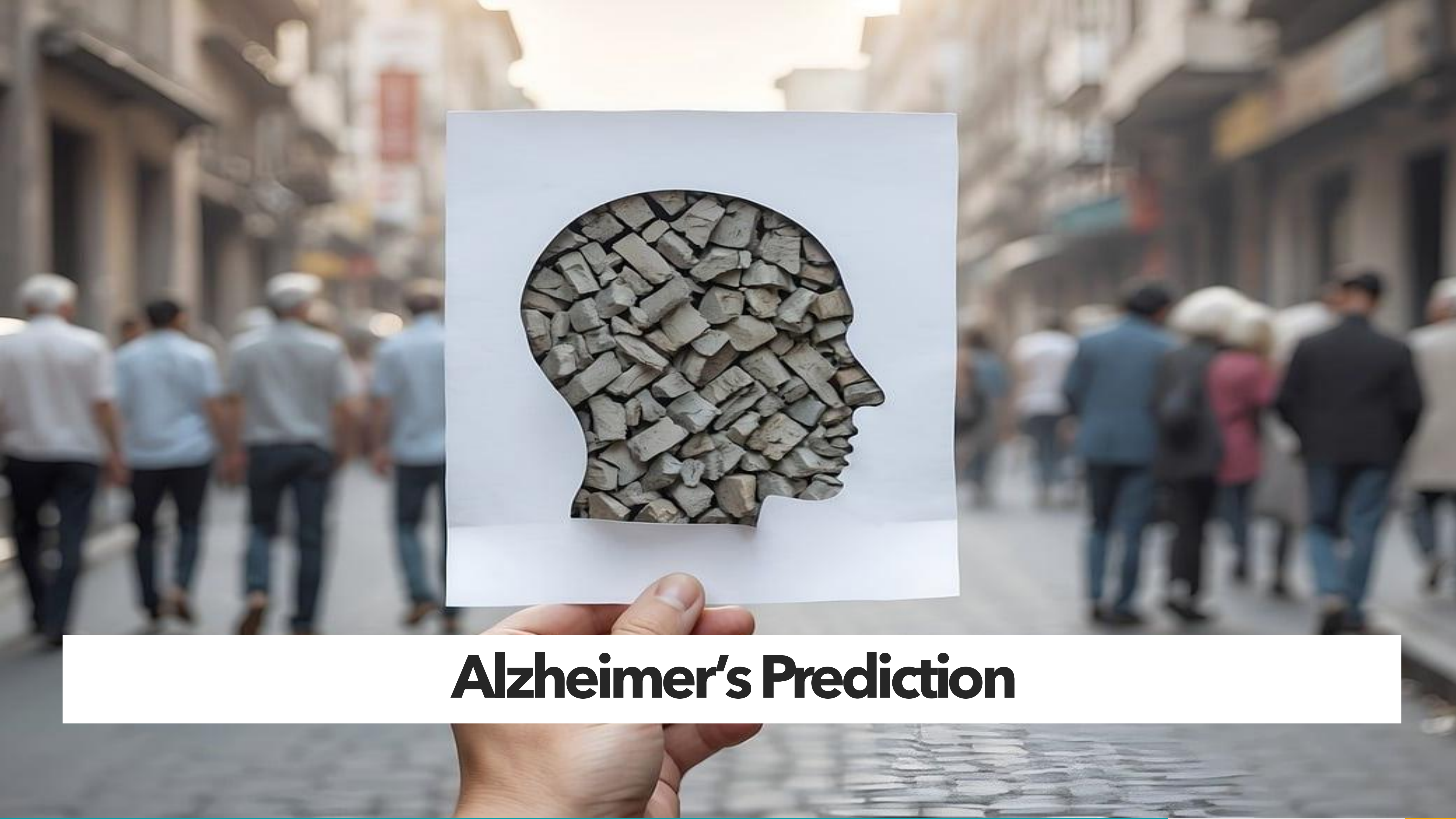


Alzheimer's Prediction

Team 8

Thanishma Bollineni (ADTA)	**Sri Ramya Katukuri (DS)**	**Yogesh Meka (DS)**
• Modelling • EDA • Presentation	• Preprocessing • Prediction • Presentation	• Preprocessing • Visualization • Presentation

Project Guide: **Prof. Dr. Boyce Leann**

Problem Statement

With the aging of Populations across the world, there has never been a better time to understand the factors affecting cognitive health. This survey-based dataset cover older adults in the area of cognitive health regarding factors such as memory issues, mental well-being, and the ability to perform daily activities. By analyzing the variations across demographic groups such as age, gender, and other characteristics the study looks to identify early signs of Alzheimer's disease. The insights derived will support healthcare professionals in improving early detection, guide targeted interventions and inform public health strategies that promote healthy aging. Our data was published by the Behavioral Risk Factor Surveillance System (BRFSS), which is a comprehensive, nationwide survey conducted by the Centers for Disease Control and Prevention (CDC).

Business Understanding

- The goal of cognitive progression in health is early detection of disorders in cognition before these disorders have adverse health consequences.
- This helps in understanding who or which groups are at risk of developing conditions such as memory loss and finding ways to support them, thus reducing suffering from such conditions as dementia.
- It improves the quality of healthcare and helps in formulating public health policies that deal with the many problems associated with aging.
- It aids in decision-making on resource allocation, promotes awareness and concern, and improves the well-being of the elderly.
- It also encourages practical steps toward decreasing the burden of cognitive decline to ensure better results for individuals and communities.

Data Understanding

The dataset “Alzheimer’s Disease and Healthy Ageing” contains **284,142** rows and **31** columns, which include year, location, mental health topics, demographic stratifications, and detailed questions about aging and mental health metrics.

