UNIVERSITÀ DEGLI STUDI DI NAPOLI
FEDERICO II

Data Mining Project by Yahya Momtaz

Heart Attack Analysis and Prediction

Professors:

Mod A: Professor Roberta Siciliano
Mod B: Professor Giuseppe Longo

University of Naples Federico II

Introduction

According to the world health organization, Cardiovascular diseases (CVDs) are the leading cause of death globally. In 2019 alone, around 17.9 million people died from CVDs. Of these deaths, **85%** of them were due to heart diseases. There are many factors that play a role in increasing the risk of heart disease. Identifying these factors and their impact is paramount in the field of healthcare. Identifying patients who are at greater risk enables medical professionals to respond quickly and efficiently, saving more lives.

About the Dataset:

The Personal Key Indicators of Heart Disease dataset contains 320K rows and 18 columns. It is a cleaned, smaller version of the 2020 annual CDC (Centers for Disease Control and Prevention) survey data of 400k adults. For each patient (row), it contains the health status of that individual. The data was collected in the form of surveys conducted over the phone. Each year, the CDC calls around 400K U.S residents and asks them about their health status, with the vast majority of questions being yes or no questions.

DataSet Features

#	Feature	Description
1	HeartDisease	Respondents that have ever reported having coronary heart disease (CHD) or myocardial infarction (MI)
2	BMI	Body Mass Index (BMI)
3	Smoking	Have you smoked at least 100 cigarettes in your entire life? [Note: 5 packs = 100 cigarettes]
4	AlcoholDrinking	Heavy drinkers (adult men having more than 14 drinks per week and adult women having more than 7 drinks per week)
5	Stroke	(Ever told) (you had) a stroke?
6	PhysicalHealth	Which includes physical illness and injury, for how many days during the past 30 days was your physical health not good?
7	MentalHealth	For how many days during the past 30 days was your mental health not good?
8	DiffWalking	Do you have serious difficulty walking or climbing stairs?
9	Sex	Are you male or female?
10	AgeCategory	Fourteen-level age category
11	Race	Imputed race/ethnicity value
12	Diabetic	(Ever told) (you had) diabetes?
13	PhysicalActivity	Adults who reported doing physical activity or exercise during the past 30 days other than their regular job
14	GenHealth	Would you say that in general your health is...
15	SleepTime	On average, how many hours of sleep do you get in a 24-hour period?
16	Asthma	(Ever told) (you had) asthma?
17	KidneyDisease	Not including kidney stones, bladder infection or incontinence, were you ever told you had kidney disease?
18	SkinCancer	(Ever told) (you had) skin cancer?
19	Height	How tall are you without shoes?
20	Weight	How much do you weigh without shoes?

Data Preprocessing

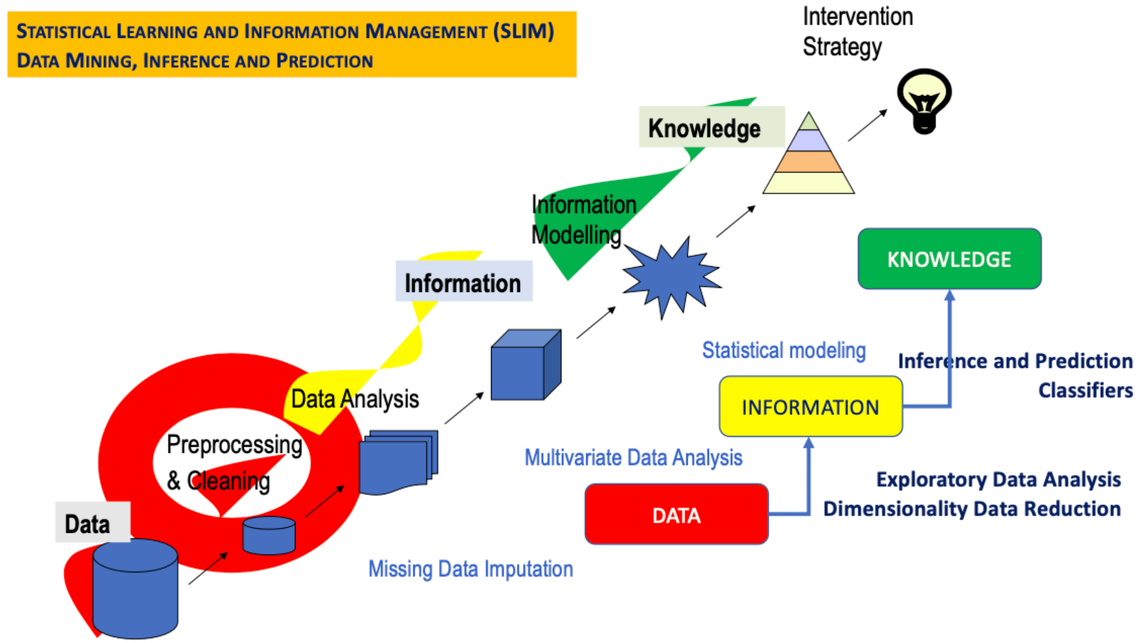
Data Cleaning: Handling Missing Values: Missing data points were identified and addressed. Depending on the extent of missing data in each column, we applied appropriate strategies such as removal and advanced imputation methods.

