The Godfather of Talent | Futurense

Futurense

Democratizing Tech Talent
to deliver impact at scale

The Godfather of Talent | Futurense

INDIAN HEALTHCARE ANALYSIS

FARYAR MEMON - FT641

The Godfather of Talent | Futurense

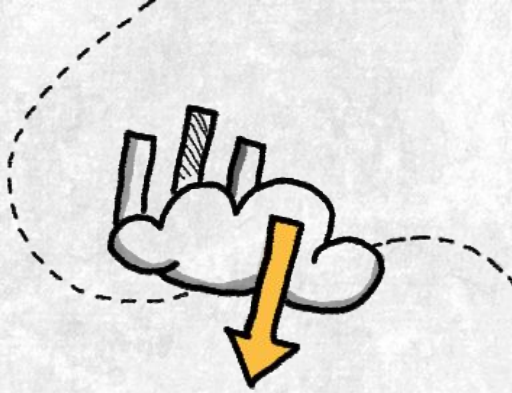
Overview:

- **Brief Description:** This project aims **to analyze the healthcare infrastructure in India** using data on **census, housing, hospitals, and other relevant factors**. With a wide range of healthcare facilities, from globally acclaimed hospitals to those providing unacceptably low quality care, issues such as a shortage of hospital beds can have serious consequences, as seen during the COVID-19 pandemic.

- **Business Problem:** The project's objective is to **identify gaps and areas of improvement** in the healthcare infrastructure of different states in India. This information can be used to develop strategies and policies to improve healthcare access and outcomes in these states.

- **Solution Approach:** The solution approach involves preprocessing the data using classes and functions, mapping relevant features using GeoPandas, and analyzing the data using various visualization techniques. The goal is to provide insights and recommendations for improving healthcare in the country, including developing strategies and policies.

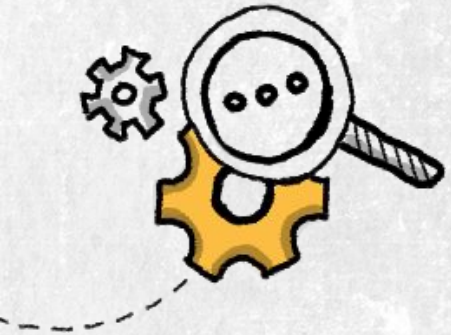
The Godfather of Talent | Futurense



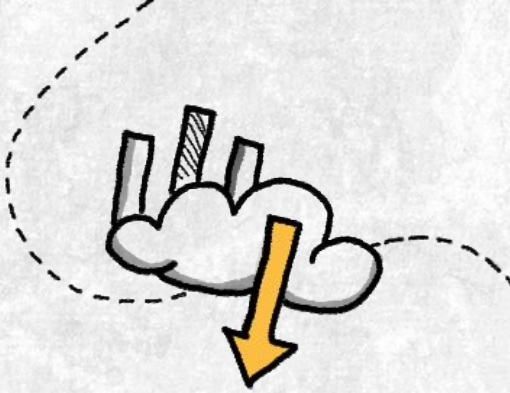
Tools and Technologies Used:

PYTHON

MySQL

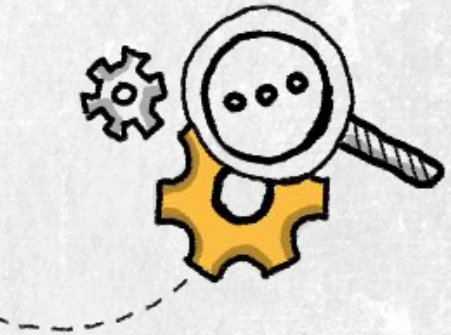


The Godfather of Talent | Futurense



Libraries/Packages/Features Used:

1. **NumPy**: Uses for mathematical calculations.
2. **Pandas**: Used to manipulate and transform the data in the tables of the database, specifically for features such as vectorization.
3. **GeoPandas**: Used to create GeoDataFrames and GeoDataSeries used to create map for geospatial analysis.
4. **Matplotlib**: Used to create basic plots and was frequently used to customize other visualizations created using Seaborn.
5. **Seaborn**: Used for data visualization mainly because of its easy syntax and available themes and palettes.
6. **Classes (OOPs)**: Created for reuse of functions, the dataframes were used as an instance variable, thereby unique to each object ex., census, housing, hospitals, etc.