Columns	**Description**
YearStart/YearEnd	Specifies the year of data collection
LocationDesc	Provides information about location where the data is collected from.
Question	Gives the context about the data collected
Data_Value	Responses to the questions asked in the survey.
Confidence_Limits	Lower and upper bounds of confidence limits while giving responses to questions in the form of data value.
Stratification	Classification of the individual for whom we have our response for. (Gender, Race/Ethinicity)

Before Preprocessing

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

```
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                   91334
Data_Value_Alt               91334
Data_Value_Footnote_Symbol  174166
Data_Value_Footnote         174166
Low_Confidence_Limit         91545
High_Confidence_Limit        91545
StratificationCategory1          0
Stratification1                  0
StratificationCategory2      36873
Stratification2              36873
Geolocation                  30489
ClassID                          0
TopicID                          0
QuestionID                       0
LocationID                       0
StratificationCategoryID1        0
StratificationID1                0
StratificationCategoryID2        0
StratificationID2                0
```

```
Data columns (total 31 columns):
 #   Column                      Non-Null Count   Dtype
---  ------                      --------------   -----
 0   RowId                       284142 non-null  object
 1   YearStart                   284142 non-null  int64
 2   YearEnd                     284142 non-null  int64
 3   LocationAbbr                284142 non-null  object
 4   LocationDesc                284142 non-null  object
 5   Datasource                  284142 non-null  object
 6   Class                       284142 non-null  object
 7   Topic                       284142 non-null  object
 8   Question                    284142 non-null  object
 9   Data_Value_Unit             284142 non-null  object
 10  DataValueTypeID             284142 non-null  object
 11  Data_Value_Type             284142 non-null  object
 12  Data_Value                  192808 non-null  float64
 13  Data_Value_Alt              192808 non-null  float64
 14  Data_Value_Footnote_Symbol  109976 non-null  object
 15  Data_Value_Footnote         109976 non-null  object
 16  Low_Confidence_Limit        192597 non-null  float64
 17  High_Confidence_Limit       192597 non-null  float64
 18  StratificationCategory1     284142 non-null  object
 19  Stratification1             284142 non-null  object
 20  StratificationCategory2     247269 non-null  object
 21  Stratification2             247269 non-null  object
 22  Geolocation                 253653 non-null  object
 23  ClassID                     284142 non-null  object
 24  TopicID                     284142 non-null  object
 25  QuestionID                  284142 non-null  object
 26  LocationID                  284142 non-null  int64
 27  StratificationCategoryID1   284142 non-null  object
 28  StratificationID1           284142 non-null  object
 29  StratificationCategoryID2   284142 non-null  object
 30  StratificationID2           284142 non-null  object
dtypes: float64(4), int64(3), object(24)
```

After Preprocessing

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

```
YearStart                0
YearEnd                  0
LocationDesc             0
Class                    0
Topic                    0
Question                 0
Data_Value_Type          0
Data_Value               0
Low_Confidence_Limit     0
High_Confidence_Limit    0
AgeGroup                 0
Stratification2          0
QuestionID               0
Alzheimers               0
dtype: int64
<class 'pandas.core.frame.DataFrame'>
Index: 227965 entries, 0 to 284141
Data columns (total 14 columns):
 #   Column                  Non-Null Count   Dtype
---  ------                  --------------   -----
 0   YearStart               227965 non-null  int64
 1   YearEnd                 227965 non-null  int64
 2   LocationDesc            227965 non-null  object
 3   Class                   227965 non-null  object
 4   Topic                   227965 non-null  object
 5   Question                227965 non-null  object
 6   Data_Value_Type         227965 non-null  object
 7   Data_Value              227965 non-null  float64
 8   Low_Confidence_Limit    227965 non-null  float64
 9   High_Confidence_Limit   227965 non-null  float64
 10  AgeGroup                227965 non-null  object
 11  Stratification2         227965 non-null  object
 12  QuestionID              227965 non-null  object
 13  Alzheimers              227965 non-null  object
dtypes: float64(3), int64(2), object(9)
memory usage: 26.1+ MB
```

Data Preprocessing

We cleaned ,the dataset of 284,412 rows using Python, the main tool used to ensure scalability and efficiency throughout the workflow

Irrelevant columns were removed to reduce the dataset to 16 columns and 227,965 rows.

All the duplicated records and blank entries were found and removed.

Missing values in the 'confidence' column were treated using mean imputation to retain data accuracy and analysed statistical properties of all columns using df.describe() to gain insights into the dataset.

Columns deemed irrelevant for EDA and modelling were removed

Misleading or unclear names were updated for clarity, such as renaming 'Stratification1' to 'Age Group'

We majorly dropped columns that are not in order and duplicated columns as (Location : LocationDesc)

Synthetic Column Generation for Alzheimer's Analysis
Purpose of Synthetic Column:

- Classify data into 'Yes' or 'No', based on the classification on the health topics, questions, data values, and their conditions.

- Emphasize trends in health topics touching on Alzheimer's, lifestyle habits, and health screenings.

- Estimate the chances of developing Alzheimer's disease and how our lifestyle, daily routine, and healthy aging would influence our cognitive health.

Logic for Creating Synthetic Column:

Set 1:

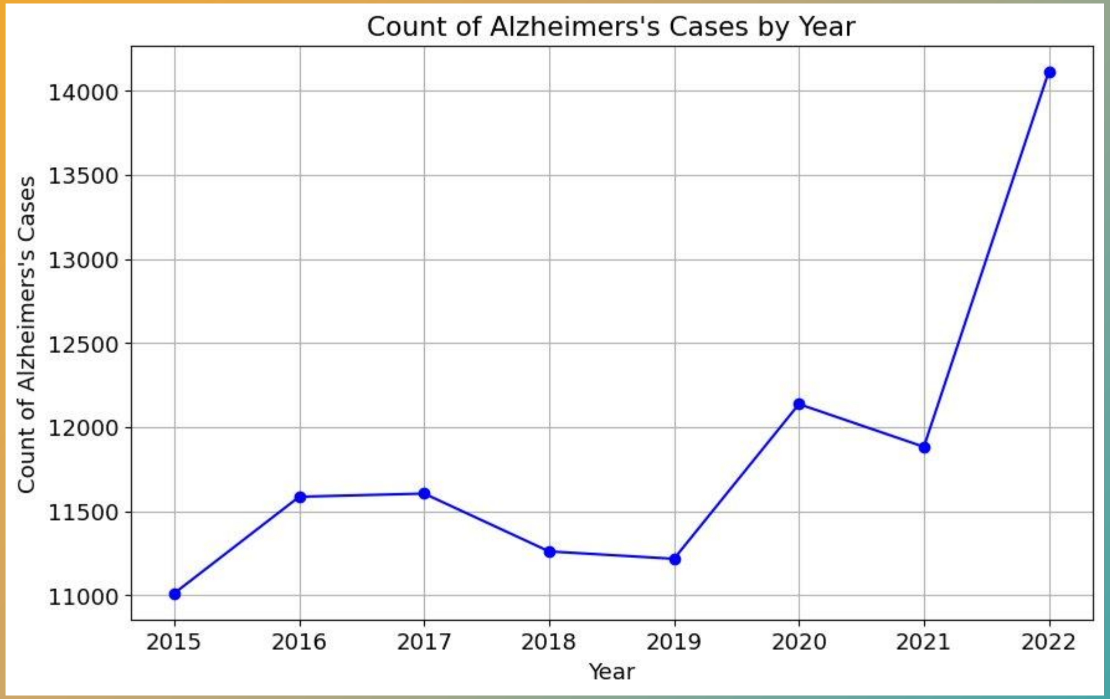
These are negative questions where the new synthetic column of likely to obtain alzheimer's would be yes (depending on the confidence bounds) if the answer for Data_Value is > Mean value, which is 39.74.

