Dataset Description:

The "Alzheimer's Disease and Healthy Ageing Data" collection is a valuable resource for gaining insights into Alzheimer's disease and factors influencing healthy aging. It provides a comprehensive view of the disease's prevalence and its impact on different demographics and geographical regions.

Key Features:

- Temporal and Spatial Coverage: Encompassing multiple years and diverse locations, the dataset offers a broad perspective on Alzheimer's disease, facilitating analyses of its prevalence and distribution over time and across regions.
- Demographic Details: The data includes detailed demographic information such as age, gender, and potentially social background, enabling analyses of how Alzheimer's affects different demographic groups differently.
- Uncertainty Ranges: Numeric data comes with uncertainty ranges, enhancing the reliability of analyses and interpretations.
- Geographical Information: Location data allows researchers to identify regions where Alzheimer's is more prevalent, aiding in the targeting of interventions and treatments.

Categorical columns :

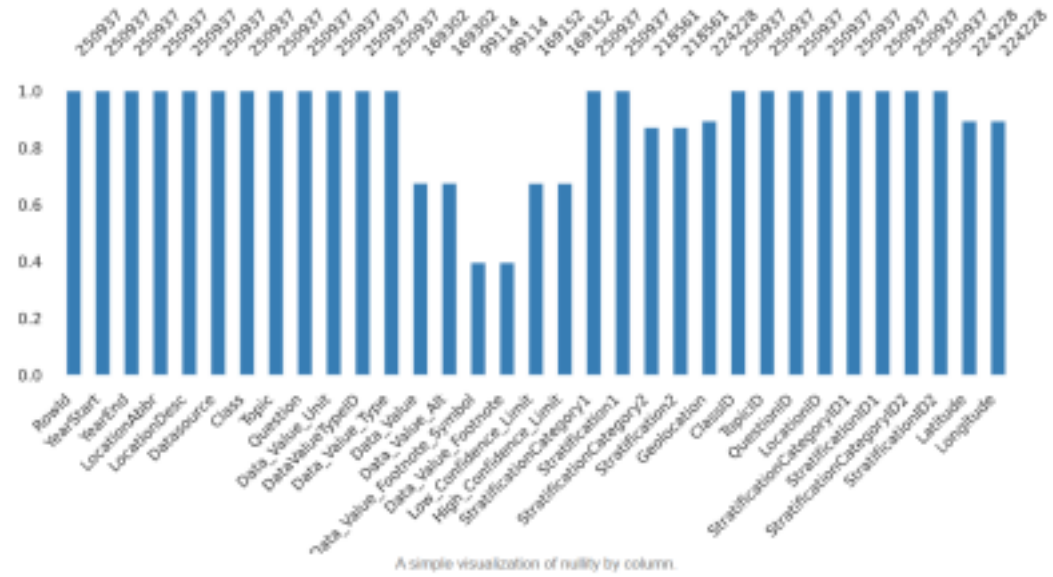
['RowId', 'LocationAbbr', 'LocationDesc', 'Datasource', 'Class', 'Topic', 'Question', 'Data_Value_Unit', 'DataValueTypeID', 'Data_Value_Type', 'Data_Value_Footnote_Symbol', 'Data_Value_Footnote', 'Low_Confidence_Limit', 'High_Confidence_Limit', 'StratificationCategory1', 'Stratification1', 'StratificationCategory2', 'Stratification2', 'Geolocation', 'ClassID', 'TopicID', 'QuestionID', 'StratificationCategoryID1', 'StratificationID1', 'StratificationCategoryID2', 'StratificationID2', 'Latitude', 'Longitude']

Numerical columns :

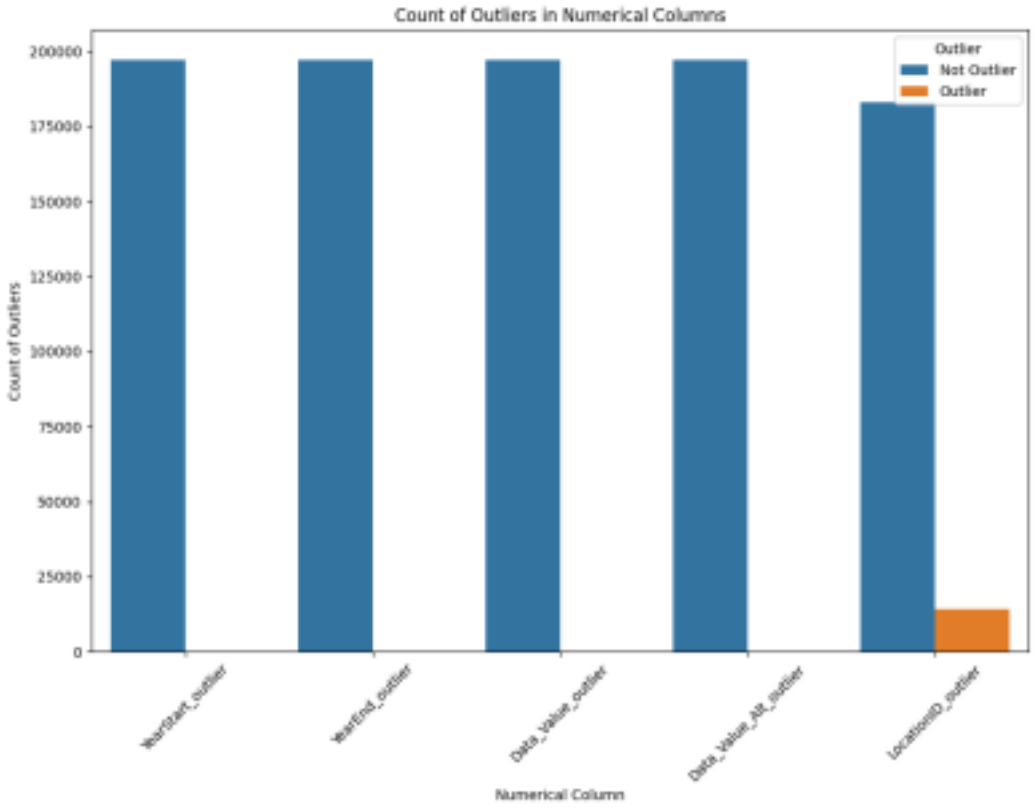
['YearStart', 'YearEnd', 'Data_Value', 'Data_Value_Alt', 'LocationID']