NumPy

pandas

GeoPandas

matplotlib

seaborn

The Godfather of Talent | Futurense

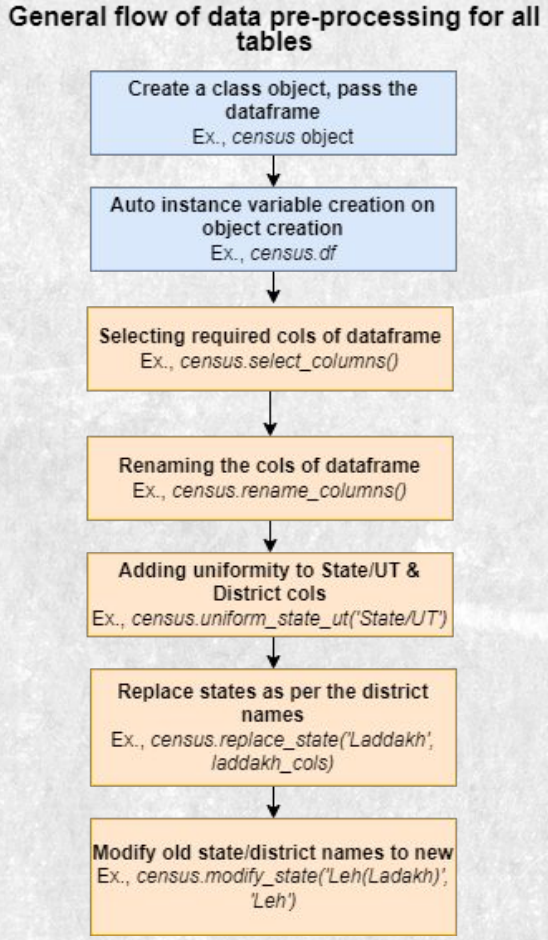
Headline Statistics:

- IDE/Code Editor Used: **VS Code**
- Function/Complexity breakdown:
  - Simple Implementations
  - **Intermediate Implementations**
  - Complex Implementations

```python
class Preprocessing:
    def __init__(self, df):
        self.df = df.copy()

    def dfInfo(self):
        '''Prints basic information such as shape, columns with missing values, etc about any input dataframe.'''
        print(self.df.shape)
        print('='*50)
        print(self.df.info())
        print('='*50)
        # null_cols = [col for col in df.columns if df[col].isnull().sum() != 0]
        # print(f'Columns with missing values: {null_cols}')

    def select_columns(self, columns):
        '''Creates a subset dataframe of the original dataframe as per the columns passed to this function.'''
        self.df = self.df[columns]
        # return self.df

    def rename_columns(self, columns):
        '''Renames the columns of the dataframe as per the columns passed to it.'''
        self.df.rename(columns=columns, inplace=True)
        # return self.df

    def uniform_state_ut(self, col):
        '''Modifies and adds uniformity to State/UT column values.'''
        self.df[col] = self.df[col].str.title().replace(['And ', '& '], 'and ', regex=True)
        self.df[col] = self.df[col].str.title().replace('\*', '', regex=True)

        # return self.df

    def replace_state(self, state, districts):
        '''Changing the state names as per the district lists.'''
        self.df.loc[self.df['District'].isin(districts), 'State/UT'] = state

    def modify_state(self, prev_state, new_state):
        '''Modify the name of the state with a new name.'''
        self.df = self.df.replace(prev_state, new_state)

    def null_percent(self):
        '''Checks the null percentage for each column in the dataframe.'''
        null_percents = self.df.isnull().sum() * 100 / len(self.df)
        null_percents = null_percents.reset_index()
        null_percents.columns = ['Columns', 'Null_Percent']
        null_percents = null_percents.sort_values(by=['Null_Percent'], ascending=False)
        return null_percents

    def fill_nulls(self, total, rest_cols):
        '''Fills the null values using other columns.'''
        self.df[total] = self.df[total].fillna(self.df[rest_cols].sum(axis=1, skipna=False))
        for col in rest_cols:
            copy_cols = [c for c in rest_cols if c != col]
            self.df[col] = self.df[col].fillna(self.df[total]-self.df[copy_cols].sum(axis=1, skipna=False))
```

The Godfather of Talent | Futurense

#1 Data Pre-processing:

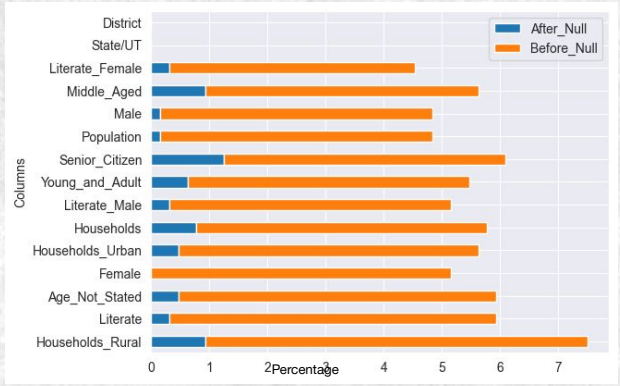



```python
class Preprocessing:
    def __init__(self, df):
        self.df = df.copy()

    def dfInfo(self):
        '''Prints basic information such as shape,
        columns with missing values, etc about any input dataframe.'''

    def select_columns(self, columns):
        '''Creates a subset dataframe of the original
        dataframe as per the columns passed to this function.'''

    def rename_columns(self, columns):
        '''Renames the columns of the dataframe as
        per the columns passed to it.'''

    def uniform_state_ut(self, col):
        '''Modifies and adds uniformity to State/UT column values.'''

    def replace_state(self, state, districts):
        '''Changing the state names as per the district lists.'''

    def modify_state(self, prev_state, new_state):
        '''Modify the name of the state with a new name.'''
```