Set 2:

These are Positive questions where the new synthetic column of likely to obtain alzheimer's would be yes (depending on the confidence bounds), if the answer for Data_Value is < Mean value, which is 39.74.

High and low confidence limits depend on the boundaries of the confidence that express how confident or unconfident the person is to give a response to the inquiry with regard to the data value; high > low means they are confident, and low > high means vice-versa.

```
Is Topic in Set 1?
  ├── Yes: Is Data_Value > 39.740172 AND High > Low?
  │         ├── Yes: Set "Yes"
  │         └── No: Set "No"
  └── No: Is Topic in Set 2?
            ├── Yes: Is Data_Value >= 39.740172 AND Low > High?
            │         ├── Yes: Set "No"
            │         └── No: Set "Yes"
            └── No: Set "No"
```



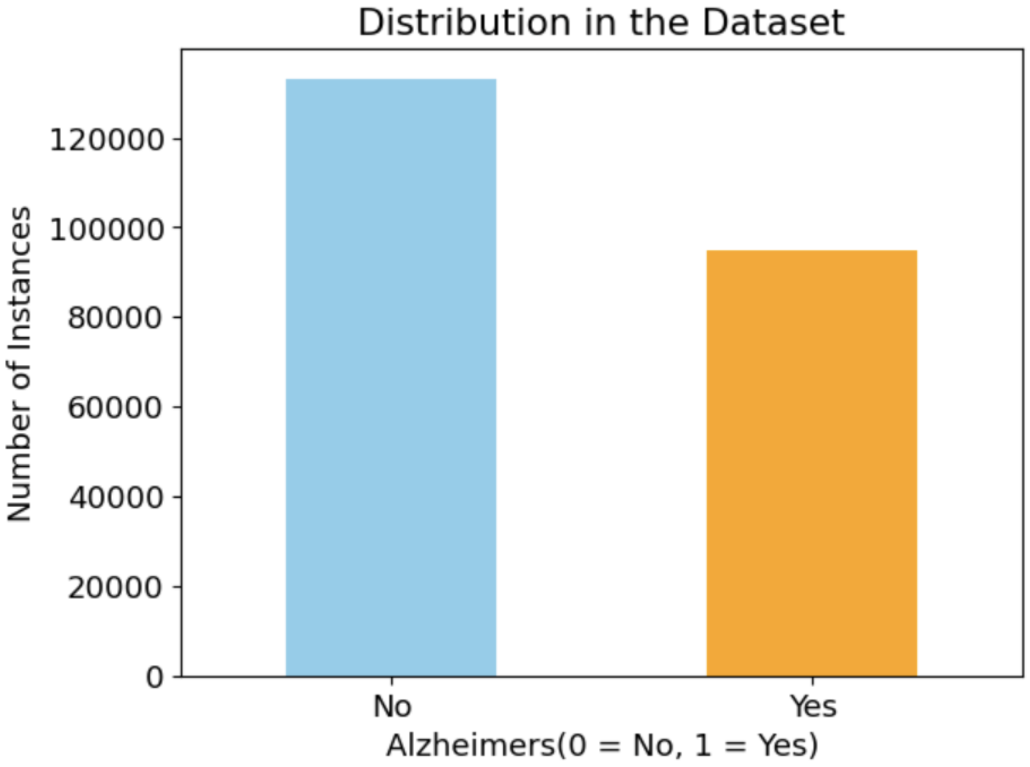
Exploratory Data Analysis (EDA) Trends and Distribution : Yearly Trends in Alzheimer's Cases

- Filtered the data on Alzheimer's cases: filtered in 'Alzheimer's=Yes', then grouped by YearEnd to count the occurrences.
- Plotted a line chart to visualize how the cases of Alzheimer's varies across years, including markers and gridlines for better readability.
- Trend analysis gave a temporal view showing the possible growth or reduction of the cases over time.

Bar Plot Visualization of Alzheimer’s

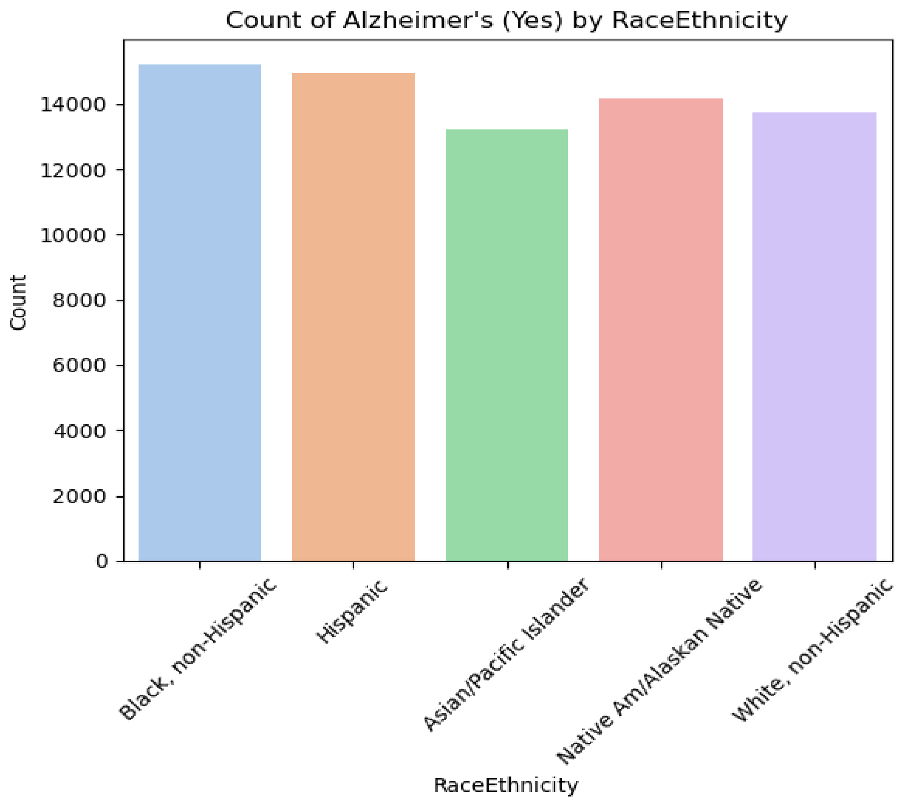
Distribution of the Alzheimer’s cases as 0=No and 1=Yes

