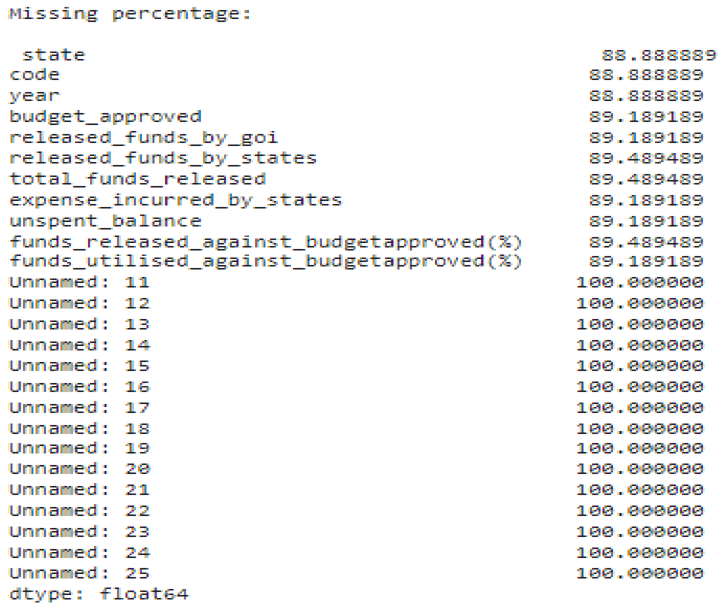
```
state                                     888
code                                      888
year                                      888
budget_approved                           891
released_funds_by_goi                     891
released_funds_by_states                  894
total_funds_released                      894
expense_incurred_by_states                891
unspent_balance                           891
funds_released_against_budgetapproved(%)  894
funds_utilised_against_budgetapproved(%)  891
Unnamed: 11                               999
Unnamed: 12                               999
Unnamed: 13                               999
Unnamed: 14                               999
Unnamed: 15                               999
Unnamed: 16                               999
Unnamed: 17                               999
Unnamed: 18                               999
Unnamed: 19                               999
Unnamed: 20                               999
Unnamed: 21                               999
Unnamed: 22                               999
Unnamed: 23                               999
Unnamed: 24                               999
Unnamed: 25                               999
dtype: int64
```

This is used to count the number of missing values (NaN values) in each column of "ssa". The method ".isna()" creates a Boolean mask indicating where there are missing values in the DataFrame, returning "True" for missing values and "False" otherwise. The method ".sum()" is then applied which counts the number of "True" values (i.e. the number of missing values) for each column of the DataFrame. Checking for missing values is important because it can affect the accuracy and reliability of any data analysis.

3.2. CHECKING MISSING PERCENTAGE



```
ssa.isnull().mean()*100
ssa_mis_pcent=100*ssa.isnull().sum()/len(ssa)
print('\n\nMissing percentage:\n\n', ssa_mis_pcent)
```



```
Missing percentage:

 state                                          88.888889
code                                           88.888889
year                                           88.888889
budget_approved                                89.189189
released_funds_by_goi                          89.189189
released_funds_by_states                       89.489489
total_funds_released                           89.489489
expense_incurred_by_states                     89.189189
unspent_balance                                89.189189
funds_released_against_budgetapproved(%)       89.489489
funds_utilised_against_budgetapproved(%)       89.189189
Unnamed: 11                                   100.000000
Unnamed: 12                                   100.000000
Unnamed: 13                                   100.000000
Unnamed: 14                                   100.000000
Unnamed: 15                                   100.000000
Unnamed: 16                                   100.000000
Unnamed: 17                                   100.000000
Unnamed: 18                                   100.000000
Unnamed: 19                                   100.000000
Unnamed: 20                                   100.000000
Unnamed: 21                                   100.000000
Unnamed: 22                                   100.000000
Unnamed: 23                                   100.000000
Unnamed: 24                                   100.000000
Unnamed: 25                                   100.000000
dtype: float64
```

The code is used to compute the percentage of missing values in a Pandas DataFrame called ssa. It computes the total percentage of missing values across all columns in the DataFrame and returns a Pandas Series object that shows the total percentage of missing values for each column.

The percentage of missing values can give you an idea of the quality of the data in each column. If a column has a very high percentage of missing values, it may not be very useful for analysis or modelling, and may need to be removed or imputed with appropriate values.

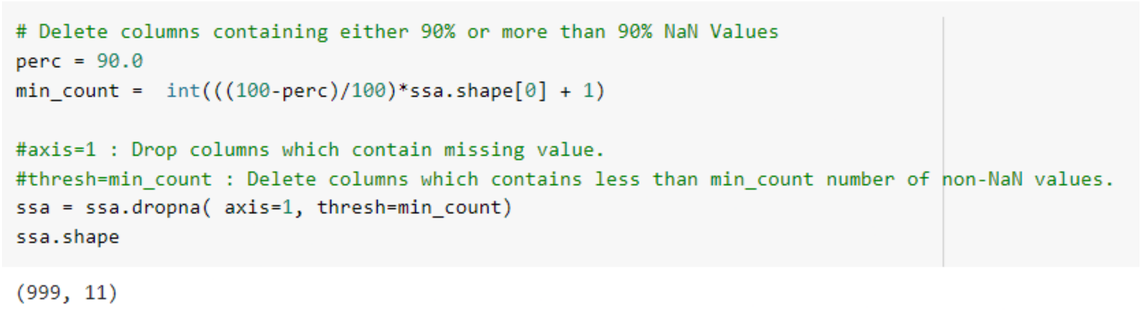
3.3. REASON FOR MISSING 100% DATA

Data is Missing Completely at Random (MCAR)

The probability of a value being missing is unrelated to both observed and unobserved data. In this case, the missingness is not related to any other variable in the dataset, and there is no systematic reason why certain values are missing. This means that the missing values are truly random, and there is no underlying pattern to their distribution.

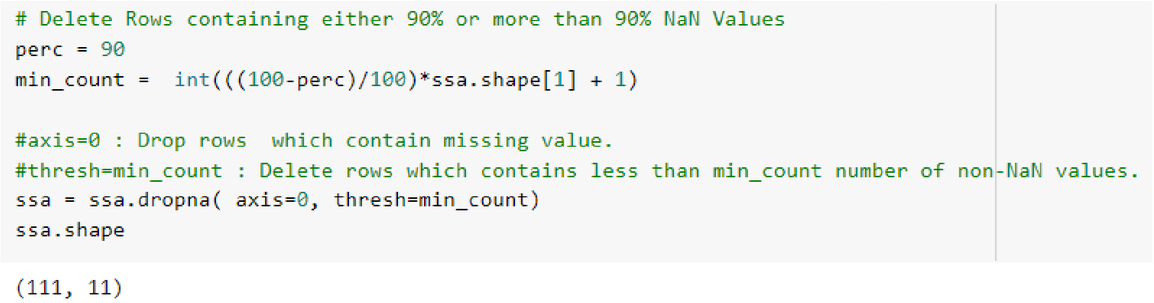
Dropping those missing values is the best solution.

3.4. DROPPING ROWS/COLUMNS



```
# Delete columns containing either 90% or more than 90% NaN Values
perc = 90.0
min_count =  int(((100-perc)/100)*ssa.shape[0] + 1)

#axis=1 : Drop columns which contain missing value.
#thresh=min_count : Delete columns which contains less than min_count number of non-NaN values.
ssa = ssa.dropna( axis=1, thresh=min_count)
ssa.shape
```

```
(999, 11)
```



```
# Delete Rows containing either 90% or more than 90% NaN Values
perc = 90
min_count =  int(((100-perc)/100)*ssa.shape[1] + 1)

#axis=0 : Drop rows  which contain missing value.
#thresh=min_count : Delete rows which contains less than min_count number of non-NaN values.
ssa = ssa.dropna( axis=0, thresh=min_count)
ssa.shape
```

```
(111, 11)
```

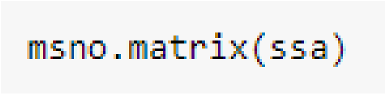
This drops columns from the DataFrame ssa that contain fewer than min_count non-NaN values. The axis=1 argument tells Pandas to drop columns, and the thresh=min_count argument tells Pandas to drop columns that contain fewer than min_count non-NaN values. Further we print the shape of the resulting DataFrame ssa after the columns containing 90% or more NaN values have been dropped.

PART B-VISUALIZING THE MISSING VALUES(GRAPHICAL)

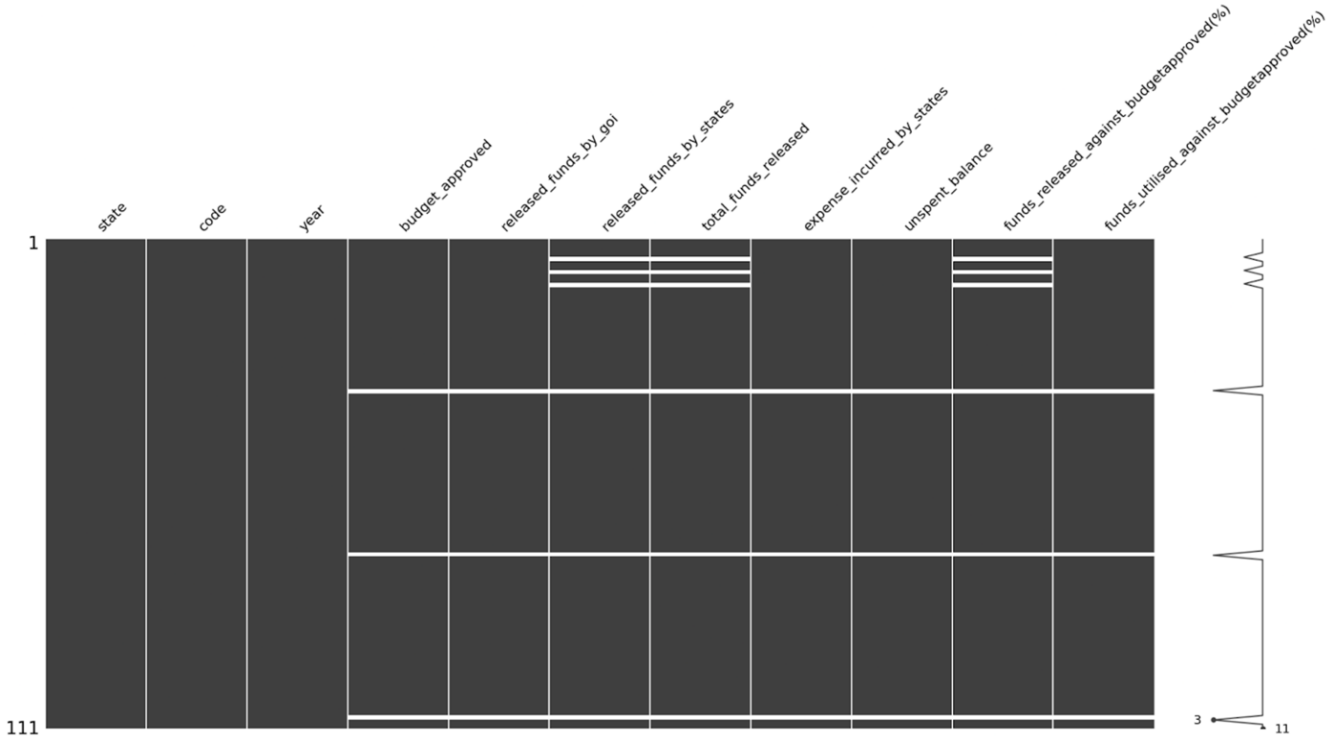
Visualizing the locations of the missing data

1. The plot appears white wherever there are missing values.
2. The sparkline on the right gives an idea of the general shape of the completeness of the data and points out the row with the minimum nullities and the total number of columns in a given dataset, at the bottom.