The Godfather of Talent | Futurense

#2 Handling Missing Data:

Graph 1.



- First find the missing value percentage for each column.
- Apply fill_nulls() for each column involved:
  - Population = Male + Female
  - Literate = Literate_Male + Literate_Female
  - Population = Young_and_Adult+ Middle_Aged + Senior_Citizen + Age_Not_Stated
  - Households = Households_Rural + Households_Urban
- Check for missing value percentage after filling them.

```python
def null_percent(self):
    '''Checks the null percentage for each column in the dataframe.'''
    null_percents = self.df.isnull().sum() * 100 / len(self.df)
    null_percents = null_percents.reset_index()
    null_percents.columns = ['Columns', 'Null_Percent']
    null_percents = null_percents.sort_values(by=['Null_Percent'],
                                              ascending=False)
    return null_percents
```

```python
def fill_nulls(self, total, rest_cols):
    '''Fills the null values using other columns.'''
    self.df[total] = self.df[total].fillna(self.df[rest_cols].sum(axis=1,
                                                                  skipna=False))
    for col in rest_cols:
        copy_cols = [c for c in rest_cols if c != col]
        self.df[col] = self.df[col].fillna(self.df[total]-self.df[copy_cols].sum(axis=1,
                                                                                 skipna=False))
```

```python
# fill_df= census.df.copy()

for i in range(2):
    census.fill_nulls('Population', ['Female', 'Male'])
    census.fill_nulls('Population', ['Young_and_Adult',
                                     'Middle_Aged', 'Senior_Citizen', 'Age_Not_Stated'])
    census.fill_nulls('Literate', ['Literate_Male', 'Literate_Female'])
    census.fill_nulls('Households', ['Households_Rural', 'Households_Urban'])


after_nulls = census.null_percent()
after_nulls
```

Code Implementation

Insights:

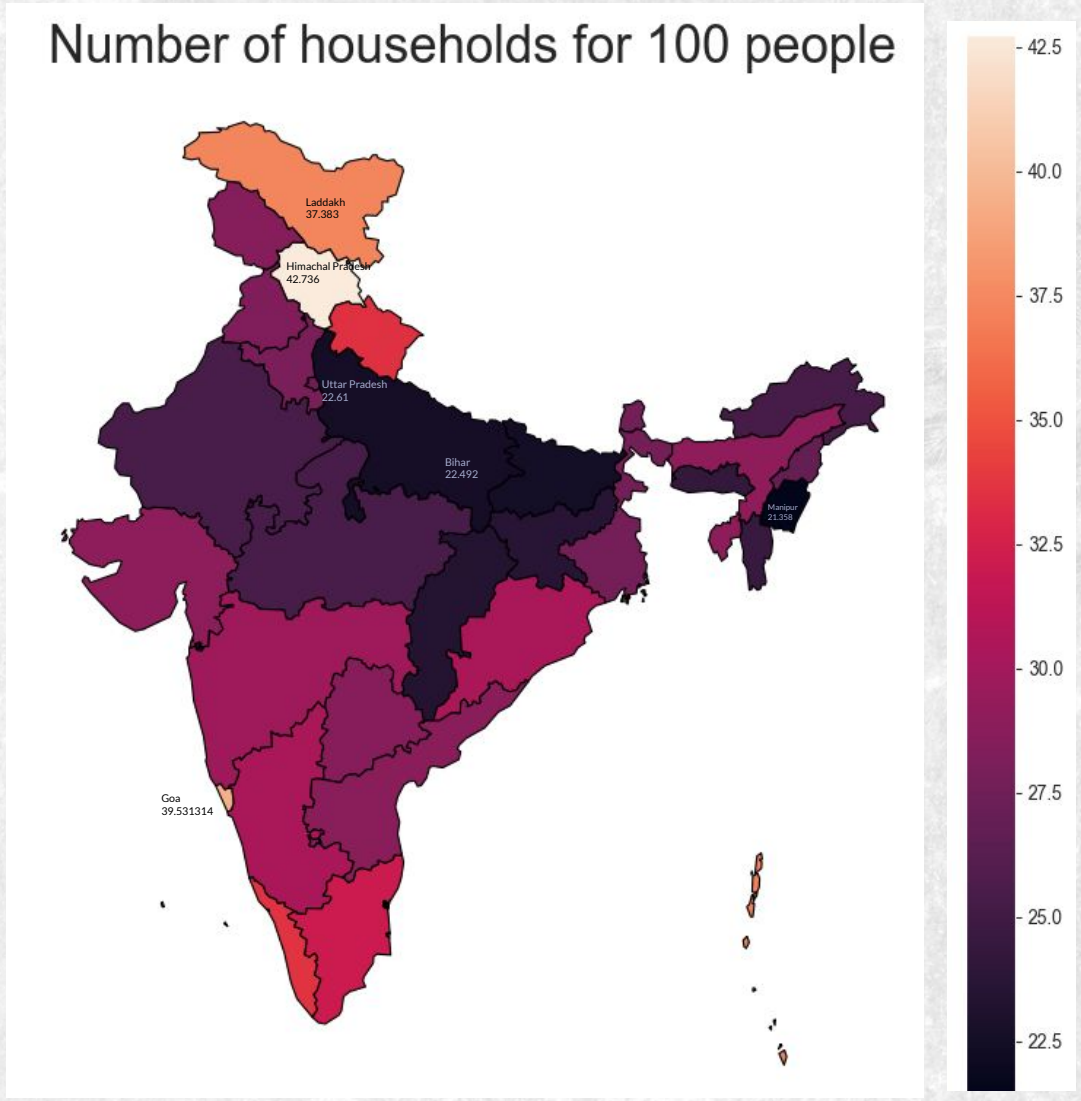
- The percentage of missing data after filling the null values is around the range of 0 to 1.5% which is relatively very low.