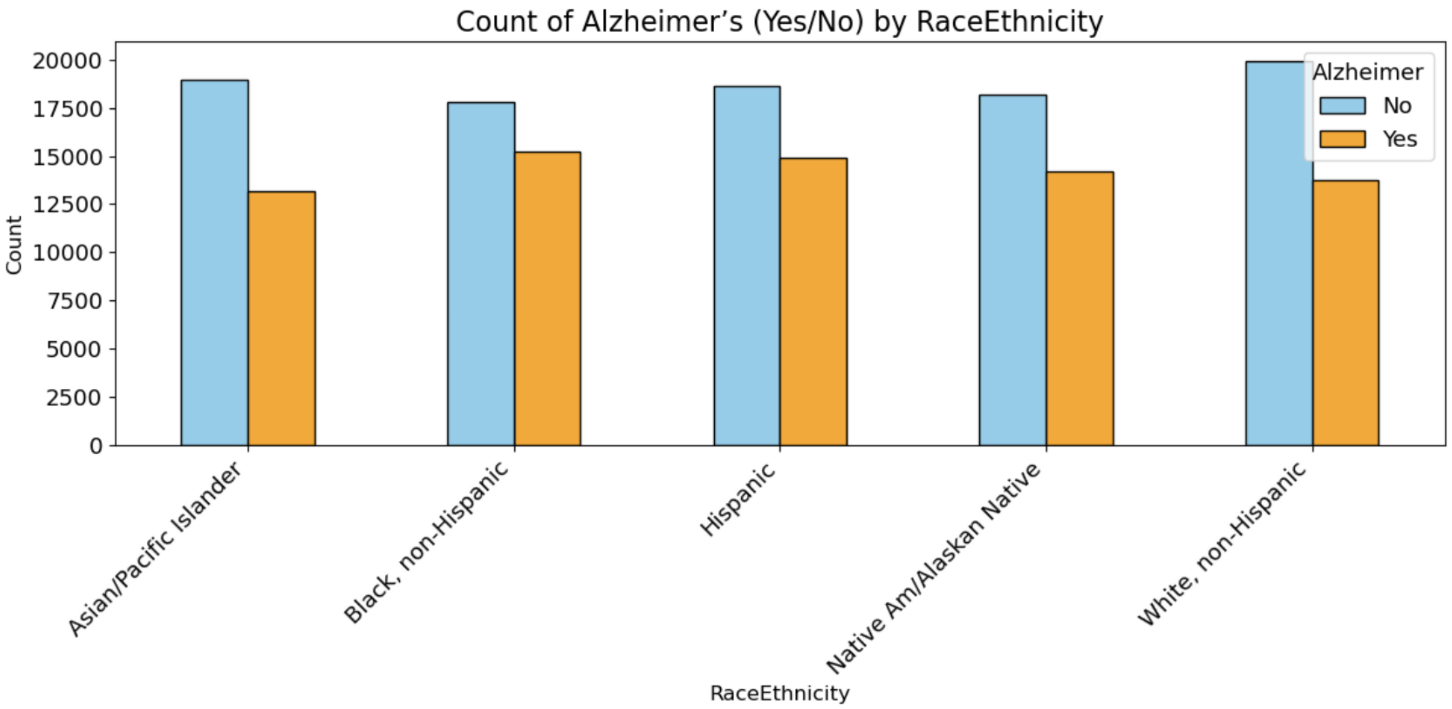
```
Distribution:
 Alzheimers
No     133157
Yes     94808
Name: count, dtype: int64

Percentage:
 Alzheimers
No     58.41116
Yes    41.58884
Name: count, dtype: float64
```



Analysis by Race/Ethnicity:

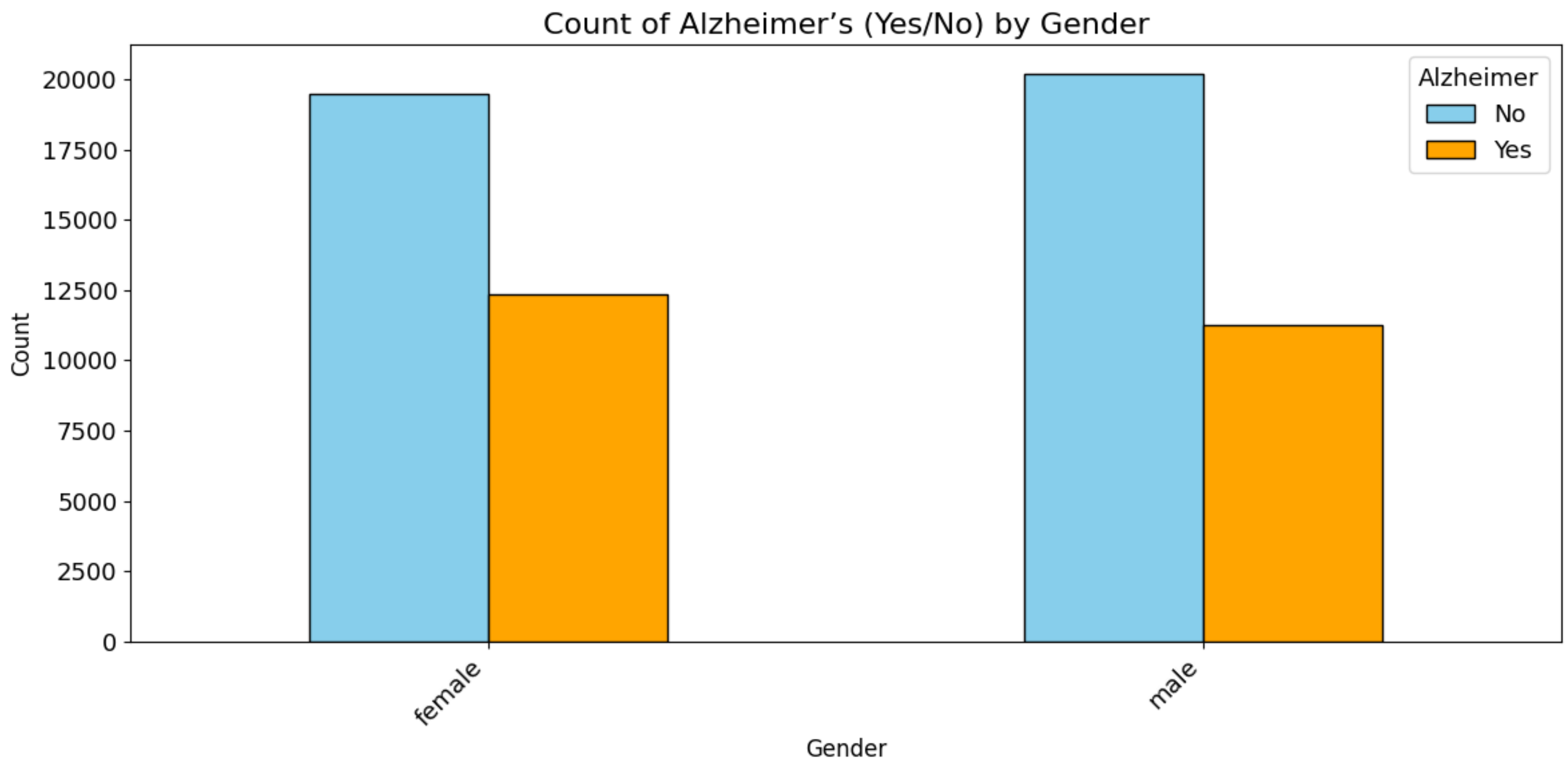
- ❑ Used count plots to study how Alzheimer's varies across racial and ethnic groups.
- ❑ Developed grouped bar plots to analyze the distribution of Alzheimer's cases across different racial and ethnic groups.
- ❑ This visualization highlights disparities and prevalence patterns across different communities.