3.5. Visualize missing values as a matrix



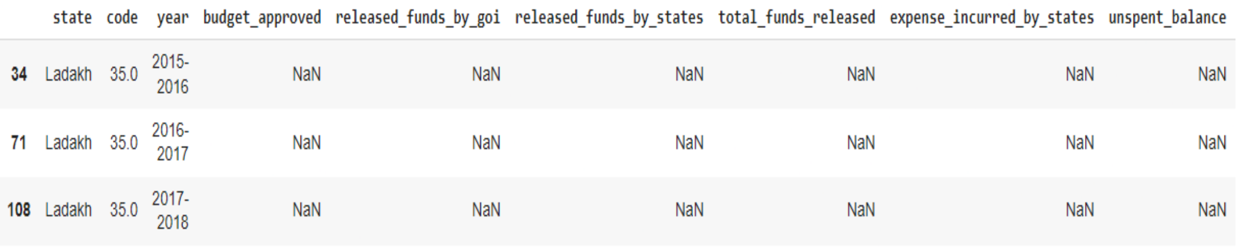
```
msno.matrix(ssa)
```



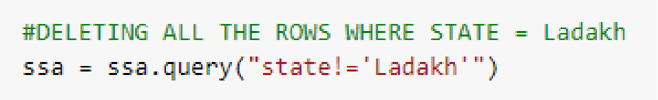
msno.matrix(ssa) is a function call that creates a matrix plot visualization of missing values in the DataFrame ssa. The plot appears white wherever there are missing values, The sparkline on the right gives an idea of the general shape of the completeness of the data and points out the row with the minimum nullities and the total number of columns in a given dataset, at the bottom.

PART C) DEALING WITH MISSING VALUES

3.6. DROP/DELETE MISSING VALUES



	state	code	year	budget_approved	released_funds_by_goi	released_funds_by_states	total_funds_released	expense_incurred_by_states	unspent_balance
34	Ladakh	35.0	2015-2016	NaN	NaN	NaN	NaN	NaN	NaN
71	Ladakh	35.0	2016-2017	NaN	NaN	NaN	NaN	NaN	NaN
108	Ladakh	35.0	2017-2018	NaN	NaN	NaN	NaN	NaN	NaN

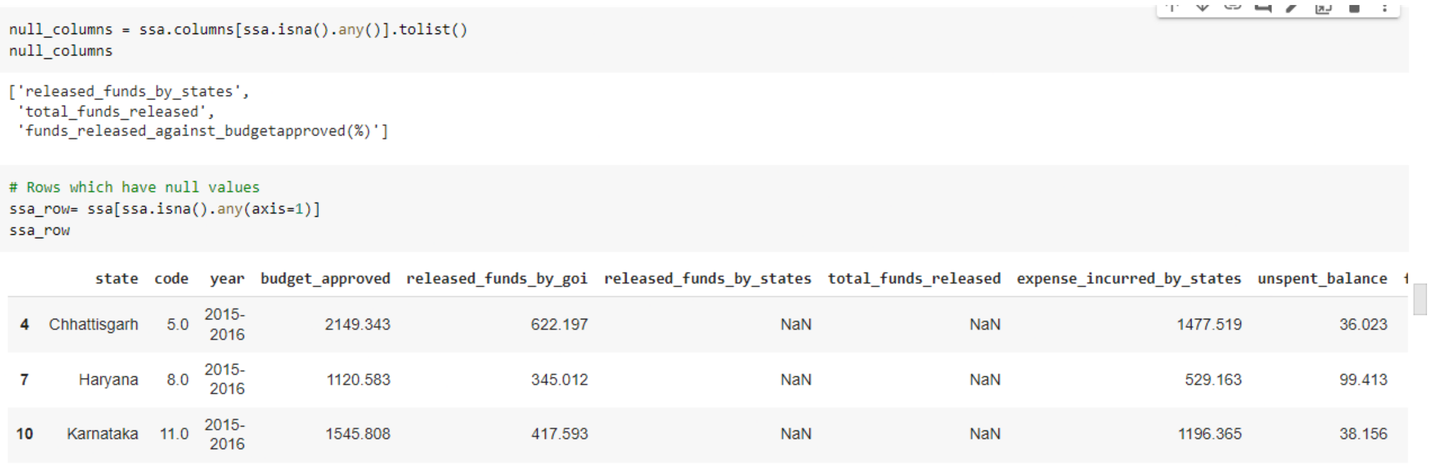


```
#DELETING ALL THE ROWS WHERE STATE = Ladakh
ssa = ssa.query("state!='Ladakh'")
```

We can see that the ‘budget_approved’ column is null for ‘state’ Ladakh. Furthermore, all columns of Ladakh are null. Since this data won't be beneficial for our analysis it's better to delete this from our dataset.

NOTE: We notice here that this missing value is a case of MISSING AT RANDOM.Missing at random (MAR) occurs when the missingness is not random, but where missingness can be fully accounted for by variables where there is complete information. We can note that the data we have is from fiscal years 2015-2018, Ladakh became a Union Territory in the year 2019 so we don’t have any substantial data for it. Since MAR is an assumption that is impossible to verify statistically, we must rely on its substantive reasonableness.

3.7. PRINTING ALL NULL STATE AND COLUMN NAMES



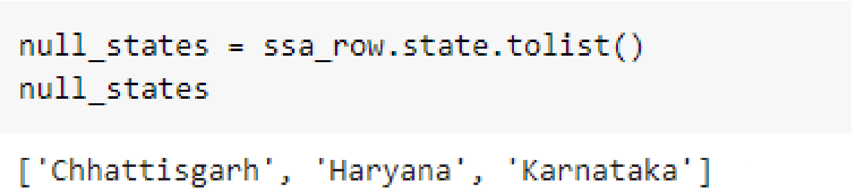
```
null_columns = ssa.columns[ssa.isna().any()].tolist()
null_columns
```

```
['released_funds_by_states',
 'total_funds_released',
 'funds_released_against_budgetapproved(%)']
```

```
# Rows which have null values
ssa_row= ssa[ssa.isna().any(axis=1)]
ssa_row
```

	state	code	year	budget_approved	released_funds_by_goi	released_funds_by_states	total_funds_released	expense_incurred_by_states	unspent_balance
4	Chhattisgarh	5.0	2015-2016	2149.343	622.197	NaN	NaN	1477.519	36.023
7	Haryana	8.0	2015-2016	1120.583	345.012	NaN	NaN	529.163	99.413
10	Karnataka	11.0	2015-2016	1545.808	417.593	NaN	NaN	1196.365	38.156

Through the above, we first extract the columns which have any null values. We need to handle missing values in order to further work on our data. The output tells us columns named 'released_funds_by_states', 'total_funds_released' and 'funds_released_against_budgetapp-roved (%)' have null values. We further create a new dataset ‘ssa_row’ which extracts all the rows with null values in any of the columns.



```
null_states = ssa_row.state.tolist()
null_states
```

```
['Chhattisgarh', 'Haryana', 'Karnataka']
```

Using the newly developed dataset, we find out the states where there are missing values. For States Chhattisgarh, Haryana and Karnataka we have missing values in any of the columns.

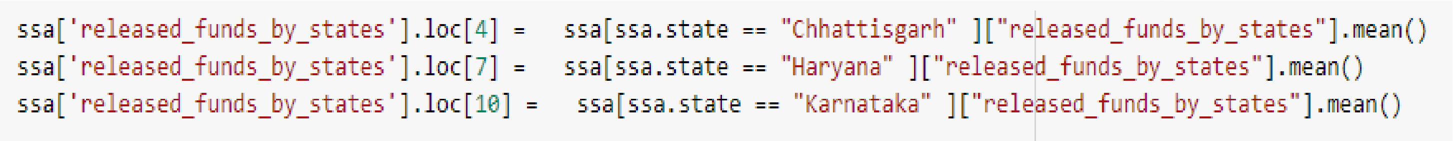
PART D) REASONS AND IMPUTING NULL VALUES

TYPES OF MISSING VALUES:

1. **Missing Completely At Random (MCAR):** In this mechanism, the missingness is completely random and unrelated to any variables in the dataset, whether observed or unobserved.
2. **Missing At Random (MAR):** In this mechanism, the missingness depends on other observed variables in the dataset, but not on the missing variable itself.
3. **Missing Not At Random (MNAR):** In this mechanism, the missingness depends on the value of the missing variable itself, or on other variables that are not observed.

3.8. REASON AND IMPUTING MISSING VALUE OF released_funds_in_state

Here missing data is MAR. To impute the missing values for the columns 'funds_released_by_the_states',for Chhattisgarh, Haryana, and Karnataka for the year 2015-2016, we can use a mean for the other two years.



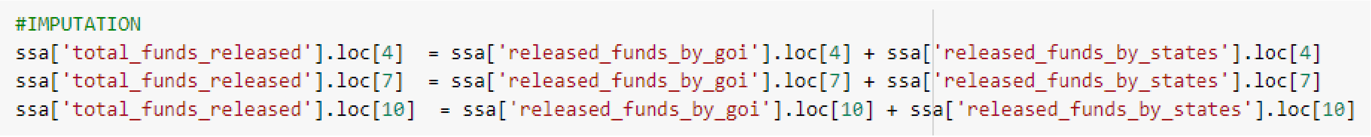
```
ssa['released_funds_by_states'].loc[4] =   ssa[ssa.state == "Chhattisgarh" ]["released_funds_by_states"].mean()
ssa['released_funds_by_states'].loc[7] =   ssa[ssa.state == "Haryana" ]["released_funds_by_states"].mean()
ssa['released_funds_by_states'].loc[10] =   ssa[ssa.state == "Karnataka" ]["released_funds_by_states"].mean()
```

This code assigns a new value to the corresponding row of the 'released_funds_by_states' column. The new value is calculated as the mean of the 'released_funds_by_states' column for all rows where the 'state' column equals that particular state. By this method we impute missing values in the 'released_funds_by_states' column for the state.