	Columns	Before_Null	After_Null
0	Households_Rural	6.56250	0.93750
1	Literate	5.62500	0.31250
2	Age_Not_Stated	5.46875	0.46875
3	Female	5.15625	0.00000
4	Households_Urban	5.15625	0.46875
5	Households	5.00000	0.78125
6	Literate_Male	4.84375	0.31250
7	Young_and_Adult	4.84375	0.62500
8	Senior_Citizen	4.84375	1.25000
9	Population	4.68750	0.15625
10	Male	4.68750	0.15625
11	Middle_Aged	4.68750	0.93750
12	Literate_Female	4.21875	0.31250
13	State/UT	0.00000	0.00000
14	District	0.00000	0.00000

Code Output

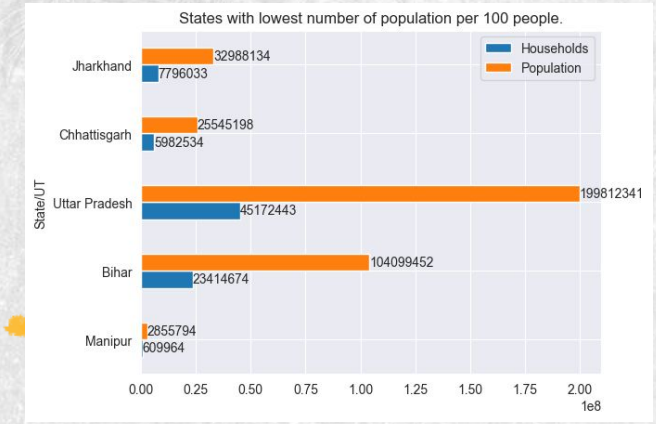
The Godfather of Talent | Futurense

#3 Household Density in India:



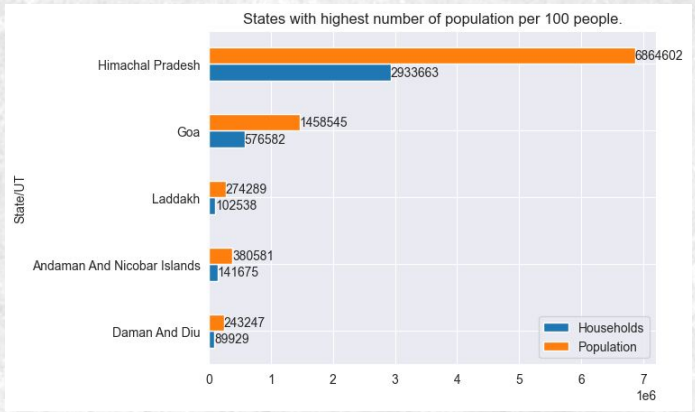
Graph 1.

- States with lowest number of households that can accommodate 100 people: **Manipur (21.36), Bihar (22.49), Uttar Pradesh (22.61)** and higher are **Himachal Pradesh (42.73), Goa (39.53), Laddakh (37.38).**
- So, such states with ***low household densities*** have fewer houses per 100 people may face challenges in providing adequate healthcare facilities to the population. Aside from that, *they have a greater risk of faster spread of diseases and infections* due to high population living in fewer households.
- States with ***high household densities*** can be densely populated or can be potentially overcrowded. We require more housing conditions to have an accurate conclusion about that.