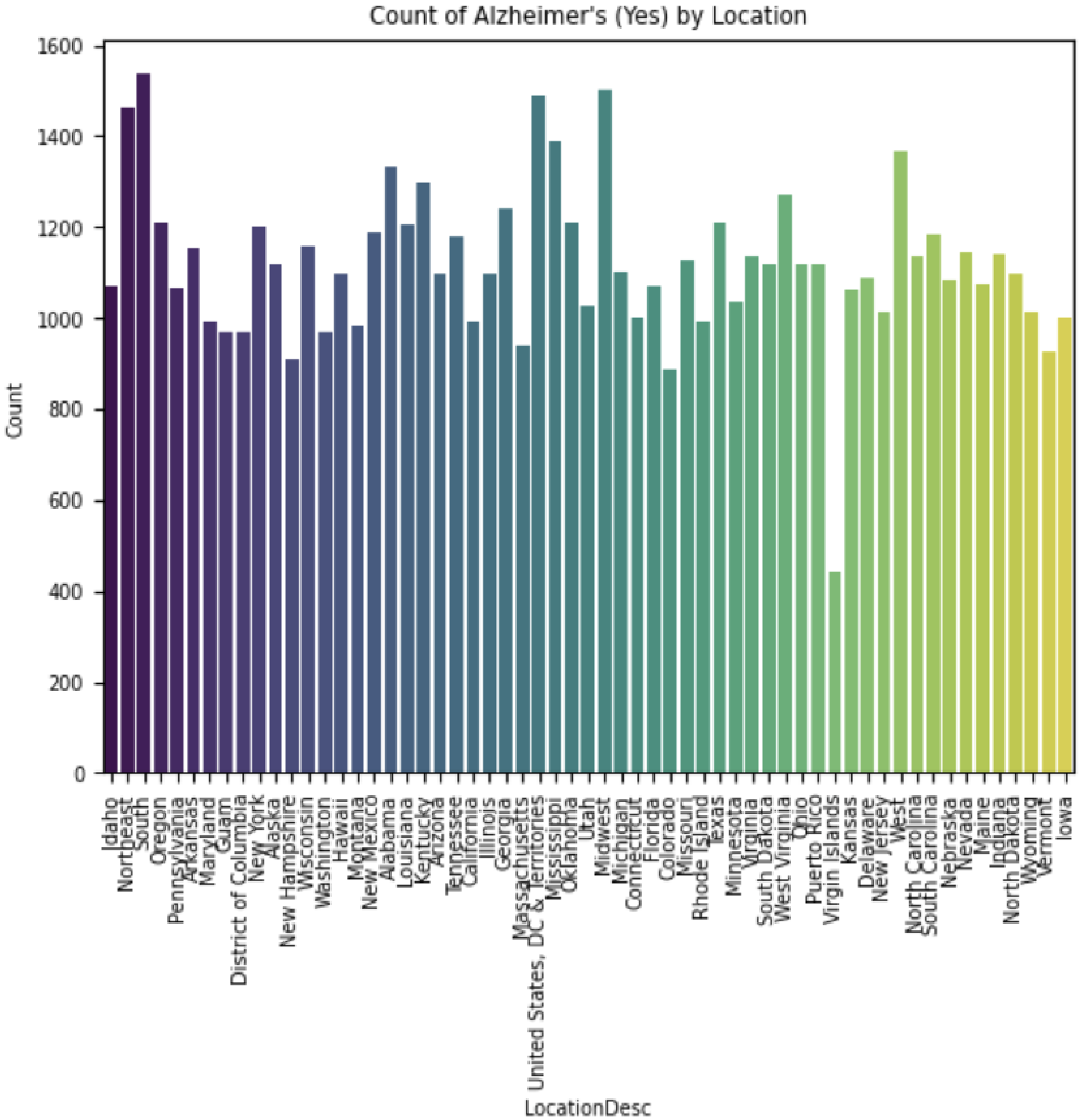


Focus on Alzheimer’s by Gender

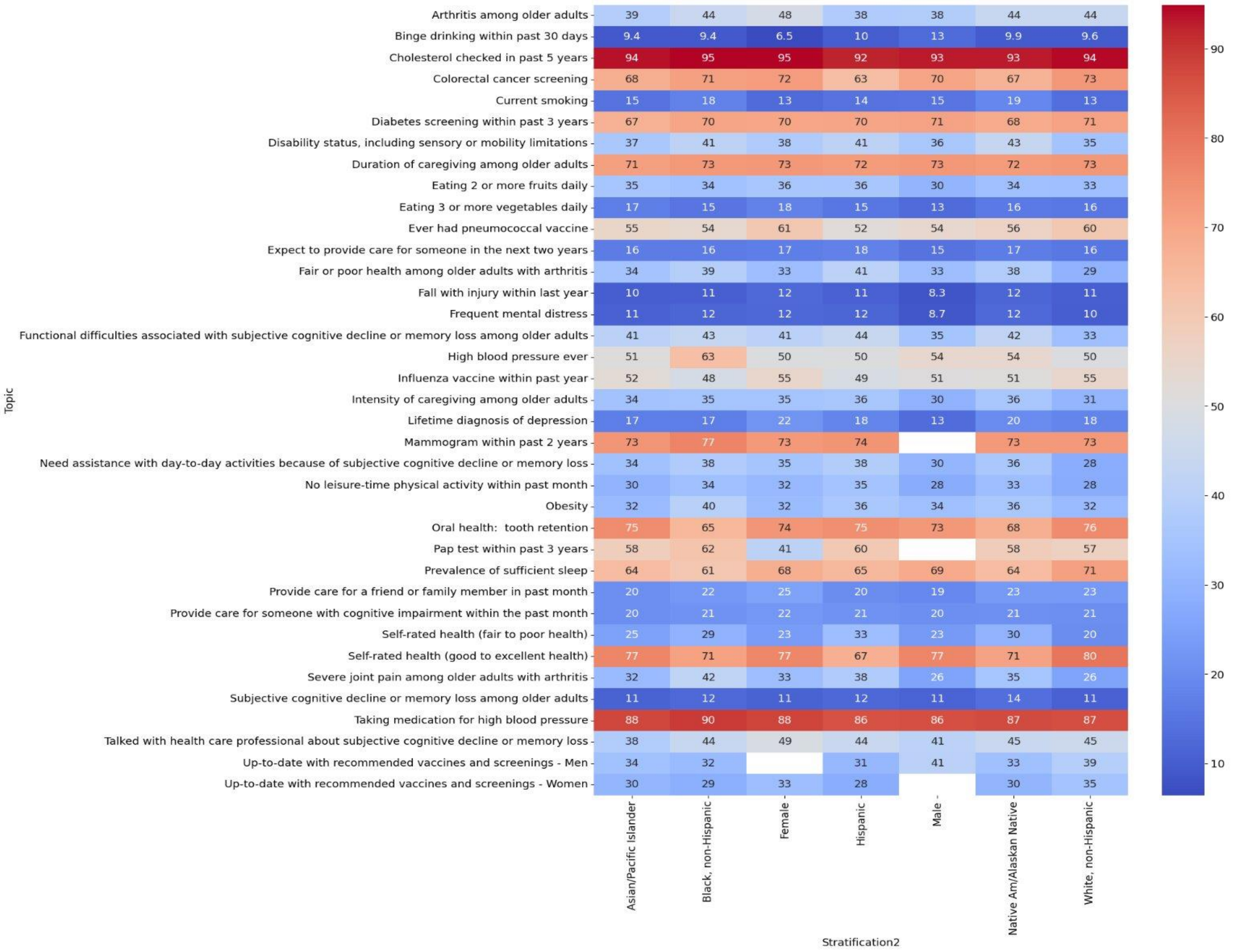
- Distribution of Alzheimer’s cases where the condition is “Yes” or “No”



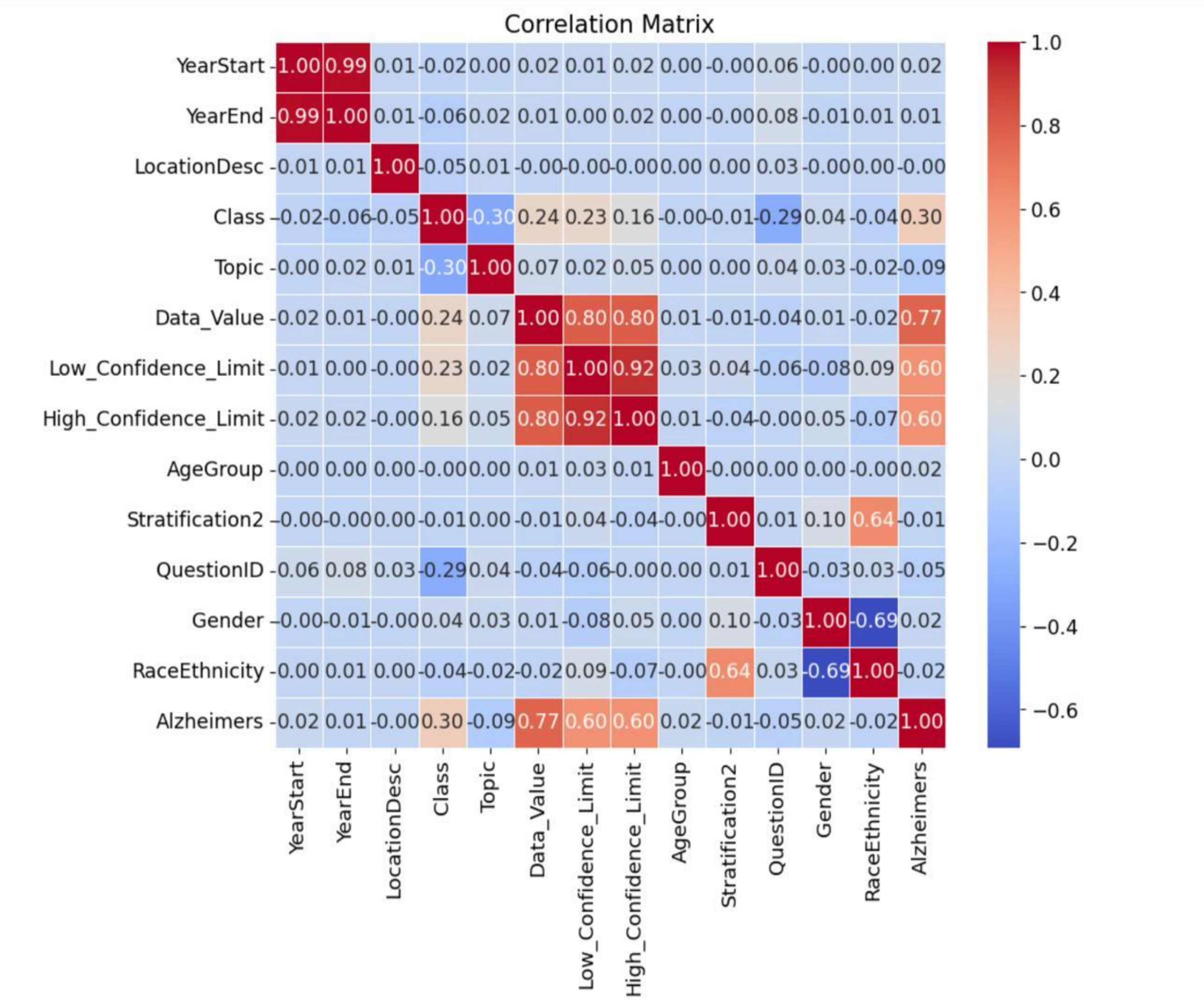
```
LocationDesc
South                               1535
Midwest                             1503
United States, DC & Territories     1490
Northeast                           1462
Mississippi                         1388
West                                1365
Alabama                             1332
Kentucky                            1298
West Virginia                       1273
Georgia                             1241
Texas                               1212
Oklahoma                            1211
Oregon                              1209
Louisiana                           1204
New York                            1203
New Mexico                          1189
South Carolina                      1183
Tennessee                           1180
Wisconsin                           1158
Arkansas                            1155
Nevada                              1145
Indiana                             1142
North Carolina                      1138
Virginia                            1134
Missouri                            1129
South Dakota                        1120
Puerto Rico                         1118
Ohio                                1117
Alaska                              1117
Michigan                            1103
Arizona                             1098
North Dakota                        1096
Hawaii                              1095
Illinois                            1095
Delaware                            1088
Nebraska                            1084
Maine                               1076
Florida                             1071
Idaho                               1070
Pennsylvania                        1068
Kansas                              1061
Minnesota                           1036
Utah                                1029
Wyoming                             1013
New Jersey                          1012
Iowa                                1002
Connecticut                         1001
Rhode Island                         994
Maryland                             993
California                           992
Montana                              985
Washington                           972
District of Columbia                 972
Guam                                 968
Massachusetts                        938
Vermont                              926
New Hampshire                        911
Colorado                             889
Virgin Islands                       442
```