Handling Outliers: Outliers in the data were identified and treated through techniques such as trimming, transformation, or removal.

Feature Engineering: Such as Creating New Features: New features were engineered to potentially enhance the predictive power of the model.

Value Mapping: Specific values in the dataset were mapped to new values to ensure consistency and to address irregularities.

Data Visualization: Exploratory Data Analysis (EDA): Visualizations were created to gain insights into the data, identify patterns, and understand the relationships between features and the target variable.



Columns Mapping

```
columns_name_mapping = {
    "cvdcrhd4": "HeartDisease",
    "weight2": "Weight",
    "height3":"Height",
    "_smoker3":"Smoking",
    "drnkany5":"AlcoholDrinking",
    "cvdstrk3":"Stroke",
    "physhlth":"PhysicalHealth",
    "menthlth":"MentalHealth",
    "diffwalk":"DiffWalking",
    "_sex":"Sex",
    "_ageg5yr":"AgeCategory",
    "_race":"Race",
    "diabete4":"Diabetic",
    "exerany2":"PhysicalActivity",
    "genhlth":"GenHealth",
    "sleptim1":"SleepTime",
    "_asthms1":"Asthma",
    "chckdny2":"KidneyDisease",
    "chcscncr":"SkinCancer",
    "_bmi5":"BMI"
}
```

Calculation of Variables

Section 2: Healthy Days		
_MENT14D ***Calculated variable for 3 level not good mental health status: 0 days, 1-13 days, 14-30 days.*** **_MENT14D is derived from MENTHLTH.**		
1	Zero days when mental health not good	Respondents who reported no days when their mental health was not good (MENTHLTH=88)
2	1-13 days when mental health not good	Respondents who reported 1-13 days when their mental health was not good (1 <= MENTHLTH <= 13)
3	14+ days when mental health not good	Respondents who reported 14 or more days when their mental health was not good (14 <= MENTHLTH <=30)
9	Don't know/ Refused/ Missing	Respondents who reported they didn't know, refused or had missing values for MENTHLTH (MENTHLTH=77,99, or missing)

	SAS Code:	`IF MENTHLTH IN (77,99,.) THEN _MENT14D=9;`
		`ELSE IF MENTHLTH=88 THEN _MENT14D=1;`
		`ELSE IF 1 LE MENTHLTH LE 13 THEN _MENT14D=2;`
		`ELSE _MENT14D=3;`

Section 2: Healthy Days		
_PHYS14D ***Calculated variable for 3 level not good physical health status: 0 days, 1-13 days, 14-30 days.*** **_PHYS14D is derived from PHYSHLTH.**		
1	Zero days when physical health not good	Respondents who reported no days when their physical health was not good (PHYSHLTH=88)
2	1-13 days when physical health not good	Respondents who reported 1-13 days when their physical health was not good (1 <= PHYSHLTH <= 13)
3	14+ days when physical health not good	Respondents who reported 14 or more days when their physical health was not good (14 <= PHYSHLTH <=30)
9	Don't know/ Refused/ Missing	Respondents who reported they didn't know, refused or had missing values for PHYSHLTH (PHYSHLTH=77,99, or missing)

	SAS Code:	`IF PHYSHLTH IN (77,99,.) THEN _PHYS14D=9;`
		`ELSE IF PHYSHLTH=88 THEN _PHYS14D=1;`
		`ELSE IF 1 LE PHYSHLTH LE 13 THEN _PHYS14D=2;`
		`ELSE _PHYS14D=3;`

Section 5: Inadequate Sleep
There are no calculated variables for Section 5.

Section 8: Demographics		
HTIN4 ***Calculated variable for reported height in inches.*** **HTIN4 is derived from HEIGHT3. HTIN4 is calculated by adding the foot portion of HEIGHT3 multiplied by 12, to the inch portion.**		
36 - 95	Height in inches	Respondents calculated height in inches. (HTIN4=(height in feet x 12) + height in inches)
.	Don't know/ Refused/ Not asked or Missing	Respondents who reported they didn't know, were not sure, refused or had missing responses for their height.