3.9. REASON AND IMPUTING MISSING VALUE OF total_funds

DATA IS MAR

total_funds_released = released_funds_by_goi + released_funds_by_states



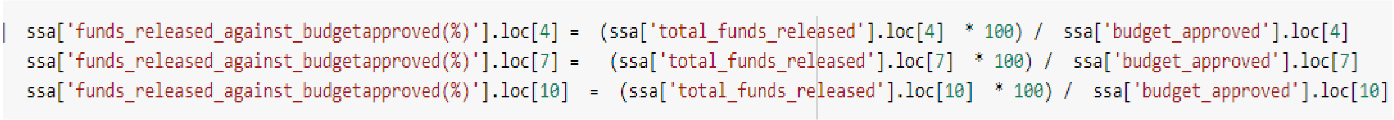
```
#IMPUTATION
ssa['total_funds_released'].loc[4]  = ssa['released_funds_by_goi'].loc[4] + ssa['released_funds_by_states'].loc[4]
ssa['total_funds_released'].loc[7]  = ssa['released_funds_by_goi'].loc[7] + ssa['released_funds_by_states'].loc[7]
ssa['total_funds_released'].loc[10]  = ssa['released_funds_by_goi'].loc[10] + ssa['released_funds_by_states'].loc[10]
```

This code assigns a new value to the corresponding row of the 'released_funds_by_states' column. The new value is calculated as the mean of the 'released_funds_by_states' column for all rows where the 'state' column equals that particular state. By this method we impute missing values in the 'released_funds_by_states' column for the state.

3.10. REASON AND IMPUTING MISSING VALUE OF funds_released_against_budgetapproved

DATA IS MISSING AT MAR

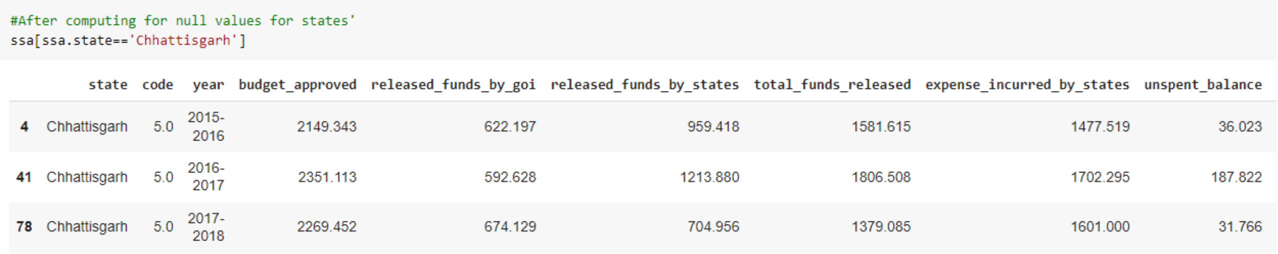
funds_released_against_budget_approved = (total_funds_released/budget_approved) × 100



```
ssa['funds_released_against_budgetapproved(%)'].loc[4] =  (ssa['total_funds_released'].loc[4]  * 100) /  ssa['budget_approved'].loc[4]
ssa['funds_released_against_budgetapproved(%)'].loc[7] =   (ssa['total_funds_released'].loc[7]  * 100) /  ssa['budget_approved'].loc[7]
ssa['funds_released_against_budgetapproved(%)'].loc[10]  =  (ssa['total_funds_released'].loc[10]  * 100) /  ssa['budget_approved'].loc[10]
```

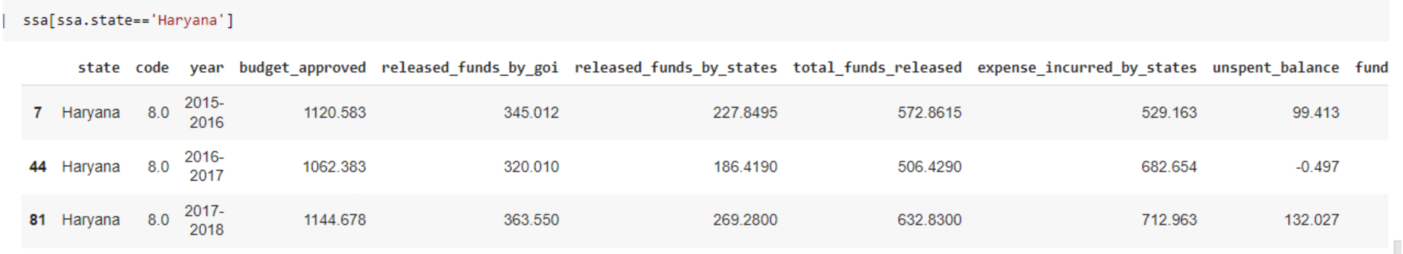
Using the above relation, we have imputed the missing values for funds_released_against_budgetapproved.

3.11. DISPLAYING VALUES AFTER COMPUTING



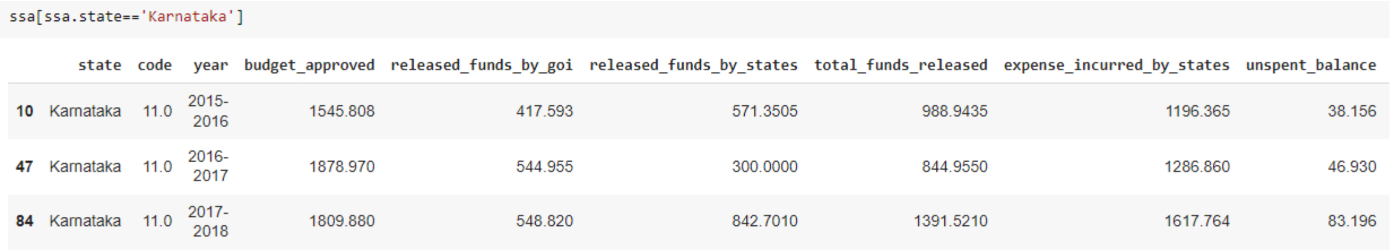
```
#After computing for null values for states'
ssa[ssa.state=='Chhattisgarh']
```

	state	code	year	budget_approved	released_funds_by_goi	released_funds_by_states	total_funds_released	expense_incurred_by_states	unspent_balance
4	Chhattisgarh	5.0	2015-2016	2149.343	622.197	959.418	1581.615	1477.519	36.023
41	Chhattisgarh	5.0	2016-2017	2351.113	592.628	1213.880	1806.508	1702.295	187.822
78	Chhattisgarh	5.0	2017-2018	2269.452	674.129	704.956	1379.085	1601.000	31.766



```
ssa[ssa.state=='Haryana']
```

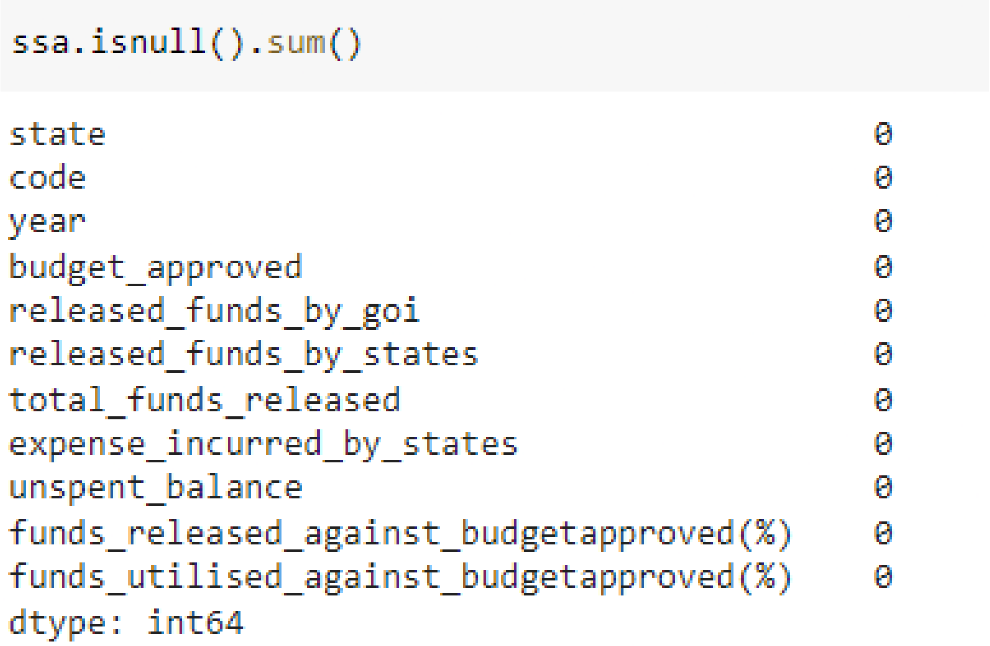
	state	code	year	budget_approved	released_funds_by_goi	released_funds_by_states	total_funds_released	expense_incurred_by_states	unspent_balance	fund
7	Haryana	8.0	2015-2016	1120.583	345.012	227.8495	572.8615	529.163	99.413	
44	Haryana	8.0	2016-2017	1062.383	320.010	186.4190	506.4290	682.654	-0.497	
81	Haryana	8.0	2017-2018	1144.678	363.550	269.2800	632.8300	712.963	132.027	



```
ssa[ssa.state=='Karnataka']
```

	state	code	year	budget_approved	released_funds_by_goi	released_funds_by_states	total_funds_released	expense_incurred_by_states	unspent_balance
10	Karnataka	11.0	2015-2016	1545.808	417.593	571.3505	988.9435	1196.365	38.156
47	Karnataka	11.0	2016-2017	1878.970	544.955	300.0000	844.9550	1286.860	46.930
84	Karnataka	11.0	2017-2018	1809.880	548.820	842.7010	1391.5210	1617.764	83.196

To verify that all missing values have been filled, we can check the count of missing values in the 'released_funds_by_states' column using the isna() method again. If there are no missing values, it means that all missing values have been successfully filled using the imputation method.



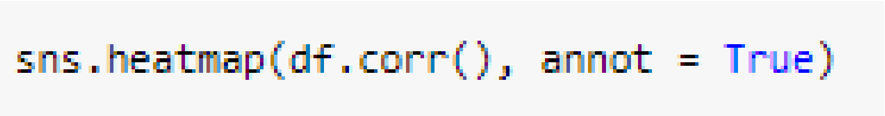
```
ssa.isnull().sum()
```

```
state                                       0
code                                        0
year                                        0
budget_approved                             0
released_funds_by_goi                       0
released_funds_by_states                    0
total_funds_released                        0
expense_incurred_by_states                  0
unspent_balance                             0
funds_released_against_budgetapproved(%)    0
funds_utilised_against_budgetapproved(%)    0
dtype: int64
```

The dataset has been cleaned and pre-processed, the dataset is ready for exploratory data analysis or any subsequent statistical modelling or analysis. This ensures that any insights or conclusions drawn from the data are valid and reliable, and can be used to inform decisions or further research.