Data Problems

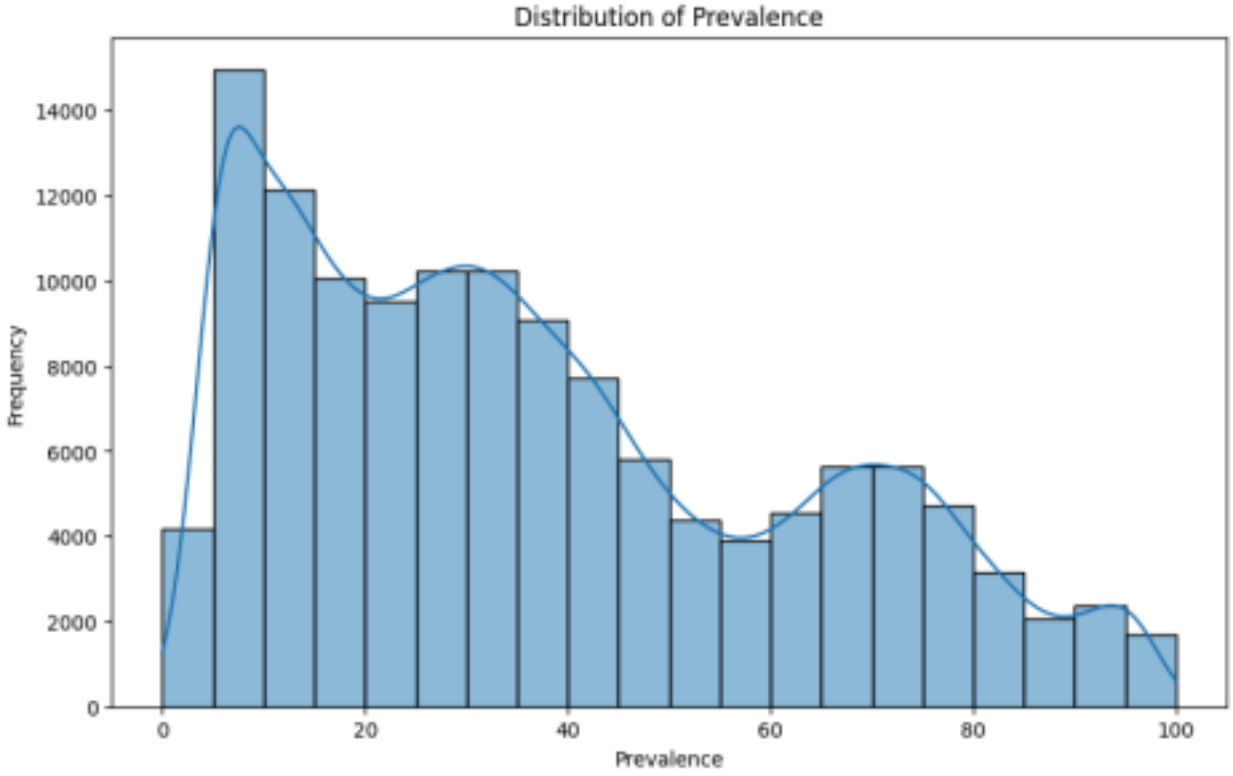
1) Missing Values Count



2) Outliers



3) Skewed Data



- **Missing Values**: can lead to biased estimates and inaccurate conclusions if not appropriately handled.
- **Outliers**: can distort summary statistics, such as means and standard deviations, affecting the central tendency and variability of the data.
- **Data Skewness**: can violate assumptions of normality underlying many statistical methods, potentially leading to biased estimates and incorrect inferences.