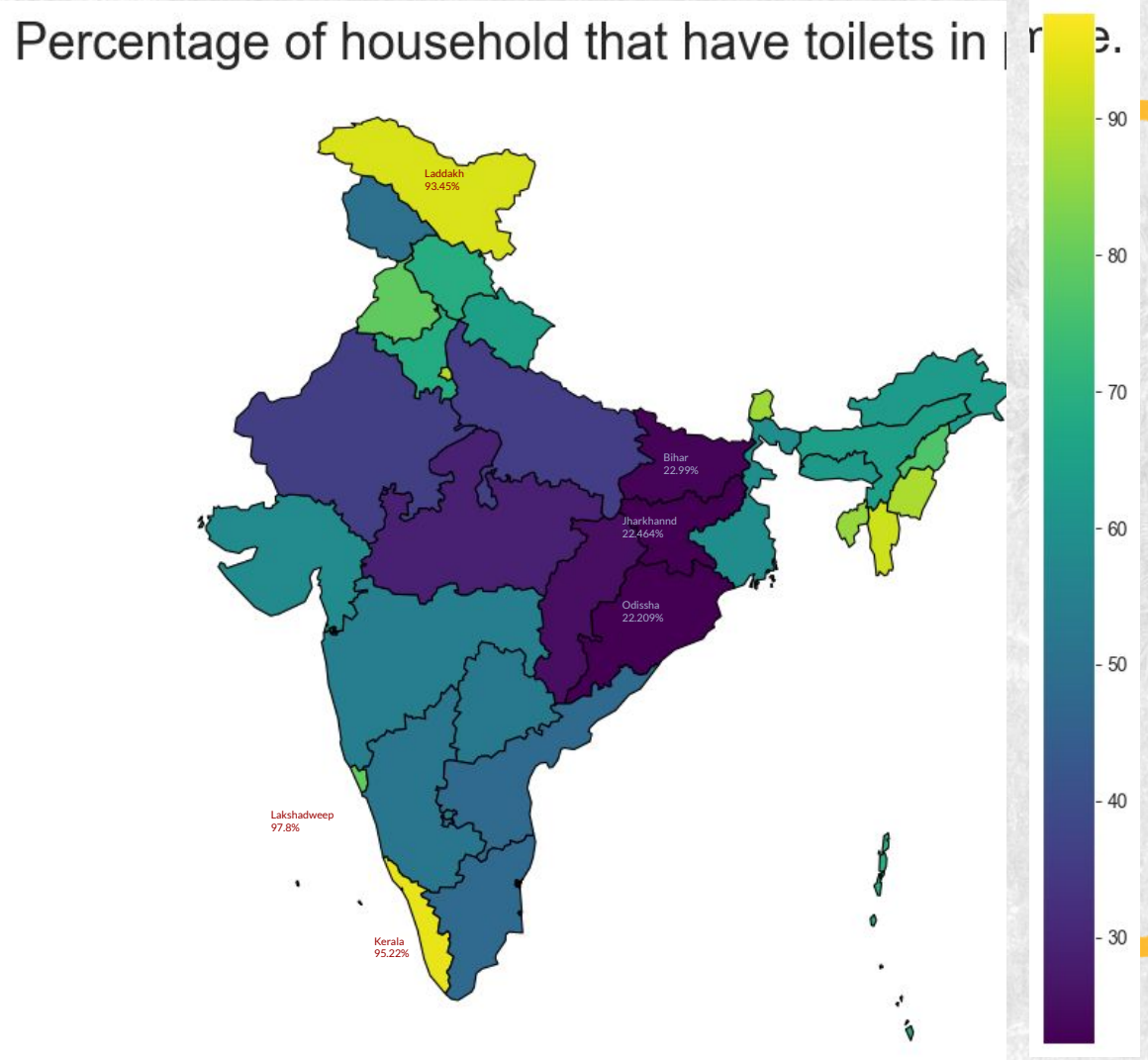
Graph 2

Graph 3



The Godfather of Talent | Futurense

#4 Household Sanitation in India:

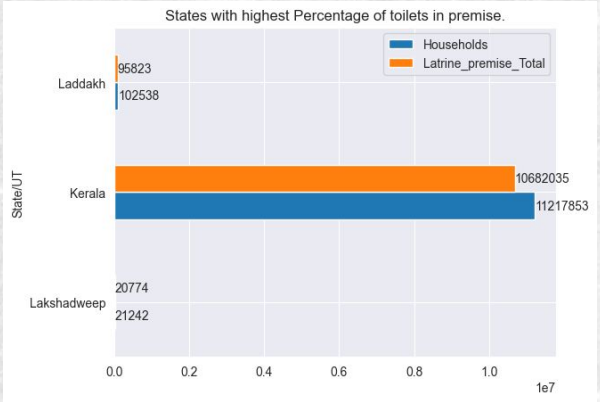


Graph 1.