	SAS Code:	`IF 300<=HEIGHT3<=311 THEN HTIN4=((HEIGHT3-300)+36);`
		`ELSE IF 400<=HEIGHT3<=411 THEN HTIN4=((HEIGHT3-400)+48);`
		`ELSE IF 500<=HEIGHT3<=511 THEN HTIN4=((HEIGHT3-500)+60);`
		`ELSE IF 600<=HEIGHT3<=611 THEN HTIN4=((HEIGHT3-600)+72);`
		`ELSE IF 700<=HEIGHT3<=711 THEN HTIN4=((HEIGHT3-700)+84);`

Section 8: Demographics		
WTKG3 ***Calculated variable for reported weight in kilograms.*** **WTKG3 is derived from WEIGHT2 by multiplying WEIGHT2 by 0.4535924 kg per lb.**		
2300 - 29500	Weight in kilograms [2 implied decimal places]	Respondents reported or calculated weight in kilograms.
.	Don't know/ Refused/ Not asked or Missing	Respondents who reported they didn't know, were not sure, or refused or had missing responses for their weight.

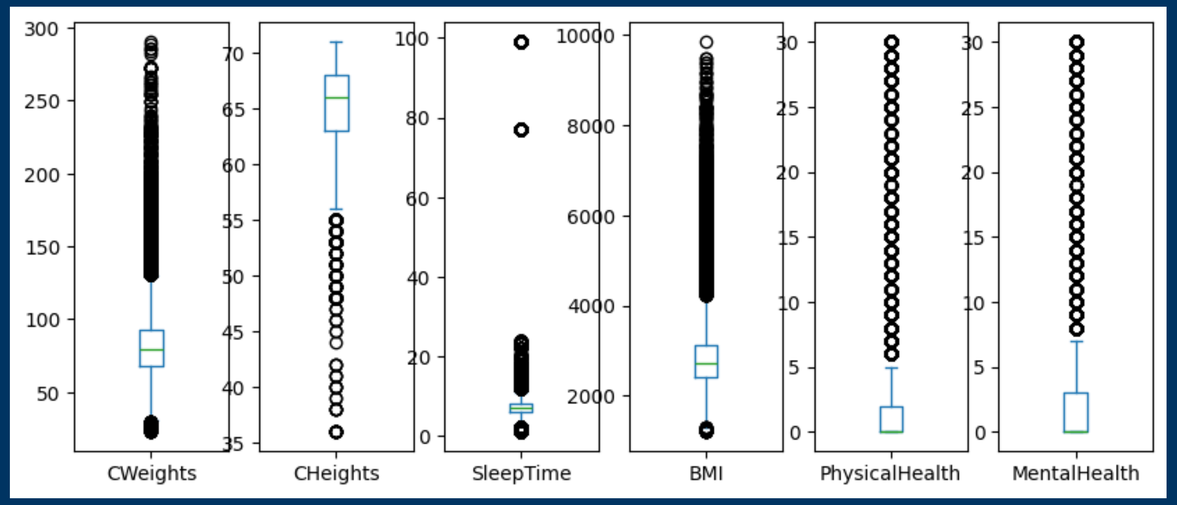
	SAS Code:	`** CONVERSION FACTOR = 0.4535924 kg/lb **;`
		` IF WEIGHT2 NOT IN (777,999,7777,9999,.) THEN DO;`
		` IF 0050 LE WEIGHT2 < 0650 THEN WTKG3=WEIGHT2*0.4535924;`
		` ELSE IF 9023 LE WEIGHT2 < 9295 THEN WTKG3=WEIGHT2-9000;`
		`END;`