Methodology

1) After identifying the categorical and numerical columns, I decided to fill null values in the numerical columns by replacing them with the **mean** of each respective column. This approach preserves the ***overall distribution*** of the data and ***maintains the statistical properties*** of the dataset. Filling null values with the mean ensures that the imputed values are representative of the existing data points in that column, thus aiding in the analysis and preventing data loss. For the categorical columns, I chose to fill null cells with the **mode** of each column. This strategy ***maintains the integrity*** of the categorical column's missing data by replacing them with the most frequently occurring category. By doing so, **the categorical nature of the data is preserved**, and the imputed values remain valid categories within the column . Both of these behaviors are effective in ensuring that the nature of the data is well-maintained, rather than blindly dropping the null values, which could lead to information loss and biased analyses.

```
numerical_cols = df.select_dtypes(include=['float64', 'int64']).columns
df[numerical_cols] = df[numerical_cols].fillna(df[numerical_cols].mean())

object_cols = df.select_dtypes(include='object').columns
for col in object_cols:
    df[col] = df[col].fillna(df[col].mode()[0])

df.drop_duplicates(inplace=True)

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

```
RowId                        0
YearStart                    0
YearEnd                      0
LocationAbbr                 0
LocationDesc                 0
Datasource                   0
Class                        0
Topic                        0
Question                     0
Data_Value_Unit              0
DataValueTypeID              0
Data_Value_Type              0
Data_Value                   0
Data_Value_Alt               0
Data_Value_Footnote_Symbol   0
Data_Value_Footnote          0
Low_Confidence_Limit         0
High_Confidence_Limit        0
StratificationCategory1      0
Stratification1              0
StratificationCategory2      0
Stratification2              0
Geolocation                  0
ClassID                      0
TopicID                      0
QuestionID                   0
LocationID                   0
StratificationCategoryID1    0
StratificationID1            0
StratificationCategoryID2    0
StratificationID2            0
Latitude                     0
Longitude                    0
dtype: int64
```

2) I developed a schema and validated my data against it. The schema ensured that the data **adhered to predefined constraints and rules**, such as data types, ranges, and formats [Here You Can Find The Generated Schema For My Data Set]

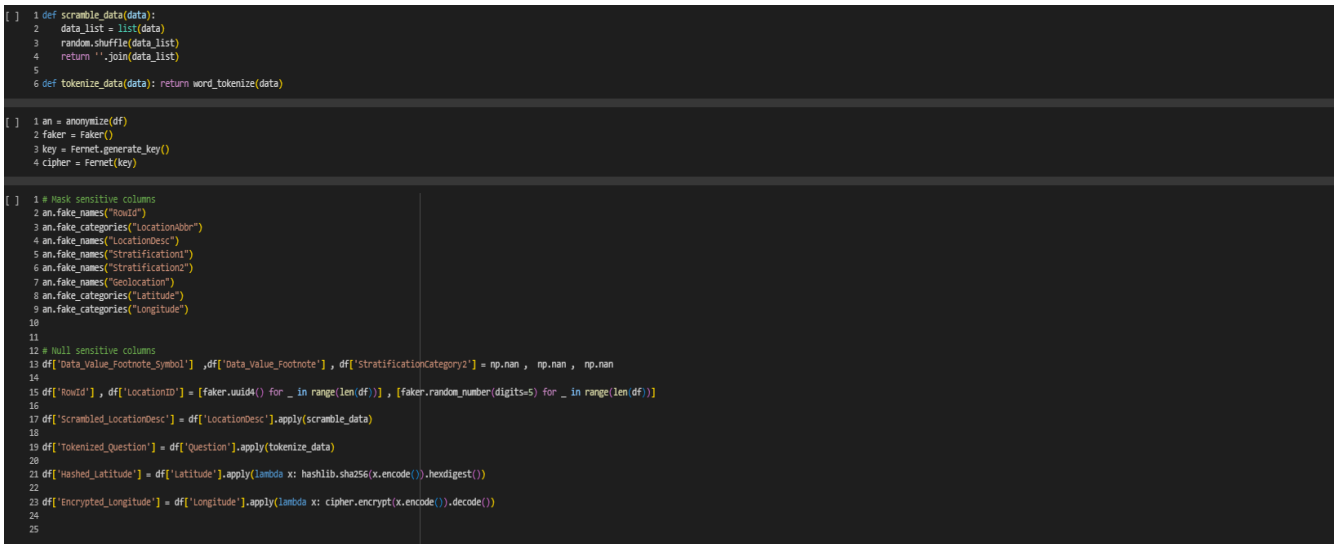
```python
schema = DataFrameSchema({
    "RowId": Column(int, nullable=True),
    "YearStart": Column(int, Check.greater_than_or_equal_to(0)),
    "YearEnd": Column(int, Check.greater_than_or_equal_to(0)),
    "LocationAbbr": Column(str),
    "LocationDesc": Column(str),
    "Datasource": Column(str),
    "Class": Column(str),
    "Topic": Column(str),
    "Question": Column(str),
    "Data_Value_Unit": Column(str),
    "DataValueTypeID": Column(str),
    "Data_Value_Type": Column(str),
    "Data_Value": Column(float, Check.greater_than_or_equal_to(0)),
    "Data_Value_Alt": Column(float, nullable=True),
    "Data_Value_Footnote_Symbol": Column(str, nullable=True),
    "Data_Value_Footnote": Column(str, nullable=True),
    "Low_Confidence_Limit": Column(float),
    "High_Confidence_Limit": Column(float),
    "StratificationCategory1": Column(str),
    "Stratification1": Column(str),
    "StratificationCategory2": Column(str, nullable=True),
    "Stratification2": Column(str, nullable=True),
    "Geolocation": Column(str, nullable=True),
    "ClassID": Column(str),
    "TopicID": Column(str),
    "QuestionID": Column(str),
    "LocationID": Column(str),
    "StratificationCategoryID1": Column(str),
    "StratificationID1": Column(str),
    "StratificationCategoryID2": Column(str, nullable=True),
    "StratificationID2": Column(str, nullable=True),
    "Latitude": Column(float, nullable=True),
    "Longitude": Column(float, nullable=True),
})