Topic-Wise Data Insights Using a Heatmap:



Correlation Matrix Calculation:



[77…

```python
import pandas as pd
import matplotlib.pyplot as plt

# Assuming df is already defined and contains your dataset

print("Dataset Shape:", df.shape)
print("Column Names:", df.columns)

if 'Alzheimers' in df.columns:
    print("Unique values in 'Alzheimers':", df['Alzheimers'].unique())
else:
    raise ValueError("'Alzheimers' column not found!")

# Filter out rows where 'Alzheimers' is null
df = df[df['Alzheimers'].notnull()]

# Map 'Yes' to 1 and 'No' to 0
df['Alzheimers'] = df['Alzheimers'].map({'Yes': 1, 'No': 0})

# Check if the conversion was successful
print("Unique values in 'Alzheimers' after mapping:", df['Alzheimers'].unique())

print("Class 0 samples:", len(df[df['Alzheimers'] == 0]))
print("Class 1 samples:", len(df[df['Alzheimers'] == 1]))

# Set sample size for balancing classes
sample_size = min(len(df[df['Alzheimers'] == 0]), len(df[df['Alzheimers'] == 1]), 113982)
print("Sample size:", sample_size)

# Sample from both classes
class_0 = df[df['Alzheimers'] == 0].sample(n=sample_size, random_state=42)
class_1 = df[df['Alzheimers'] == 1].sample(n=sample_size, random_state=42)

# Concatenate and shuffle the DataFrame
balanced_df = pd.concat([class_0, class_1]).sample(frac=1, random_state=42).reset_index(drop=True)

print("\nBalanced Class Distribution:\n", balanced_df['Alzheimers'].value_counts())

# Save the balanced DataFrame to a CSV file
balanced_df.to_csv('balanced_dataset.csv', index=False)
print("Balanced dataset saved to 'balanced_dataset.csv'.")
```

```
Dataset Shape: (227965, 16)
Column Names: Index(['YearStart', 'YearEnd', 'LocationDesc', 'Class', 'Topic', 'Question',
       'Data_Value_Type', 'Data_Value', 'Low_Confidence_Limit',
       'High_Confidence_Limit', 'AgeGroup', 'Stratification2', 'QuestionID',
       'Gender', 'RaceEthnicity', 'Alzheimers'],
      dtype='object')
Unique values in 'Alzheimers': ['No' 'Yes']
Unique values in 'Alzheimers' after mapping: [0 1]
Class 0 samples: 133157
Class 1 samples: 94808
Sample size: 94808

Balanced Class Distribution:
 Alzheimers
0    94808
1    94808
Name: count, dtype: int64
Balanced dataset saved to 'balanced_dataset.csv'.
```

Modelling

```python
#Modeling

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder

selected_features = ['Alzheimers', 'Data_Value', 'AgeGroup', 'YearStart', 'YearEnd', 'Stratification2']
X = balanced_df[selected_features]
y = balanced_df['Alzheimers']


X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.3, random_state=42, stratify=y)


X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5, random_state=42, stratify=y_temp)


print(f"Training set: {X_train.shape}, {y_train.shape}")
print(f"Validation set: {X_val.shape}, {y_val.shape}")
print(f"Test set: {X_test.shape}, {y_test.shape}")
```

✓ 0.1s

```
Training set: (159575, 6), (159575,)
Validation set: (34195, 6), (34195,)
Test set: (34195, 6), (34195,)
```

Model Evaluation and Selection

Model Evaluation:

- Applied several machine learning algorithms: Random Forest, KNN, SVM and Logistic Regression
- Feature selection and data imputation were performed to prepare the dataset for robust analysis.

Insights into Model Performance :

- Some models showed signs of overfitting, making unreliable for unseen data.
- KNN and Random Forest showed strong performance but overfitting and interpretability concerns came forward.

K-Nearest Neighbors(KNN):

- Achieved 98.68% Validation accuracy and 98.72% test accuracy, indicating strong performance.
- However, KNN may overfit easily in case of noisy data, affecting its reliability.

```
#KNN
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix


knn = KNeighborsClassifier()
knn.fit(X_train, y_train)


y_val_pred = knn.predict(X_val)
print("KNN_Validation Accuracy:", accuracy_score(y_val, y_val_pred))

y_test_pred = knn.predict(X_test)
print("Knn_Test Accuracy:", accuracy_score(y_test, y_test_pred))
```

✓ 3.2s

```
KNN_Validation Accuracy: 0.9868401813130575
Knn_Test Accuracy: 0.9872788419359555
```

Model Selection: Why SVM?

Support Vector Machine (SVM):

- The model shows consistency in results with a validation accuracy of 82.31%
- Precision, recall and F1-scores of the model provide evidence that it generalizes well with no overfitting.
- The usage of SVM is highly reliable in this case as the model is robust in handling high dimensional data and works great when the dataset size is low.