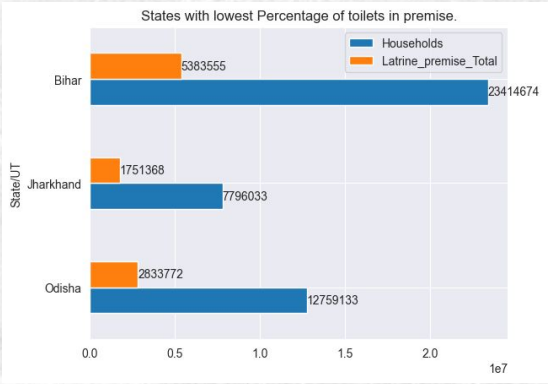
- States with *highest* percentage of toilets in premise: **Lakshadweep, Kerala, Ladakh** have better access to basic sanitation facilities, which can lead to better health outcomes.
- States with lowest percentage of toilets in premise: **Odisha, Jharkhand & Bihar** may have a higher prevalence of diseases related to poor sanitation.

Recommendations:

- The government should invest more to increase the no of households with in-house latrine premises
- Prioritize improving access to basic sanitation facilities to reduce the prevalence of diseases related to poor sanitation.
- The government can launch awareness campaigns, provide financial incentives to households that build toilets in their homes, etc.



Graph 2

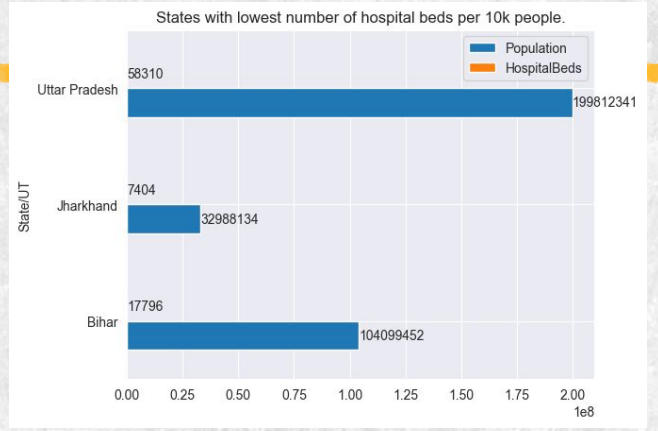


Graph 3

The Godfather of Talent | Futurense

#5 State-wise Hospital Bed Capacity in India:

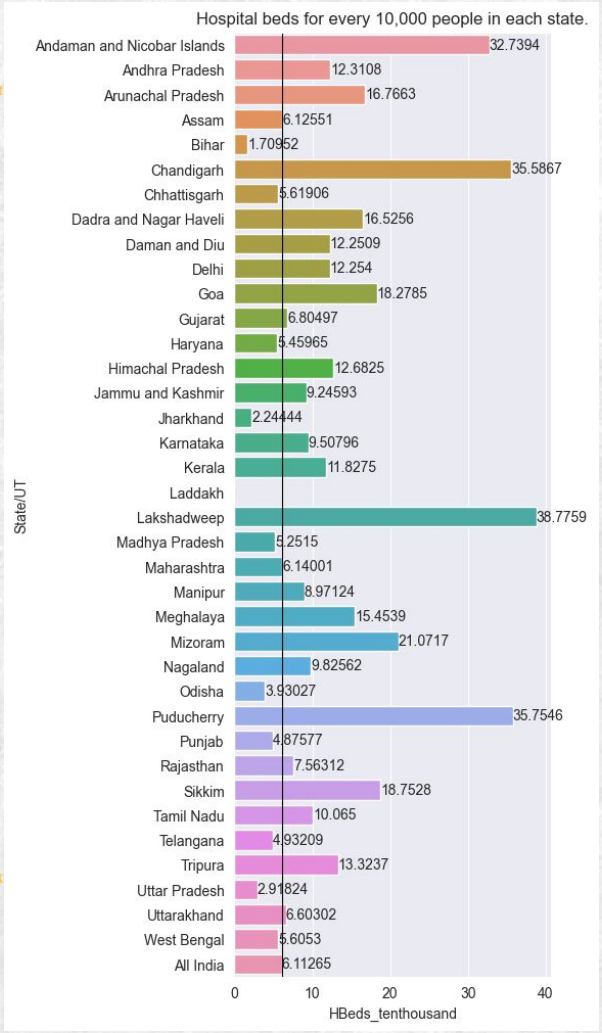
- Considering All India as benchmark, we got an approx value of 6 beds per 10k people.
- States exceeding the threshold: **Lakshadweep** (*max*) and there are some states where they fall below this range and may be more vulnerable to healthcare crises and may require more investment in healthcare infrastructure.
- Top 3 states with *least* no. of hospital beds per 10k people: **Bihar, Jharkhand & Uttar Pradesh.**
- **Reasons**: State Population doesn't meet the demand of the hospital beds per 10k people.