try:
    schema.validate(df, lazy=True)
except pa.errors.SchemaErrors as exc:
    failure_cases_df = exc.failure_cases
    display(exc.failure_cases)
```

	schema_context	column	check	check_number	failure_case	index
0	Column	RowId	dtype('int64')	None	object	None
131579	Column	LocationID	dtype('str')	None	12.0	131576
131509	Column	LocationID	dtype('str')	None	18.0	131506
131510	Column	LocationID	dtype('str')	None	18.0	131507
131511	Column	LocationID	dtype('str')	None	20.0	131508

3)

- Fake data is substituted for sensitive details like rows id and locations.
- Some data is left blank on purpose to make it harder to identify individuals.
- New, fake data is added to non-sensitive areas to keep the data structure intact.
- Descriptions of places are jumbled up to hide real locations. , Text is broken down into pieces for easier analysis.
- Numbers used for location (like latitude) are converted into a code to hide them entirely.
- Other location data (like longitude) is scrambled to make it unreadable without a key.

Here You Can Find The New Modified Data Set : [Click Here]



```
[ ]  1 def scramble_data(data):
     2     data_list = list(data)
     3     random.shuffle(data_list)
     4     return ''.join(data_list)
     5
     6 def tokenize_data(data): return word_tokenize(data)
```

```
[ ]  1 an = anonymize(df)
     2 faker = Faker()
     3 key = Fernet.generate_key()
     4 cipher = Fernet(key)
```

```
[ ]  1 # Mask sensitive columns
     2 an.fake_names("RowId")
     3 an.fake_categories("LocationAbbr")
     4 an.fake_names("LocationDesc")
     5 an.fake_names("Stratification1")
     6 an.fake_names("Stratification2")
     7 an.fake_names("Geolocation")
     8 an.fake_categories("Latitude")
     9 an.fake_categories("Longitude")
    10
    11
    12 # Null sensitive columns
    13 df['Data_Value_Footnote_Symbol']  ,df['Data_Value_Footnote'] , df['StratificationCategory2'] = np.nan ,  np.nan ,  np.nan
    14
    15 df['RowId'] , df['LocationID'] = [faker.uuid4() for _ in range(len(df))] , [faker.random_number(digits=5) for _ in range(len(df))]
    16
    17 df['Scrambled_LocationDesc'] = df['LocationDesc'].apply(scramble_data)
    18
    19 df['Tokenized_Question'] = df['Question'].apply(tokenize_data)
    20
    21 df['Hashed_Latitude'] = df['Latitude'].apply(lambda x: hashlib.sha256(x.encode()).hexdigest())
    22
    23 df['Encrypted_Longitude'] = df['Longitude'].apply(lambda x: cipher.encrypt(x.encode()).decode())
    24
    25
```

4) After completing the data processing tasks, including *data cleaning*, *validation* against a schema, and *securing sensitive columns with encryption keys, hashing, and tokenization*, I decided to implement an **RBAC (Role-Based Access Control)** system. I chose RBAC because it allows for adaptability in accessing the given data, accommodating various roles efficiently. Initially, I defined three roles for experimental purposes: Data Analyst, Researcher, and Administrator. However, this system can be expanded in the future to include additional roles.

With RBAC, each role is assigned specific permissions, ensuring that users only have access to the columns relevant to their role's requirements. This approach **eliminates the need for users to view unnecessary information, enhancing data security and efficiency in accessing relevant data**.