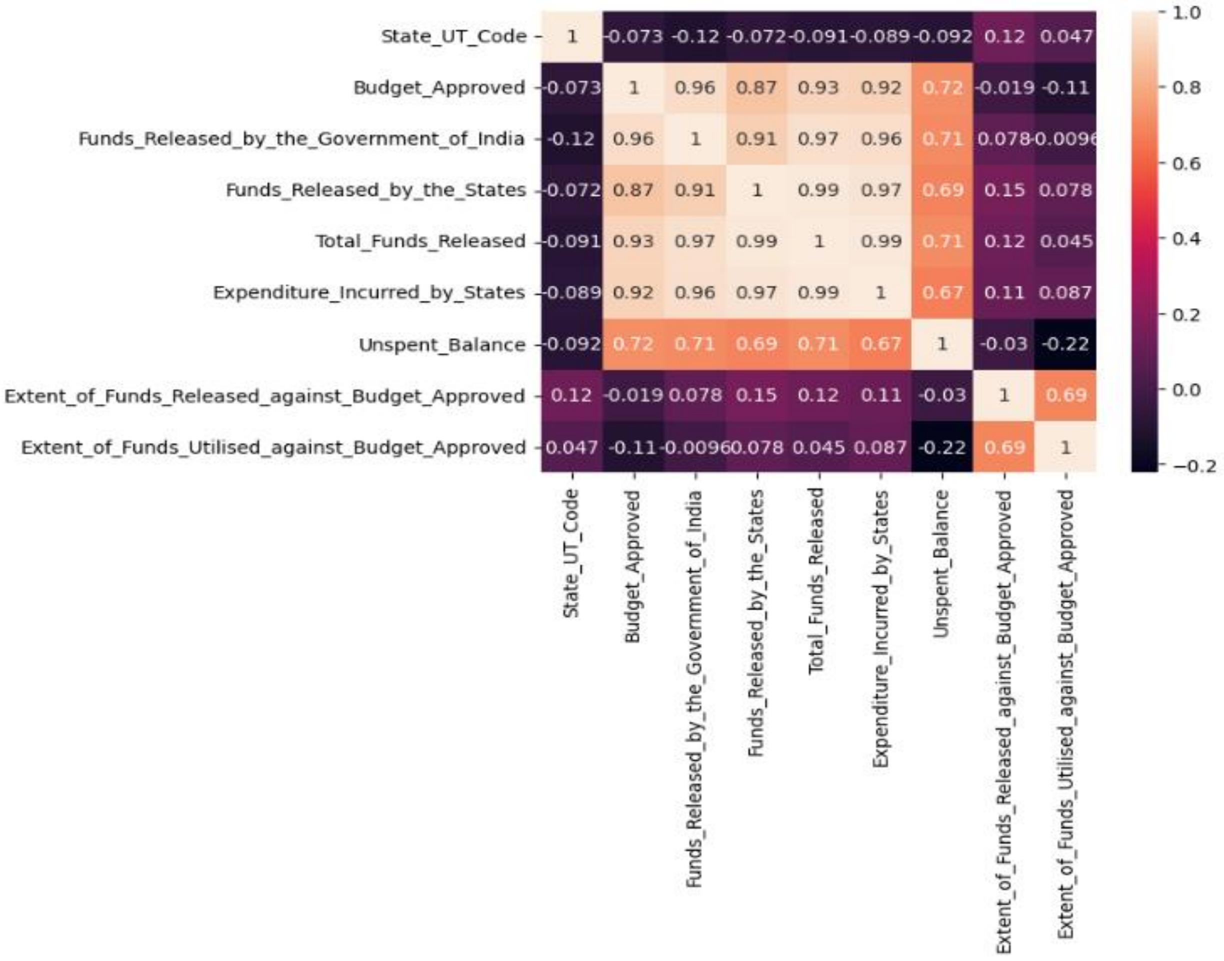
4. DATA VISUALIZATION

4.1. HEATMAP



```
sns.heatmap(df.corr(), annot = True)
```

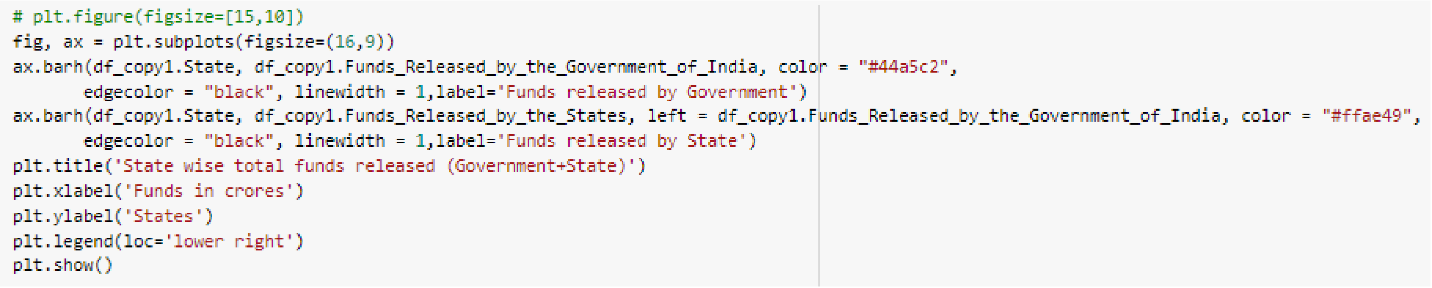
```
<Axes: >
```



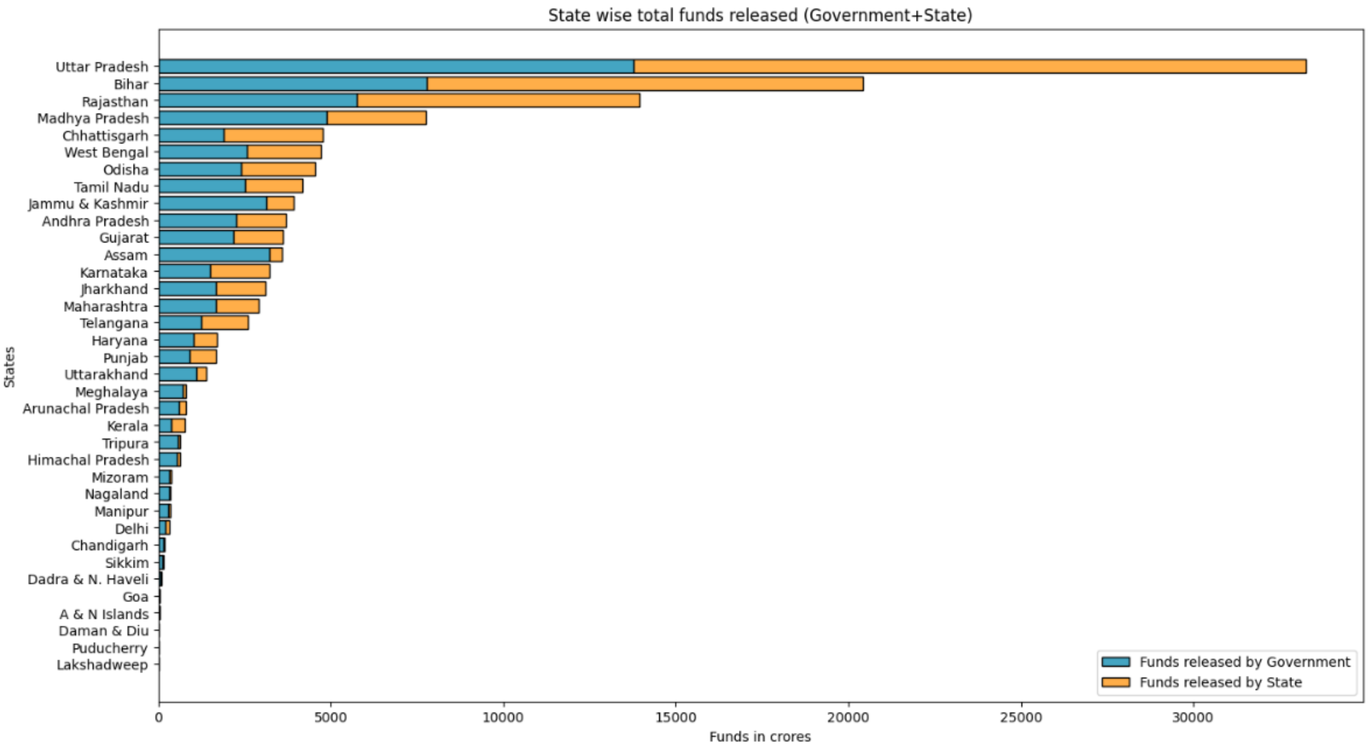
Heatmap shows the correlation between all the feature. Correlation ranges from -1 to 1 where 1 denotes high correlation between two feature, -1 means it is oppositely correlated, and 0 means no correlation between features.

Observation : Here we can see that many columns are highly correlated with each other.

4.2. STACKED BARCHART



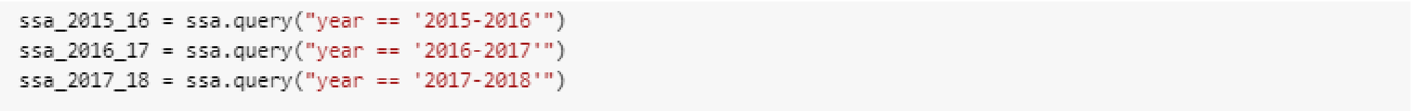
```
# plt.figure(figsize=[15,10])
fig, ax = plt.subplots(figsize=(16,9))
ax.barh(df_copy1.State, df_copy1.Funds_Released_by_the_Government_of_India, color = "#44a5c2",
       edgecolor = "black", linewidth = 1,label='Funds released by Government')
ax.barh(df_copy1.State, df_copy1.Funds_Released_by_the_States, left = df_copy1.Funds_Released_by_the_Government_of_India, color = "#ffae49",
       edgecolor = "black", linewidth = 1,label='Funds released by State')
plt.title('State wise total funds released (Government+State)')
plt.xlabel('Funds in crores')
plt.ylabel('States')
plt.legend(loc='lower right')
plt.show()
```



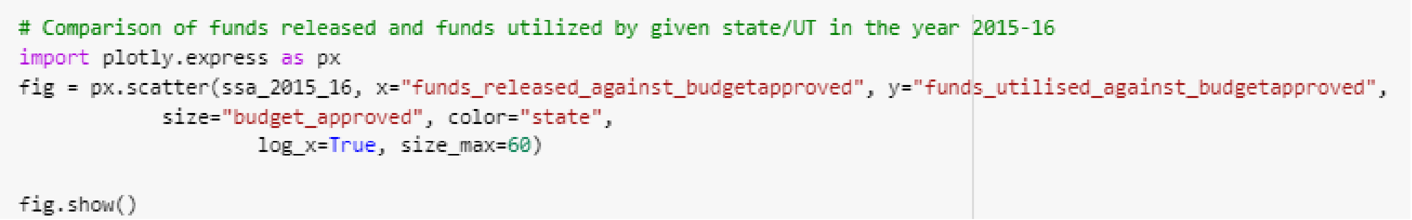
Here, in stacked bar chart, the length of each bar represents the total funds released to the respective states, and the segments within the bar represent the funds released by state and government.