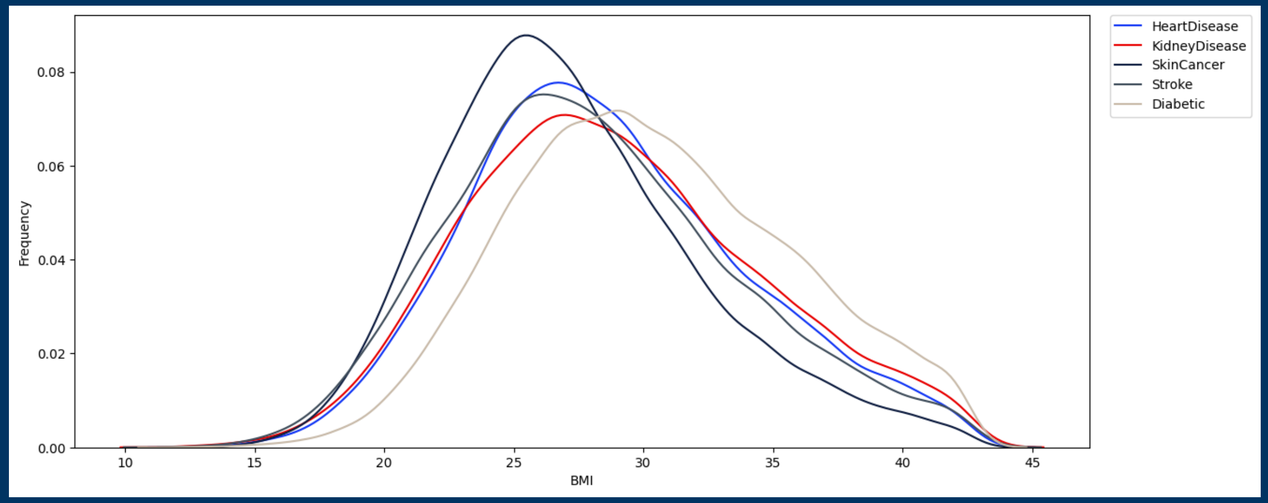
EDA

Numerical Variables Statistics

	CWeights	CHeights	SleepTime	BMI	PhysicalHealth	MentalHealth
count	367010.000000	316254.000000	395903.000000	290219.000000	393267.000000	394029.000000
mean	82.320386	65.747172	7.066529	27.598779	3.457595	3.916334
std	20.994057	3.306357	1.366156	5.299549	8.081710	8.045309
min	22.680000	36.000000	1.000000	12.400000	0.000000	0.000000
25%	68.040000	63.000000	6.000000	23.710000	0.000000	0.000000
50%	79.380000	66.000000	7.000000	26.930000	0.000000	0.000000
75%	92.990000	68.000000	8.000000	30.900000	2.000000	3.000000
max	290.300000	71.000000	12.000000	42.770000	30.000000	30.000000

- Underweight: BMI less than 18.5
- Normal weight: BMI 18.5 – 24.9
- Overweight: BMI 25 – 29.9
- Obesity: BMI 30 or greater