```
rbac.add_role('Data Analyst', ['LocationAbbr', 'YearStart', 'Data_Value', 'Class', 'StratificationID1', 'StratificationID2'])
rbac.add_role('Researcher', ['LocationAbbr', 'Data_Value', 'Low_Confidence_Limit', 'Data_Value_Footnote', 'Question'])
rbac.add_role('Administrator',['RowId', 'YearStart', 'YearEnd', 'LocationAbbr', 'LocationDesc',
   'Datasource', 'Class', 'Topic', 'Question', 'Data_Value_Unit',
   'DataValueTypeID', 'Data_Value_Type', 'Data_Value', 'Data_Value_Alt',
   'Data_Value_Footnote_Symbol', 'Data_Value_Footnote',
   'Low_Confidence_Limit', 'High_Confidence_Limit',
   'StratificationCategory1', 'Stratification1', 'StratificationCategory2',
   'Stratification2', 'Geolocation', 'ClassID', 'TopicID', 'QuestionID',
   'LocationID', 'StratificationCategoryID1', 'StratificationID1',
   'StratificationCategoryID2', 'StratificationID2', 'Latitude',
   'Longitude'])
```

For example, I have defined only three users, with each user assigned to a specific role.

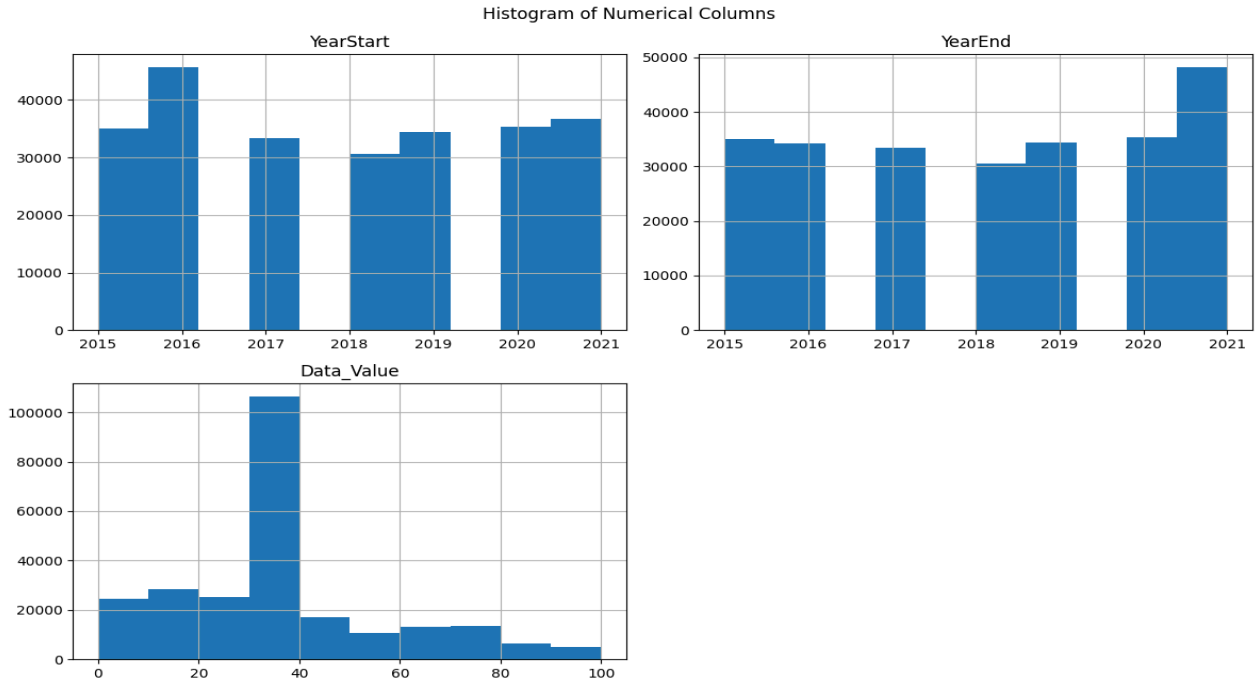
```
user1.username = 'Adham' ; user1.password='Big Boss'
user2.username = 'Youssef'; user2.password='Medium Boss'
user3.username = 'Hozien' ; user3.password='Small Boss'


rbac.add_user(user1)
rbac.add_user(user2)
rbac.add_user(user3)

rbac.assign_user_role(user3, 'Data Analyst')
rbac.assign_user_role(user2, 'Researcher')
rbac.assign_user_role(user1, 'Administrator')