OBSERVATION : We can clear see the distribution of the funds among each state.

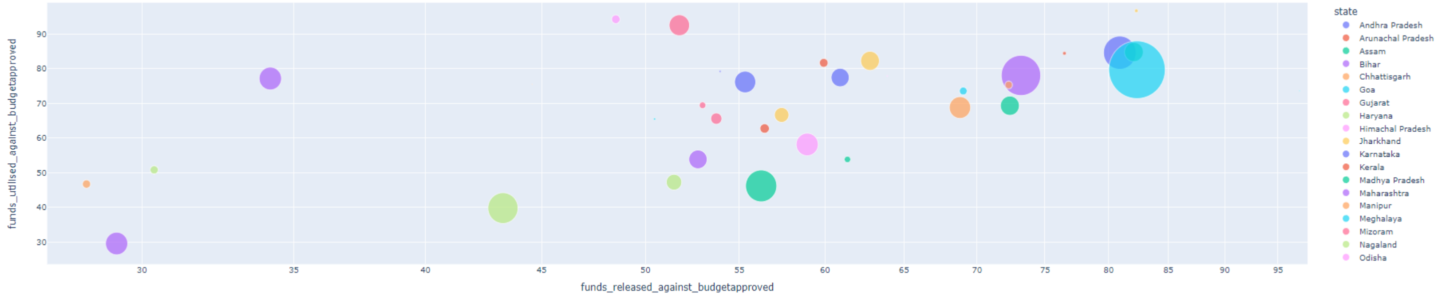
4.3. WEIGHTED SCATTER PLOT



```
ssa_2015_16 = ssa.query("year == '2015-2016'")
ssa_2016_17 = ssa.query("year == '2016-2017'")
ssa_2017_18 = ssa.query("year == '2017-2018'")
```



```
# Comparison of funds released and funds utilized by given state/UT in the year 2015-16
import plotly.express as px
fig = px.scatter(ssa_2015_16, x="funds_released_against_budgetapproved", y="funds_utilised_against_budgetapproved",
            size="budget_approved", color="state",
                    log_x=True, size_max=60)

fig.show()
```



Here in weighted scatter plot, scatter plot is of funds released vs utilised against the budget. Weights in the scatter is of the budget approved.

OBSERVATION : We can see that as funds released increases the utilization of that funds also increases.

4.4. VISUALIZATION DASHBOARD

A visualization dashboard is a collection of visualizations and interactive components that provide a high-level overview of key performance indicators (KPIs).

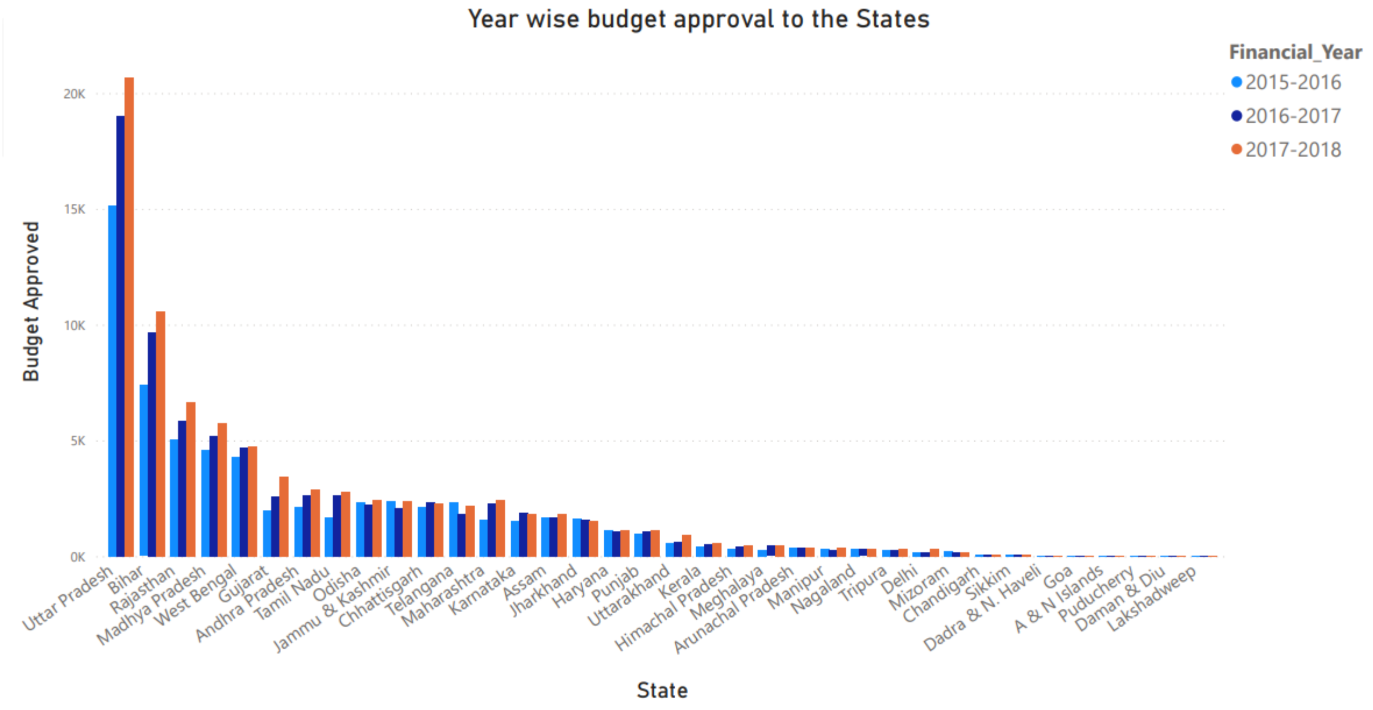
A visualization dashboard typically includes a variety of visualizations such as charts, graphs, tables, and maps, as well as interactive components such as filters, drop-down menus, and sliders. The visualizations and interactive components are designed to be easy to read and interpret, allowing users to quickly identify important trends and patterns in the data.

Visualization dashboards can be created using various tools and software, including Excel, Tableau, Power BI, and other data visualization software.

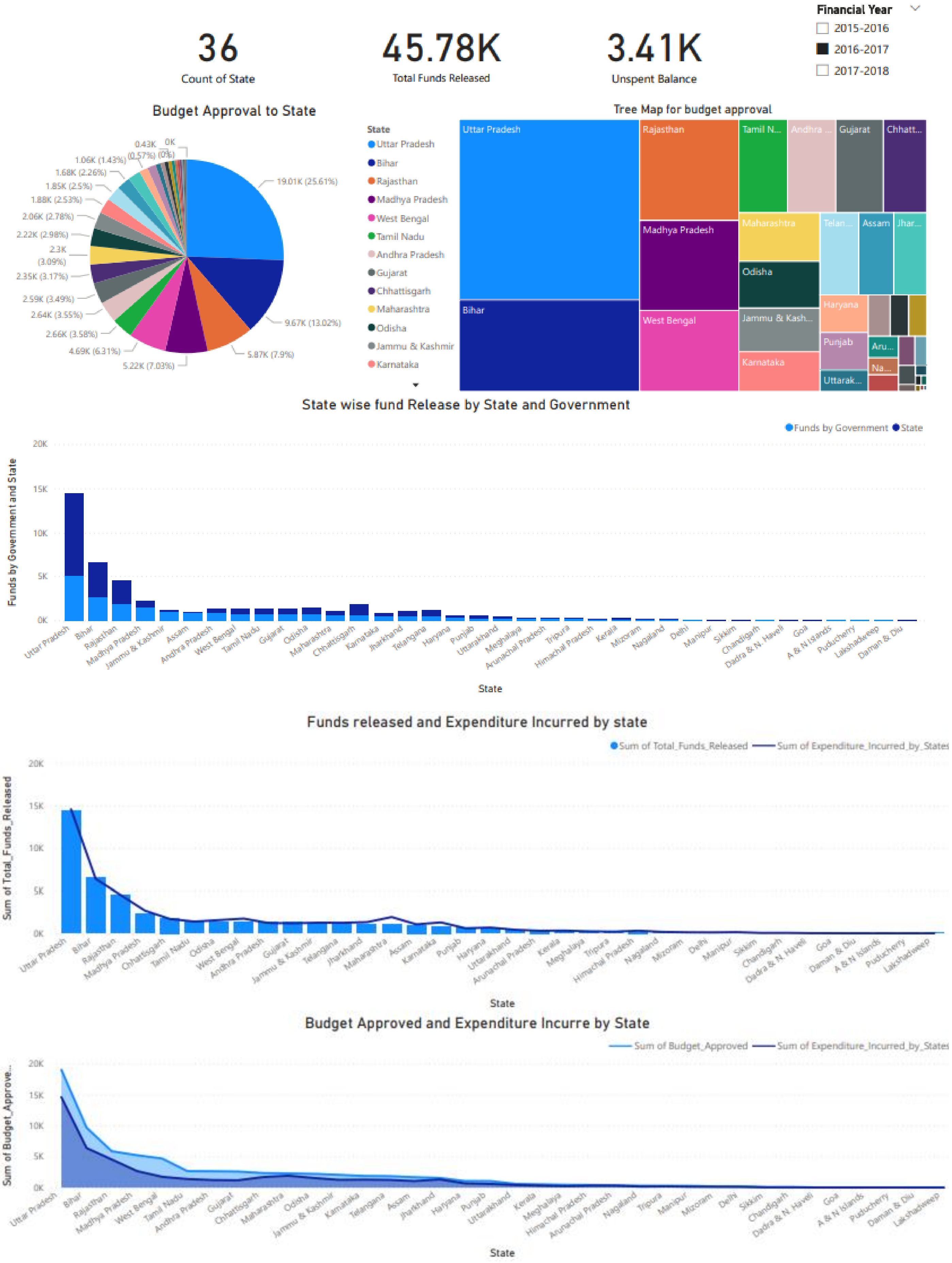
In here, we have used Power BI to make the dashboard.