Graph 2.

	State/UT	Population	HospitalBeds	HBeds_tenthousand
4	Bihar	1.040995e+08	17796.0	1.709519
15	Jharkhand	3.298813e+07	7404.0	2.244443
34	Uttar Pradesh	1.998123e+08	58310.0	2.918238
26	Odisha	4.197422e+07	16497.0	3.930270
28	Punjab	2.774334e+07	13527.0	4.875765
32	Telangana	3.519398e+07	17358.0	4.932094
20	Madhya Pradesh	7.262681e+07	38140.0	5.251504
12	Haryana	2.535146e+07	13841.0	5.459646
36	West Bengal	9.127612e+07	51163.0	5.605300
6	Chhattisgarh	2.554520e+07	14354.0	5.619060
37	All India	1.209008e+09	739024.0	6.112648

Code Output



Graph 1.

The Godfather of Talent | Futurense

#6 State-wise Government Hospital Bed Capacity in India:

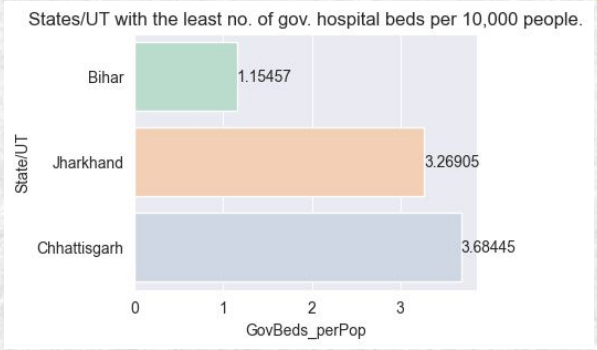
States with *lowest number* of government hospitals per 10k people:

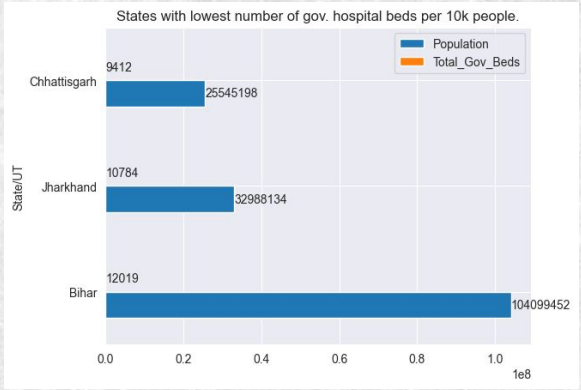
- **Bihar**
- **Jharkhand**
- **Chhattisgarh**

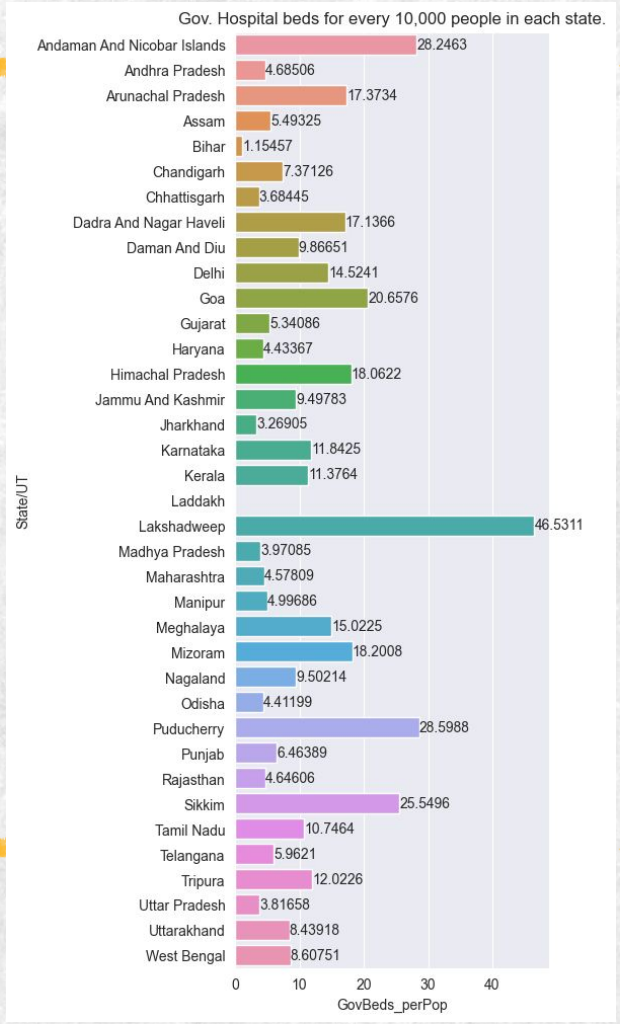
Reasons: High population doesn't meet the hospital beds requirement in government hospitals.