Why?

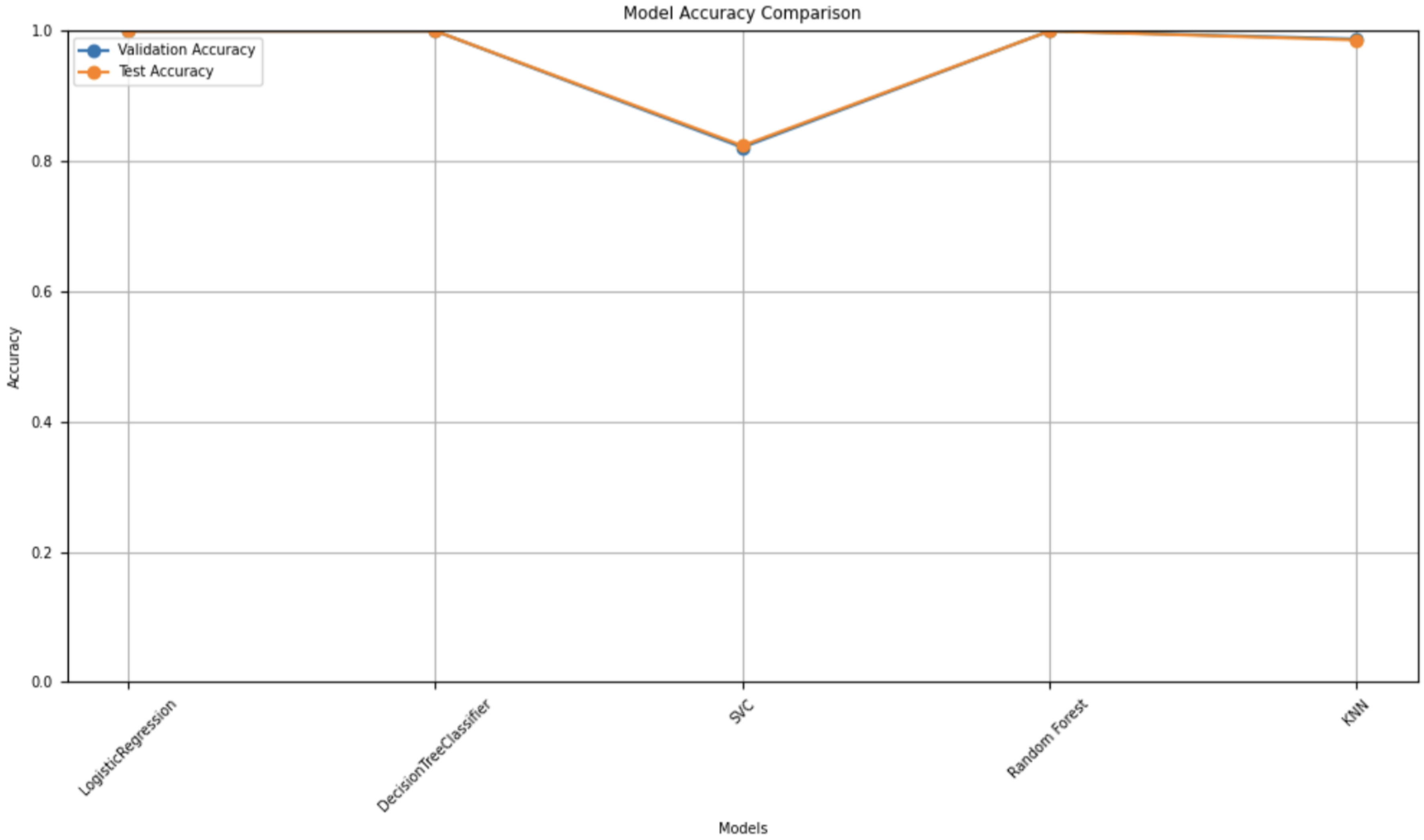
- Because the other models overfitted or didn't generalize well and hence cannot be very reliable on unseen data.
- KNN and Random Forest did a great job but raised some flags on overfitting.
- SVM was chosen since it balanced the trade-off between accuracy, generalization, and reliability.
- This ability of the model to avoid overfitting and provide robust predictions across datasets made it the fittest for our problem.

```python
from sklearn.svm import SVC

# Support Vector Machine Classifier
svm = SVC(random_state=42)
svm.fit(X_train, y_train)


y_val_pred = svm.predict(X_val)
print("SVC_Validation Accuracy:", accuracy_score(y_val, y_val_pred))

y_test_pred = svm.predict(X_test)
print("SVC_Test Accuracy:", accuracy_score(y_test, y_test_pred))


print("\nClassification Report (Validation):")
print(classification_report(y_val, y_val_pred))

print("\nConfusion Matrix (Validation):")
print(confusion_matrix(y_val, y_val_pred))
```

✓ 21m 12.5s

```
SVC_Validation Accuracy: 0.8231320368474924
SVC_Test Accuracy: 0.8217868109372716

Classification Report (Validation):
              precision    recall  f1-score   support

           0       0.81      0.92      0.86     19973
           1       0.86      0.69      0.76     14222

    accuracy                           0.82     34195
   macro avg       0.83      0.80      0.81     34195
weighted avg       0.83      0.82      0.82     34195
```

Model Accuracy Comparision



Prediction Results :

- The model predicts possibility of Alzheimer’s disease based on given features of demographic and cognitive health metrics.
- For this prediction, based on the input features, the model predicts no signs of Alzheimer’s, hence the individual is cognitively healthy and safe.
- These results pinpoint the model’s capability for reliable insights into early detection and informed decision making

```
svm_model=SVC(class_weight='balanced',random_state=42)
svm_model.fit(X_train,y_train)
if svm_model.predict([X_test.iloc[2]]):
  print("Indication of Alzheimer's disease detected")
else:
  print("No indication of Alzheimer's disease")
```

```
No indication of Alzheimer's disease
```

```
svm_model=SVC(class_weight='balanced',random_state=42)
svm_model.fit(X_train,y_train)
if svm_model.predict([X_test.iloc[25]]):
  print("Indication of Alzheimer's disease detected")
else:
  print("No indication of Alzheimer's disease")
```

```
Indication of Alzheimer's disease detected
c:\Users\Thanishma\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\base.py:464: UserWarning: X does r
  warnings.warn(
```

References: https://docs.google.com/document/d/1X-RHfXRFxaBbKhgfoEd_gVrKFLEe72a0h-9EYsIxaZM/edit?usp=sharing

Thank you