Power BI is a business analytics service provided by Microsoft that allows users to connect, analyse, and visualize data from various sources. Power BI is a powerful tool for creating interactive dashboards, reports, and visualizations that enable users to gain insights from data and make informed decisions.

4.4.1. STATIC VISUALS :



4.4.2. DYNAMIC DASHBOARD:



36
Count of State

45.78K
Total Funds Released

3.41K
Unspent Balance

Financial Year
□ 2015-2016
■ 2016-2017
□ 2017-2018

Financial Year

□ 2015-2016

□ 2016-2017

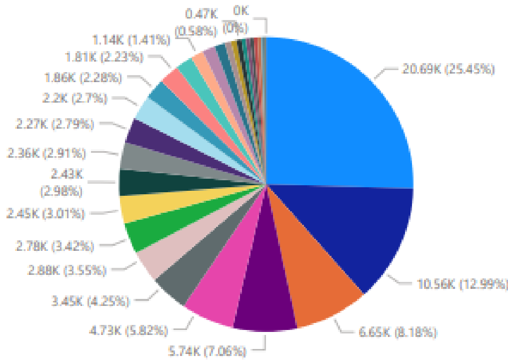
■ 2017-2018

36
Count of State

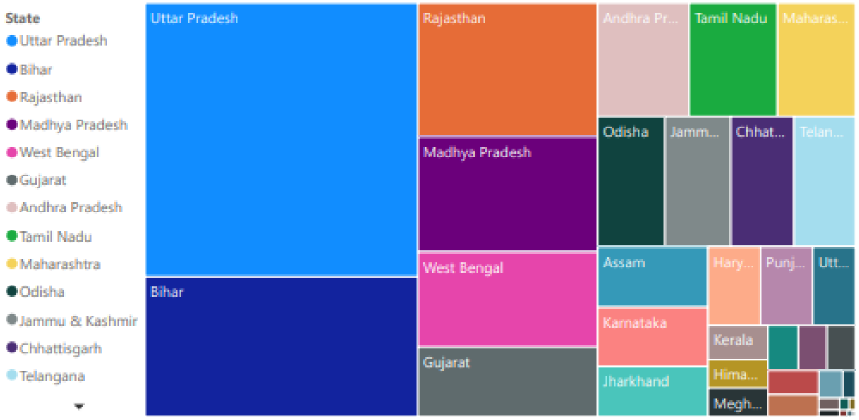
43.27K
Total Funds Released

5.45K
Unspent Balance

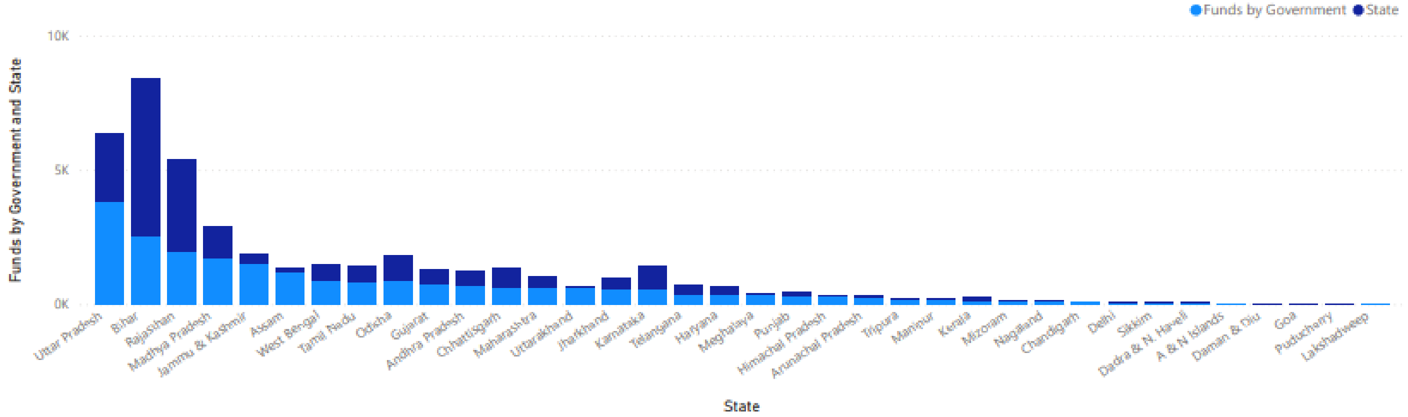
Budget Approval to State



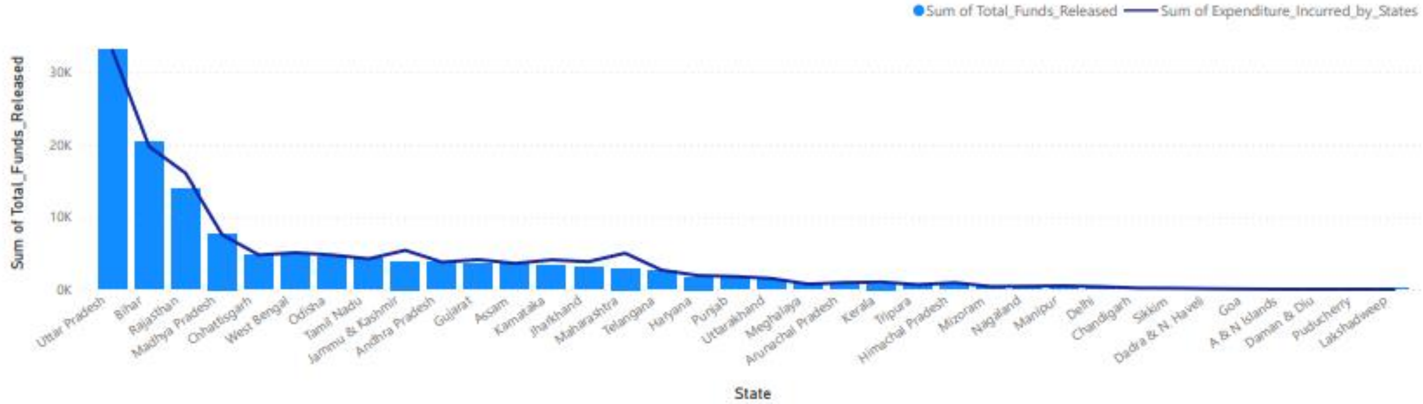
Tree Map for budget approval



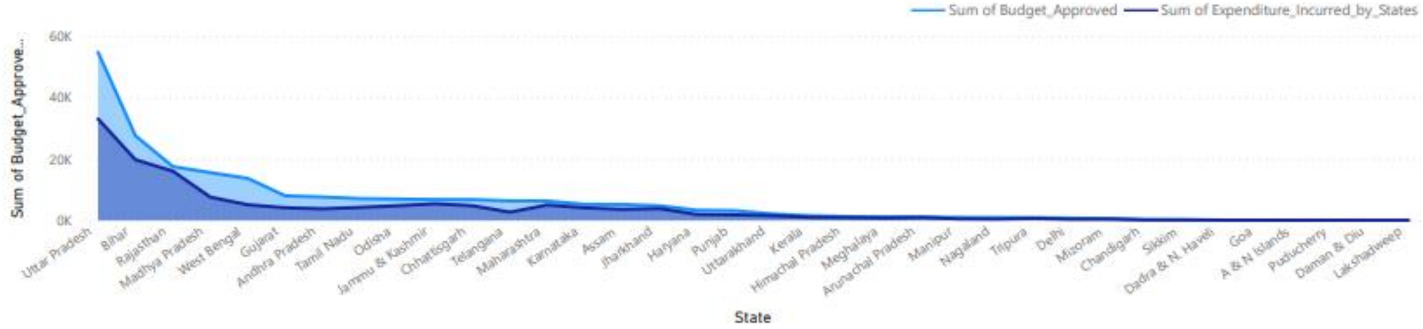
State wise fund Release by State and Government



Funds released and Expenditure Incurred by state



Budget Approved and Expenditure Incurre by State



5. DATA PREDICTION

5.1. OBJECTIVES

OBJECTIVE 1 - We will train data for 2 years(2015-16 to 2016-17) and will take 'approved_budget' as test data for 2017-18 year and decide which ML model will work efficiently

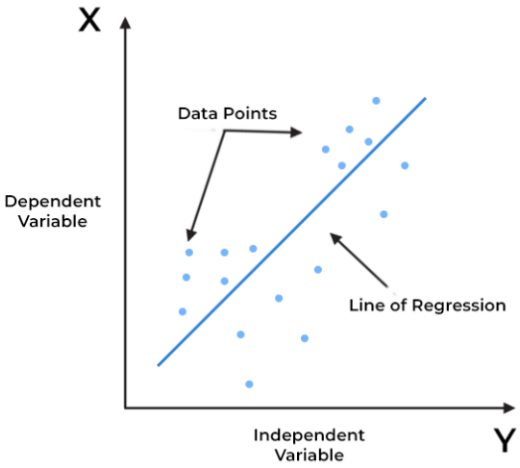
OBJECTIVE 2 - We will take ML model from obejective 1 and train data for 3 years(2015-16 to 2017-18) and will predict 'approved_budget' for 2018-19 financial year

5.2. MODEL USED FOR THE PREDICTION

1. Linear Regression:

   Linear regression is a statistical method used to analyze the relationship between a dependent variable (usually denoted as "y") and one or more independent variables (usually denoted as "x"). The goal of linear regression is to find the linear relationship between the variables, which can be used to make predictions and understand the strength and direction of the relationship.

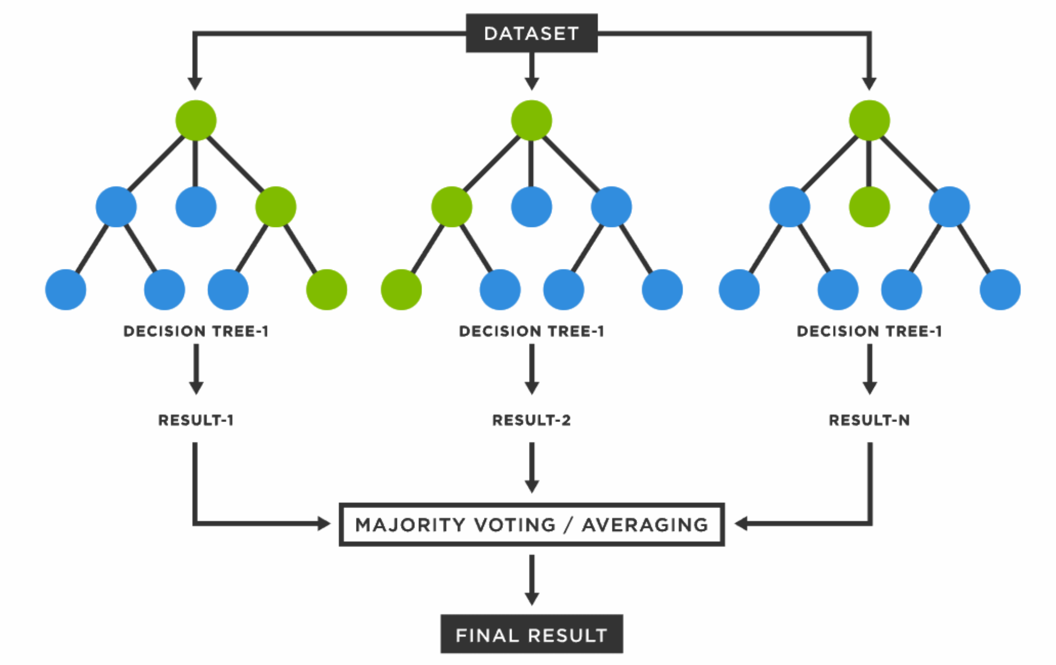
   In linear regression, the relationship between the variables is represented by a straight line equation in the form of y = mx + b, where "m" is the slope of the line (representing the change in y for every one-unit change in x) and "b" is the intercept (representing the value of y when x is equal to zero). The slope and intercept are estimated from the data using a method called least squares regression.