States with *highest number* of government hospitals per 10k people:

- **Lakshadweep**
- **Puducherry**
- **Andaman And Nicobar Islands**





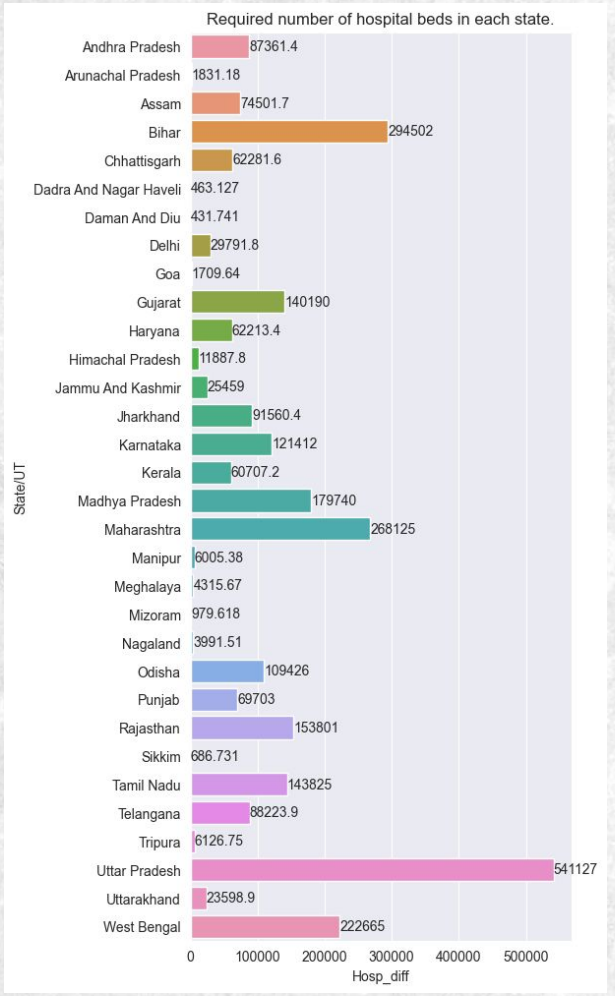


The Godfather of Talent | Futurense

#7 Gap analysis of hospital bed availability in Indian states with WHO standard:

World Health Organization

- **WHO Standards**: 3 beds per 1000 people
- The state which requires the maximum amount of hospital beds to meet the WHO standards is **Uttar Pradesh** (541127), followed by **Maharashtra** (268125).



The Godfather of Talent | Futurense

Conclusions & Recommendations

- The government needs to invest more to increase the number of households in **Manipur** to reduce the spread of infectious diseases during a healthcare crisis.
- **Bihar** & **Jharkhand** needs more households and especially **house with toilets.**
- **Bihar** also requires to meet the WHO standard number of hospital beds along with **Uttar Pradesh, Maharashtra** and **West Bengal.**

The Godfather of Talent | Futurense

Highlights & Standout Features:

- **Efficient data preprocessing**: The use of classes and their functions to preprocess the dirty dataframes multiple times in the same order for different tables is a very efficient and organized way of handling data cleaning and preparation.
- **Geospatial analysis**: The use of Geopandas to map out important features adds a layer of geospatial analysis to the project, allowing for a better understanding of the geographical distribution of the data.
- **Visualization**: The project involves the creation of multiple visualizations, such as bar plots and choropleth maps, which help to convey insights and findings to stakeholders in an easily understandable way.

Optimizations and Best Practices Used:

- Use vectorization: Avoid using loops when possible and instead use vectorization to perform operations on entire arrays or data frames at once.
- Write modular and reusable code: Write code that can be easily reused and extended for future projects by using modular design patterns and creating functions and classes.
- Document your code: Document your code using comments and docstrings to help other developers understand your code and make it easier to maintain.

The Godfather of Talent | Futurense

Future Scope:

- Collect more **recent and accurate data** to provide up-to-date analysis and solutions.
- Analyze the correlation between different variables and how they affect the healthcare system in each state.

- Explore the **impact of government policies on the healthcare system** in each state.
- Compare the healthcare system in each state to other countries or regions with similar demographics and geography.

The Godfather of Talent | Futurense

THANKYOU