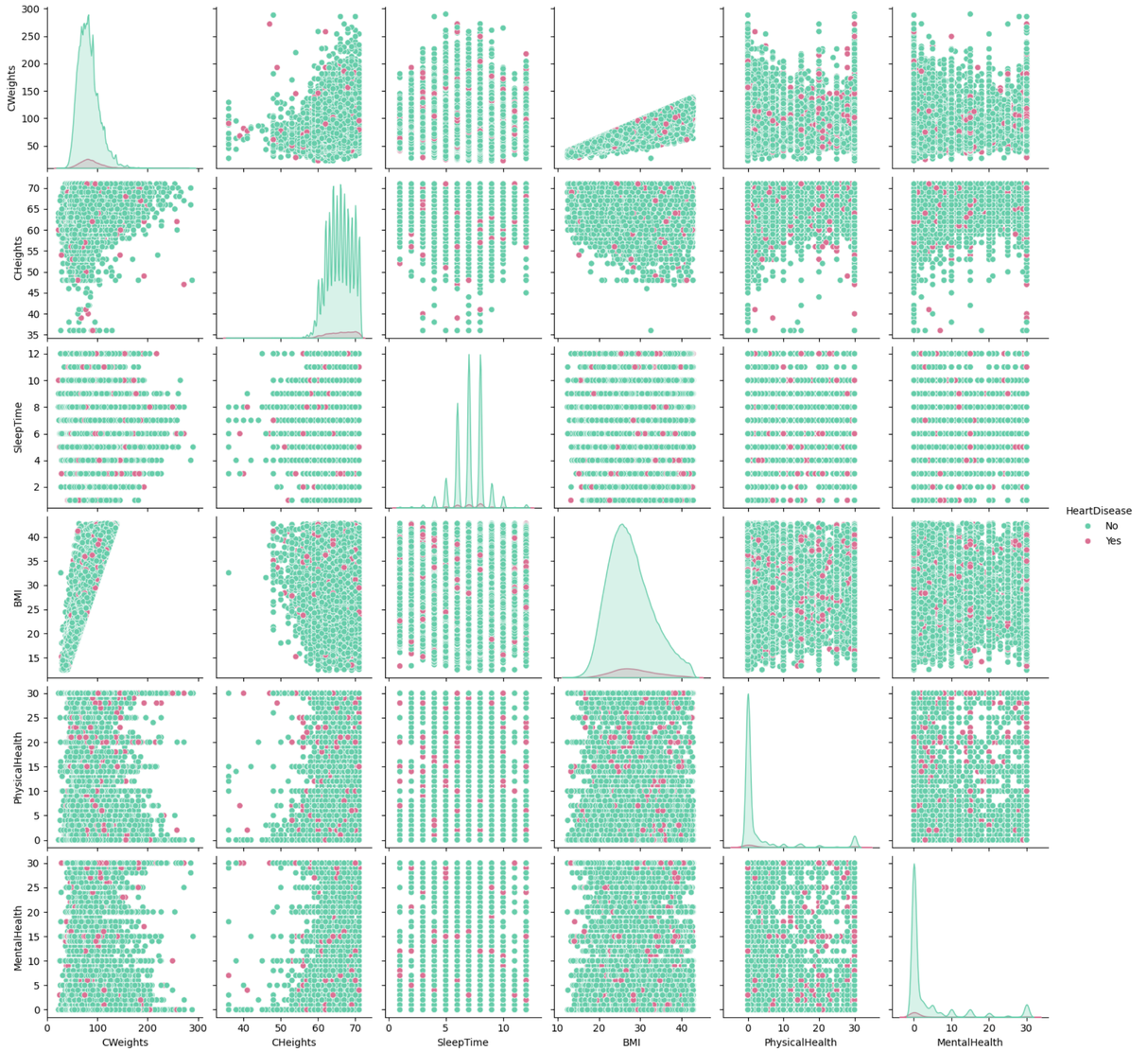


- **Weights vs. Heights**:
- There is a positive correlation between weight and height, as expected,
- but this does not seem to differ significantly by heart disease status.

- **Weights vs. BMI**:
- A clear positive relationship is observed here, which is expected since BMI is calculated from weight and height.
- Individuals with heart disease tend to have higher BMI values, which is visible in the plot.

- **Heights vs. BMI**:
- There is a weakly inverse relationship since BMI calculation inversely squares the height.
- No clear distinction is visible between the heart disease groups.

- **SleepTime vs. BMI**:
- The relationship between sleep time and BMI does not show a clear trend related to heart disease status.



Heart Disease Correlations:

‘KidneyDisease’ , ‘Stroke’ and ‘SkinCancer’ show relations with Hear Disease more than others.
There is a notable positive correlation between heart disease and stroke (0.22).

Lifestyle Factors:

Variables such as 'Smoking', 'AlcoholDrinking', 'PhysicalActivity', and 'BMI'
might show correlations with 'HeartDisease' that could be expected based on medical literature.
For example, a positive correlation with 'Smoking' would be consistent
with the known health risks associated with tobacco use.
These lifestyle factors have low correlation coefficients with heart disease in this dataset,
which could be a point of interest given that they are commonly known risk factors.

Medical Conditions:

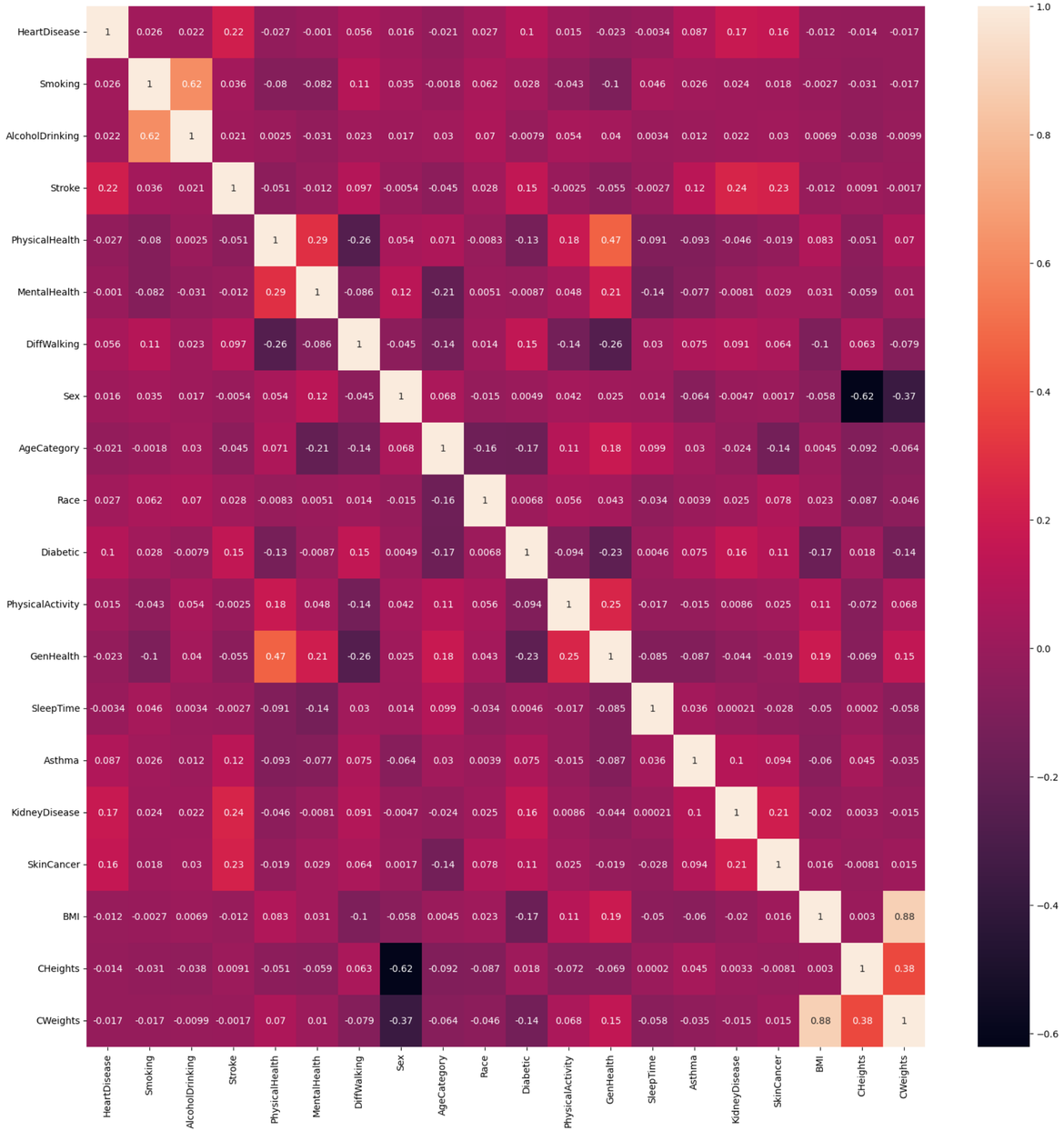
Conditions such as 'Stroke', 'Diabetic', 'Asthma', 'KidneyDisease',
and 'SkinCancer' are included and may show correlations with each other and with 'HeartDisease'.
These correlations could reflect common risk factors or co-morbidities.

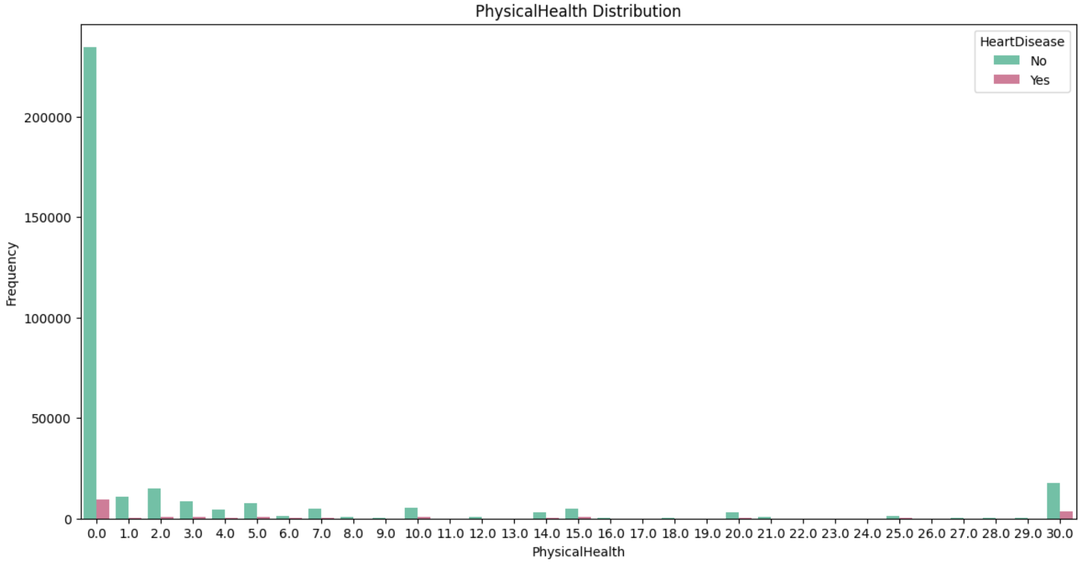
Demographic Factors:

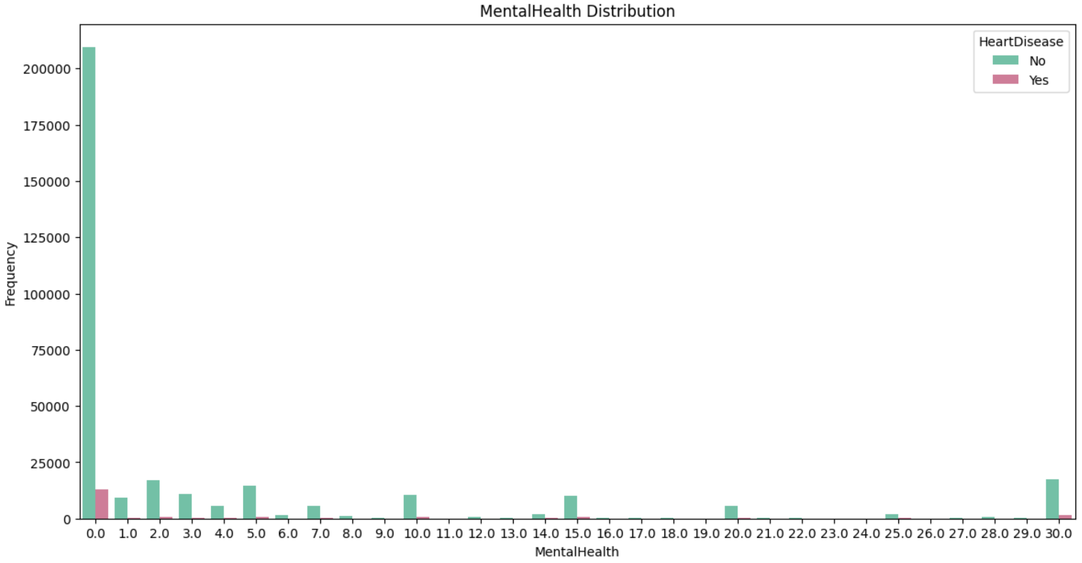
'AgeCategory', 'Sex', and 'Race' are included,
which might show correlations with health conditions,
reflecting known demographic patterns in health outcomes.

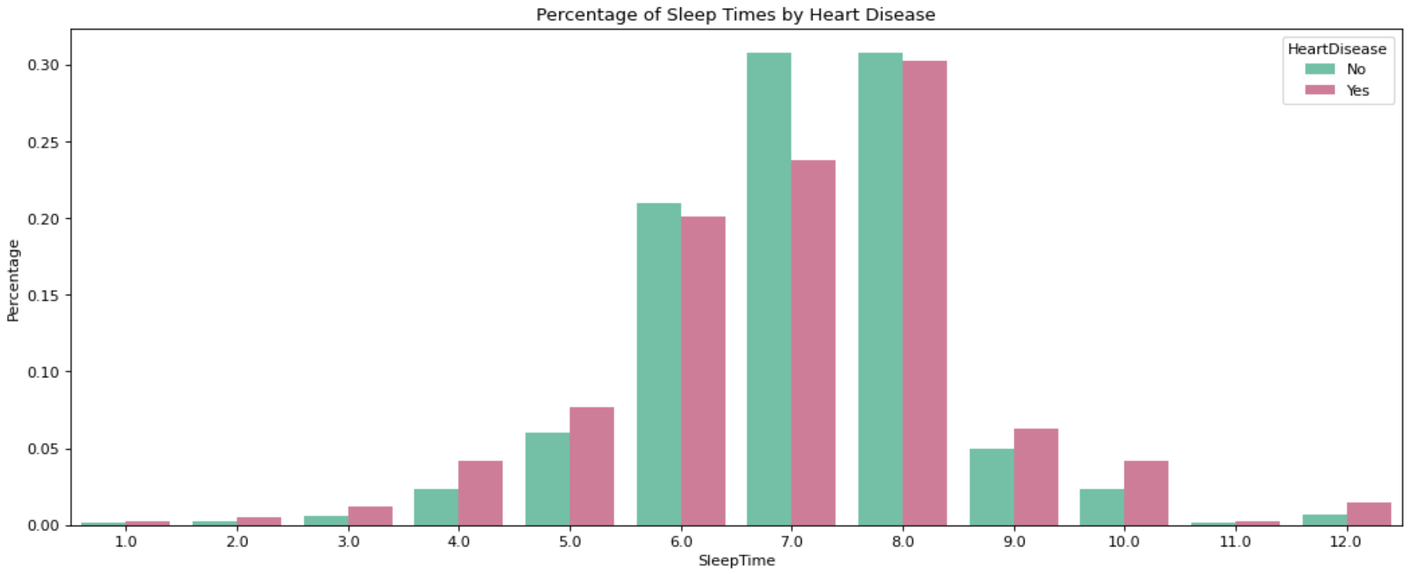
Physical Health and Mental Health:

There's a negative correlation between good physical/mental health and heart disease, as expected.