rbac.set_user_credentials(user3.username,user3.password)
rbac.set_user_credentials(user2.username,user2.password)
rbac.set_user_credentials(user1.username,user1.password )
```

When Youssef, acting as a **Researcher**, attempted to access the system, only the columns authorized for his role were displayed to him.

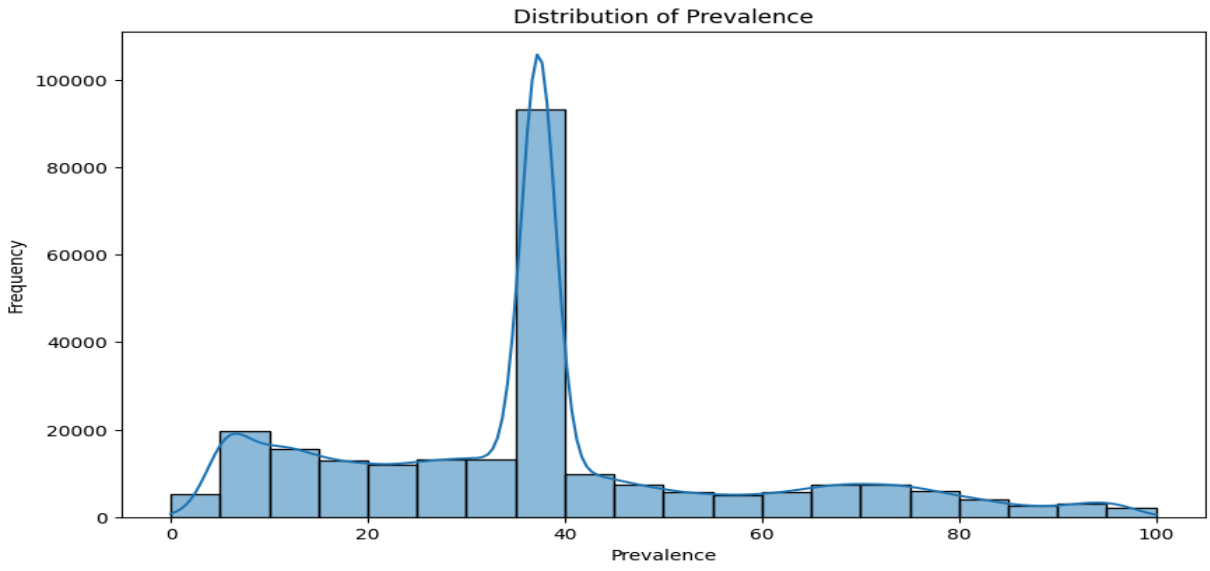
```
Enter your username: Youssef
Enter your password: Medium Boss
Authentication successful!
Your role: Researcher
Authorized columns: LocationAbbr, Data_Value, Low_Confidence_Limit, Data_Value_Footnote, Question




Loading ..........