2. Random Forest:

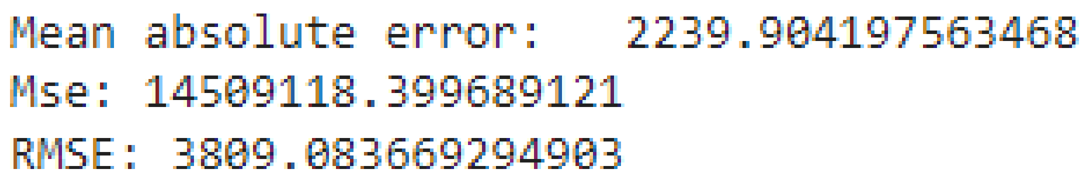
Random forest is a machine learning algorithm that is used for classification, regression, and other tasks that involve supervised learning. It is an ensemble method that combines multiple decision trees to improve accuracy and reduce overfitting.

In a random forest, a large number of decision trees are generated, with each tree trained on a randomly selected subset of the data and a randomly selected subset of the features. Each tree makes a prediction, and the final prediction is made by taking the majority vote of all the trees. This process helps to reduce the risk of overfitting and increase the accuracy of the predictions.

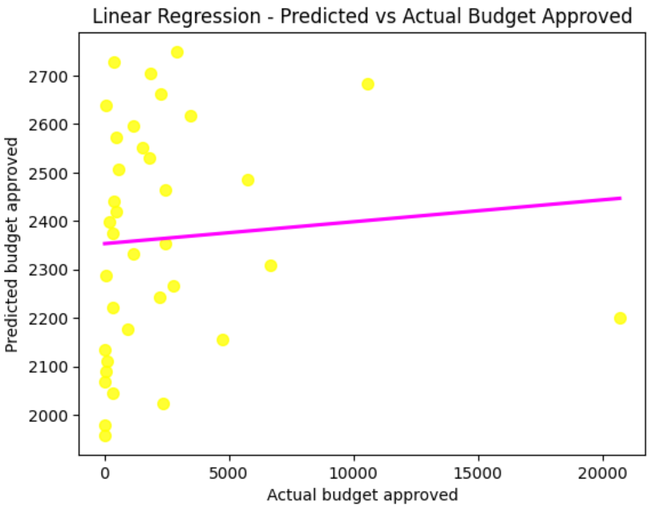


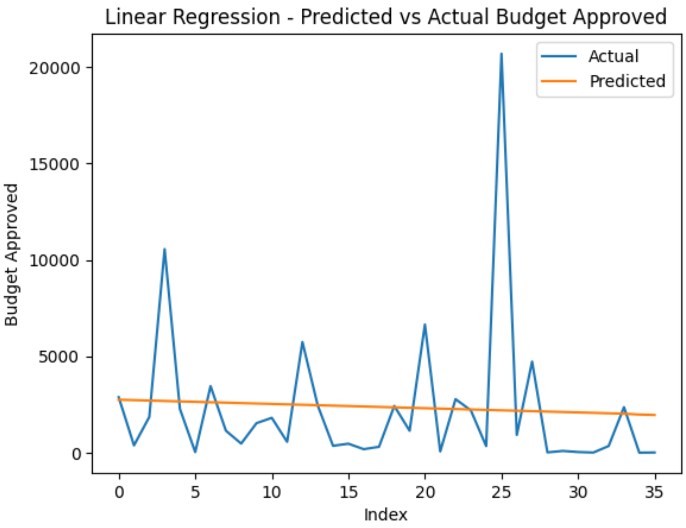
5.3. OBJECTIVE 1 USING LINEAR REGRESSION

Using Linear Regression model for prediction. We will find the error factors like MAE, MSE, RMSE and after that we will conclude the performance of the model.



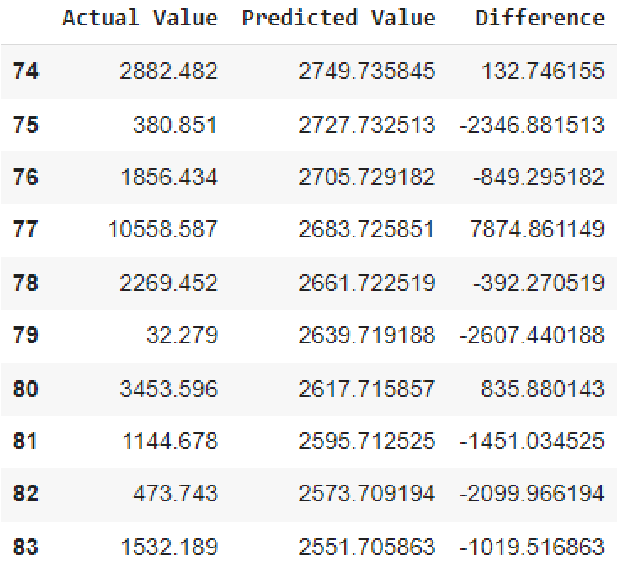
```
Mean absolute error:   2239.904197563468
Mse: 14509118.399689121
RMSE: 3809.083669294903
```





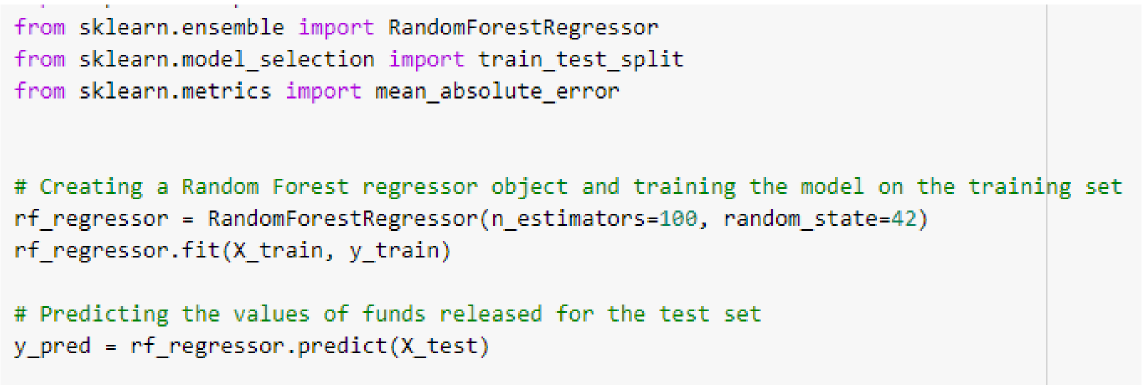
OBSERVATION : We can clearly see that the model is unfit as the Error factors are too high. Also, we can see the graph where there is huge difference between Actual and Predicted line.

Here below is difference of the actual and predicted value.

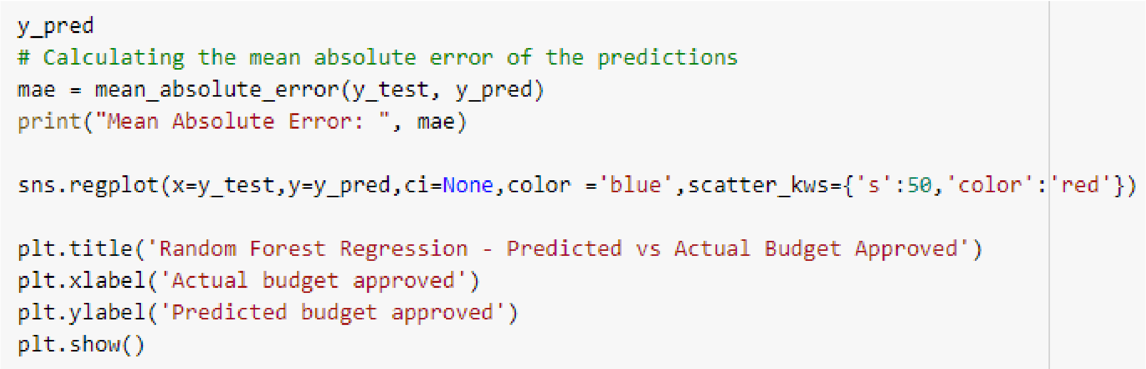


	Actual Value	Predicted Value	Difference
74	2882.482	2749.735845	132.746155
75	380.851	2727.732513	-2346.881513
76	1856.434	2705.729182	-849.295182
77	10558.587	2683.725851	7874.861149
78	2269.452	2661.722519	-392.270519
79	32.279	2639.719188	-2607.440188
80	3453.596	2617.715857	835.880143
81	1144.678	2595.712525	-1451.034525
82	473.743	2573.709194	-2099.966194
83	1532.189	2551.705863	-1019.516863