   LocationAbbr  Data_Value Low_Confidence_Limit  \
0           MDW   10.600000                 9.9
1           MDW    4.300000                 4.1
2          WEST   33.200000                32.1
3            KY   37.328349                 4.7
4           SOU   55.500000                53.0
```

Results (For Good Quality All Graphs Are In The Uploded NoteBook:

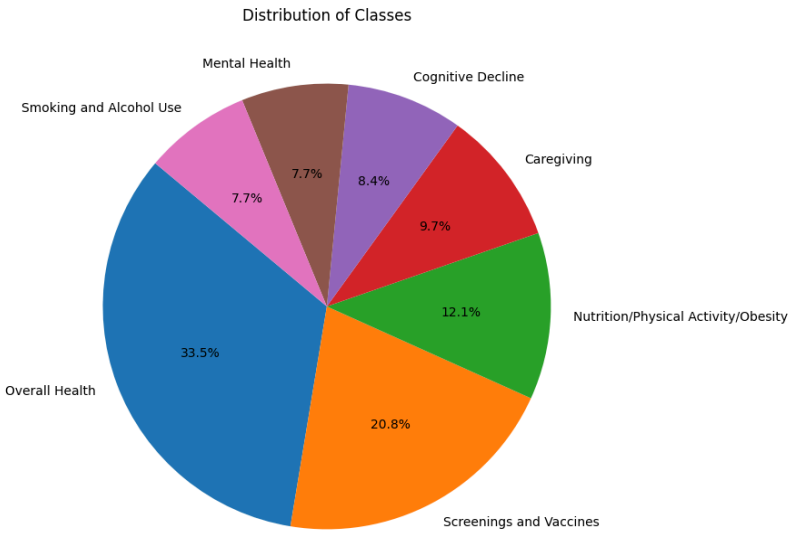
Following data quality checks **(Profiling,Validation, Vrivacy, and Security)**, I decided to move through data exploration to assess the data's reusability. To gain a comprehensive understanding of the numerical columns, I have constructed histograms. These visualizations effectively illustrated the distribution of **'YearStart', 'YearEnd', 'Data_Value', 'Low_Confidence_Limit', and 'High_Confidence_Limit'**. This analysis provided valuable insights into the data's spread across different years and confidence intervals, laying the groundwork for further exploration and potential reuse.



Shifting our focus to **prevalence**, a histogram was augmented with a Kernel Density Estimate **(KDE)** to visualize its distribution. This analysis revealed a **Normal distribution**, suggesting a symmetrical spread of prevalence levels across the dataset. Consequently, the **skewness issue** previously observed has been effectively addressed..

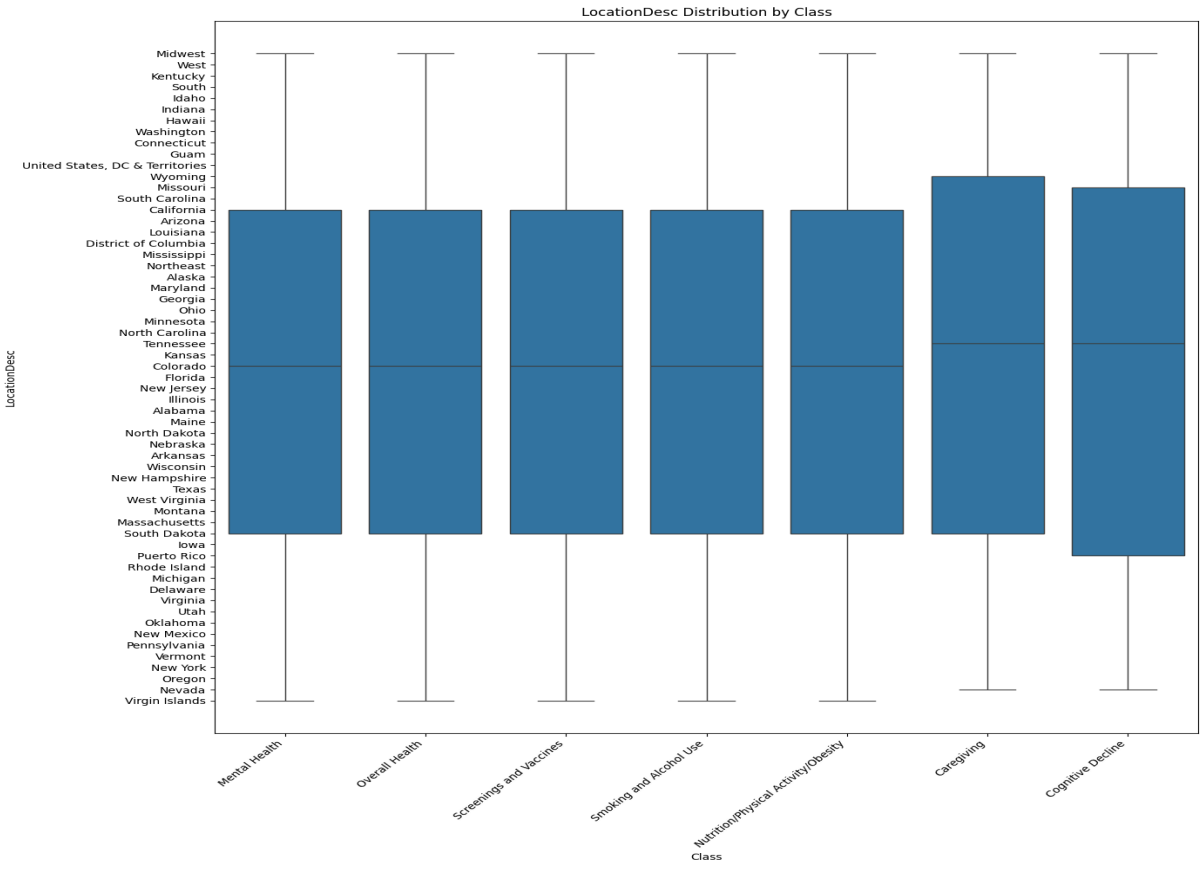


This visualization effectively depicted the proportional representation of each health class, providing a clear understanding of the prevalence of different health conditions. The pie chart might suggest a potential link between a lack of interest in **overall health and a higher prevalence of Alzheimer's Disease**.