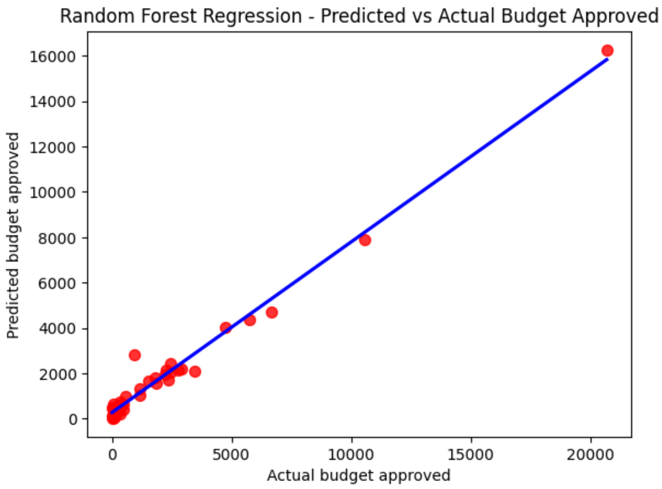
5.4. OBJECTIVE 1 USING RANDOM FOREST

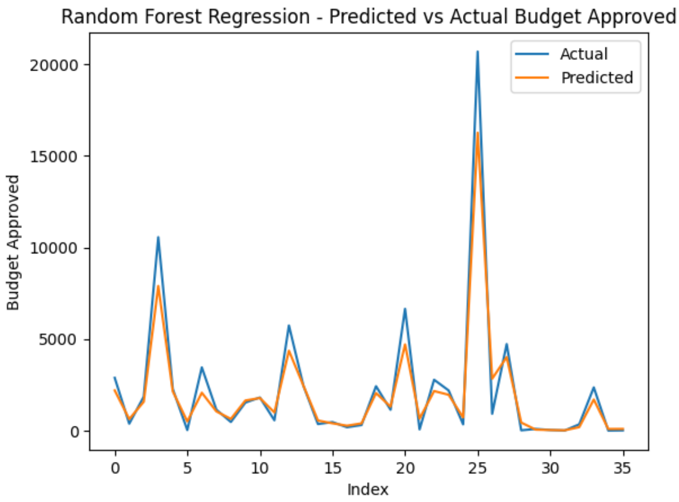


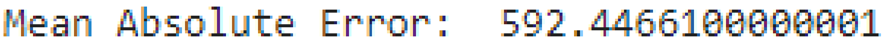
```
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error


# Creating a Random Forest regressor object and training the model on the training set
rf_regressor = RandomForestRegressor(n_estimators=100, random_state=42)
rf_regressor.fit(X_train, y_train)

# Predicting the values of funds released for the test set
y_pred = rf_regressor.predict(X_test)
```



```
y_pred
# Calculating the mean absolute error of the predictions
mae = mean_absolute_error(y_test, y_pred)
print("Mean Absolute Error: ", mae)

sns.regplot(x=y_test,y=y_pred,ci=None,color ='blue',scatter_kws={'s':50,'color':'red'})

plt.title('Random Forest Regression - Predicted vs Actual Budget Approved')
plt.xlabel('Actual budget approved')
plt.ylabel('Predicted budget approved')
plt.show()
```



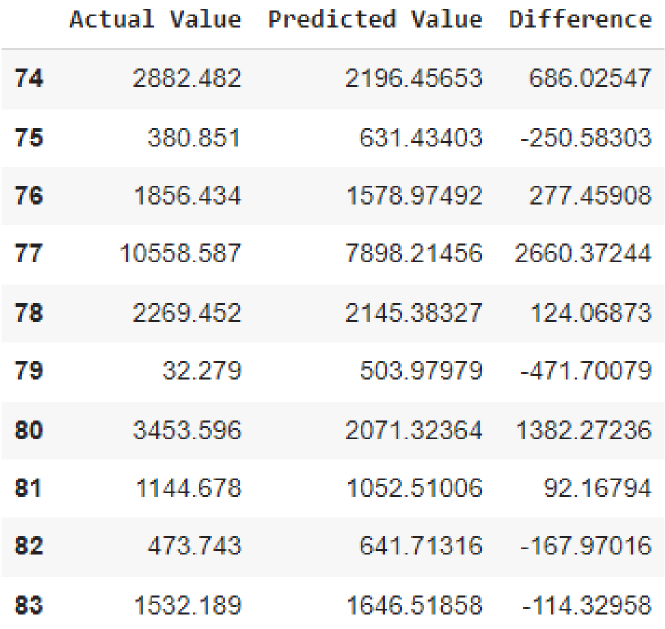




```
Mean Absolute Error:  592.4466100000001
```

OBSERVATION:

We can see that the MAE has come down. Also in the graph, actual and predicted line are almost same. So we can conclude that the random forest is better model than the linear regression.



	Actual Value	Predicted Value	Difference
74	2882.482	2196.45653	686.02547
75	380.851	631.43403	-250.58303
76	1856.434	1578.97492	277.45908
77	10558.587	7898.21456	2660.37244
78	2269.452	2145.38327	124.06873
79	32.279	503.97979	-471.70079
80	3453.596	2071.32364	1382.27236
81	1144.678	1052.51006	92.16794
82	473.743	641.71316	-167.97016
83	1532.189	1646.51858	-114.32958

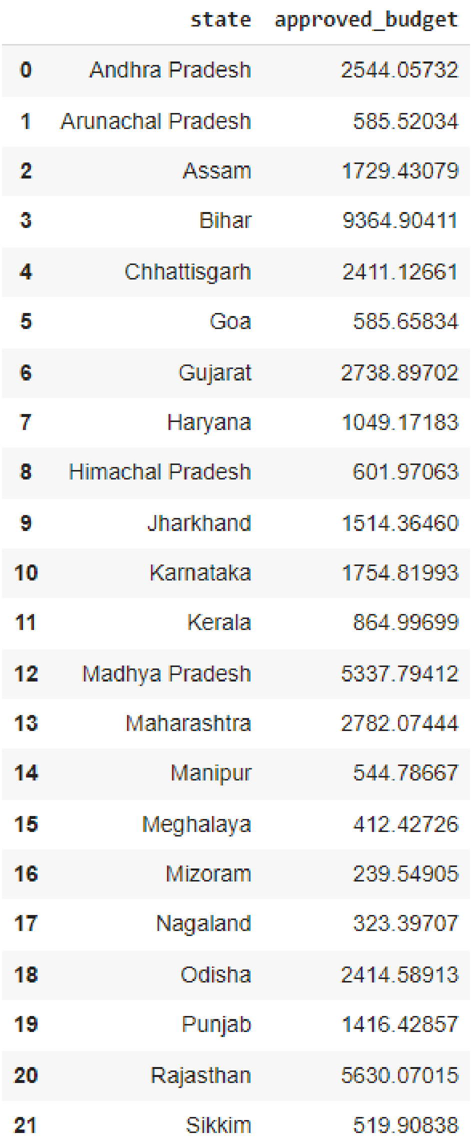
5.5. OBJECTIVE 2 USING RANDOM FOREST

Here instead of giving all the features, we have given only states and ut_code as other features comes after the budget gets decided. So we cannot give the that features in the test data.

In the objective 1 we have seen that random forest is the better model so we are apply that model only in this objective.

Using the random forest, data of year 2015-17 as training, we have predicted the values of budgets of each states for the 2018 year.

Below are the value of that



	state	approved_budget
0	Andhra Pradesh	2544.05732
1	Arunachal Pradesh	585.52034
2	Assam	1729.43079
3	Bihar	9364.90411
4	Chhattisgarh	2411.12661
5	Goa	585.65834
6	Gujarat	2738.89702
7	Haryana	1049.17183
8	Himachal Pradesh	601.97063
9	Jharkhand	1514.36460
10	Karnataka	1754.81993
11	Kerala	864.99699
12	Madhya Pradesh	5337.79412
13	Maharashtra	2782.07444
14	Manipur	544.78667
15	Meghalaya	412.42726
16	Mizoram	239.54905
17	Nagaland	323.39707
18	Odisha	2414.58913
19	Punjab	1416.42857
20	Rajasthan	5630.07015
21	Sikkim	519.90838

CURRICULUM VITAE

Group 1

Muskan Khare(Student ID - 202218037)

“I am Muskan Khare from MSc Data Science(2022-24). I am from Lucknow, Uttar Pradesh. I completed my schooling from Lucknow and my bachelors in BSc(Honours) mathematics from Jesus and Mary College, University of Delhi. With a background in Maths, my interests involve calculus, linear algebra and statistics. I have experience coding in a variety of programming languages including Python, R, Matlab, Java, and SQL. Each of these languages has its own unique features and areas of application, which allows me to work on a wide range of projects and tasks.”

LinkedIn Profile Link: https://www.linkedin.com/in/muskan-khare-6a1872189

Riya Kumari(Student ID - 202218049)

“I am Riya Kumari from MSc Data Science(2022-2024). I am from Ranchi, Jharkhand.

I did my BSc in Statistics from St. Xavier's College , Ranchi . As a background in statistics, I love studying it and working around it. My skills include Data Visualisation, Data Analysis and Machine Learning modelling and some statistical tests. My hobbies include travelling, listening to music and learning new things . I consider myself as a dedicated and hardworking student who is always in a hunger to do the best.”

LinkedIn Profile Link: https://www.linkedin.com/in/riyakumari19

Dhruv Solanki(Student ID - 202218053)

My name is Dhruv Solanki, and I am from MSc Data Science (2022 batch). My hometown is Surat, Gujarat. In my free time, I enjoy listening to music. I also like to talk about experiences in different fields with different people.

I have done my BE-IT from D.Y. Patil, Pune, and after that, I have a 3-month internship as a Data Analyst in a company.

As I have worked as a Data Analyst, I know how crucial EDA is for the one who wants their career or is interested in Data related field. And in the internship, the EDA part excited me the most. So these were the things that motivated me to enroll in this course.

LinkedIn Profile Link: www.linkedin.com/in/dhruvsolanki5555a

Github Link: https://github.com/Ejru5

Chinmaya Pandey(Student ID - 202218054)

“I am Chinmaya Pandey, M.Sc. in Data Science (2022-2024).I am from Ujjain, Madhya Pradesh. My Bachelor's degree is in B.Sc. (Hons.) Electronics and Mathematics from IEHE, Bhopal. My academic background in Mathematics has provided me with a strong foundation and a profound comprehension of mathematical concepts and problem-solving skills. My areas of interest include calculus, linear algebra, abstract algebra and topology. Additionally, I have hands-on experience with Python, R and SQL programming languages, which I have used to solve complex mathematical problems. I have expertise in Statistical analysis, Data Visualization, and Machine Learning techniques. Pursuing data science has also helped me to develop a profound understanding of Big Data and NLP. Apart from academics, I love exploring new places, trying out different cuisines, and reading fantasy and sci-fi books.I consider myself to have good communication skills and possess leadership qualities.”

LinkedIn Profile Link: https://www.linkedin.com/in/chinmaya-pandey-60211b264

Jatan Sahu(student ID - 202218061)

“Hello, I am Jatan Sahu, currently pursuing a Master's degree in Data Science with an expected graduation date of 2024. I completed my Bachelor's degree in Computer Science from St. Aloysius College in Jabalpur, Madhya Pradesh. I possess proficient programming skills in Python and SQL, as well as a sound understanding of Big Data, Machine Learning, and some aspects of NLP. I consider myself proficient in several essential skills required for a Data Scientist, including programming, Machine Learning, Big Data, business acumen, data visualisation, communication, and problem-solving. I am confident in my ability to work with large and complex datasets, develop predictive models, and communicate insights effectively to stakeholders.”

LinkedIn Profile Link: https://www.linkedin.com/in/jatan-sahu-2a06381b1/

Github Link: https://github.com/Jatansahu