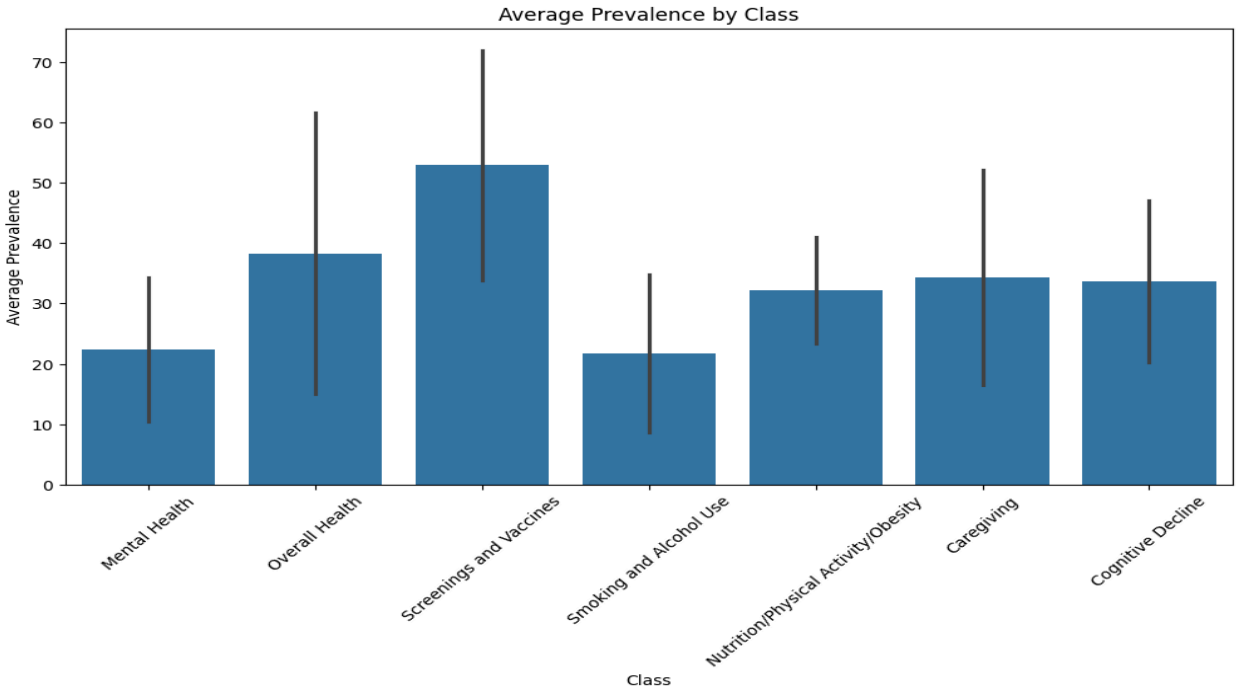


To investigate potential geographical variations in health outcomes, Relationship between **'LocationDesc' and 'Class'**. Visualizing the distribution of health classes across different locations. This analysis provided insights into potential regional disparities in health outcomes, highlighting areas that might warrant further investigation. **(Tennessee For E.G)**

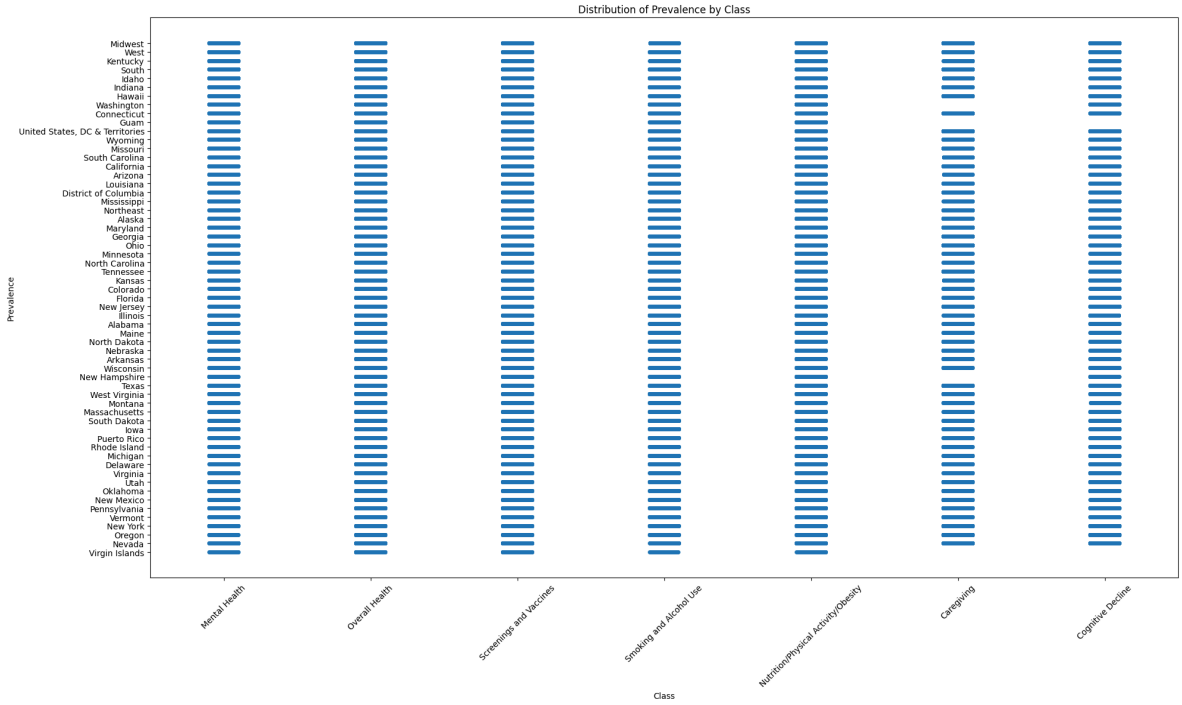
.



To delve deeper into variations in health outcomes across different health categories, The average prevalence within each class . was highlighted withing the differences in average prevalence levels for each health class.



This visualization effectively depicted the distribution of prevalence across various locations and health classes this way ease the i**dentification process** of **potential interactions between location and health class**..



Conclusion

The visualizations collectively offer a rich understanding of the dataset, laying the foundation for more in-depth analysis and actionable insights. This can remark us for many ways :

- The dataset's wealth of numerical features, encompassing values like **'Data_Value', 'YearStart', and 'YearEnd'**, presents a valuable resource for machine learning model development. By analyzing the distribution and patterns within these variables, data scientists can make informed decisions regarding feature selection and engineering.

- This exploration of the dataset's geographical and prevalence data paves the way for the application of machine learning algorithms. By training these algorithms on the identified relationships, we can potentially predict health classes based on factors such as location and prevalence rates.