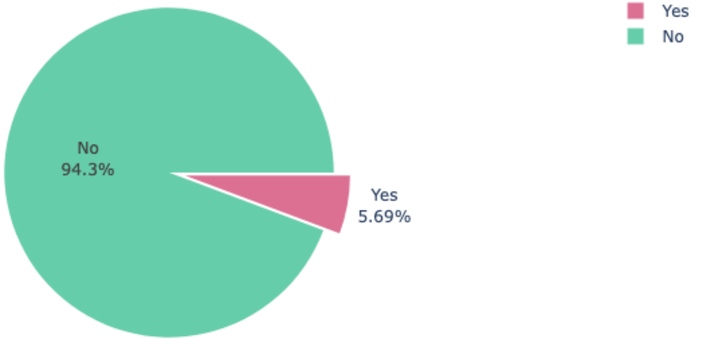



PhysicalHealth Distribution
HeartDisease
No
Yes
Frequency
PhysicalHealth

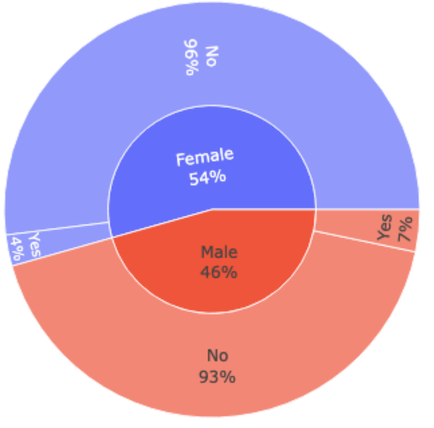

MentalHealth Distribution
HeartDisease
No
Yes
Frequency
MentalHealth

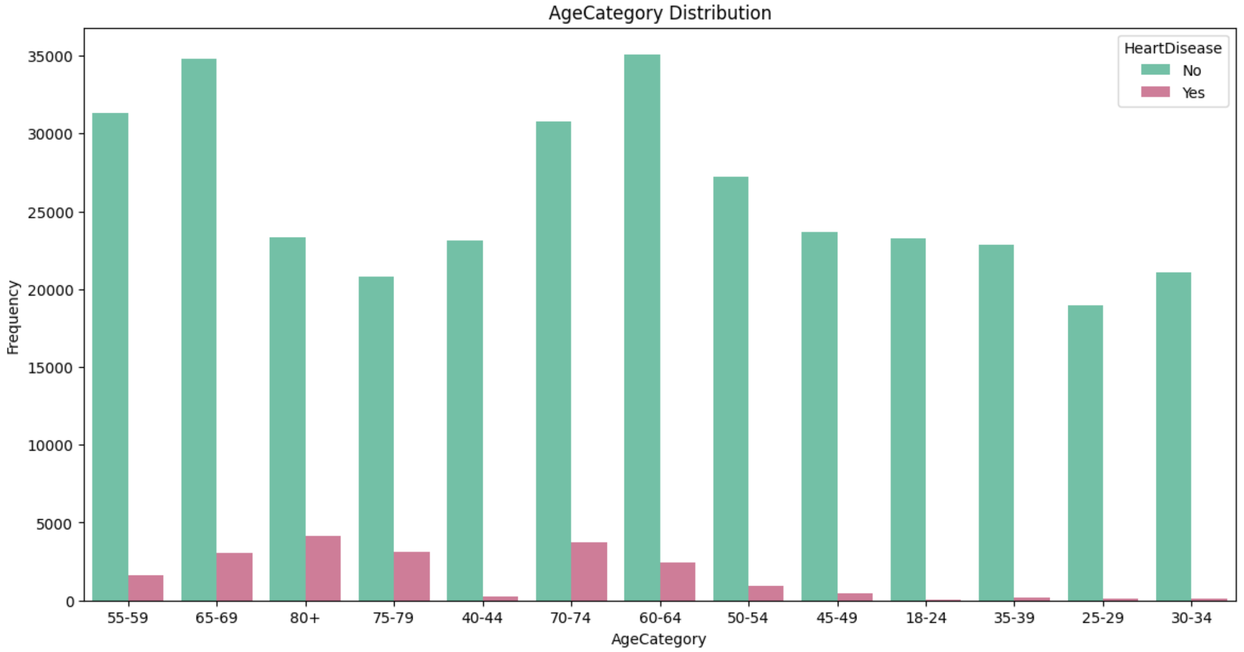

Percentage of Sleep Times by Heart Disease
HeartDisease
No
Yes
Percentage
SleepTime

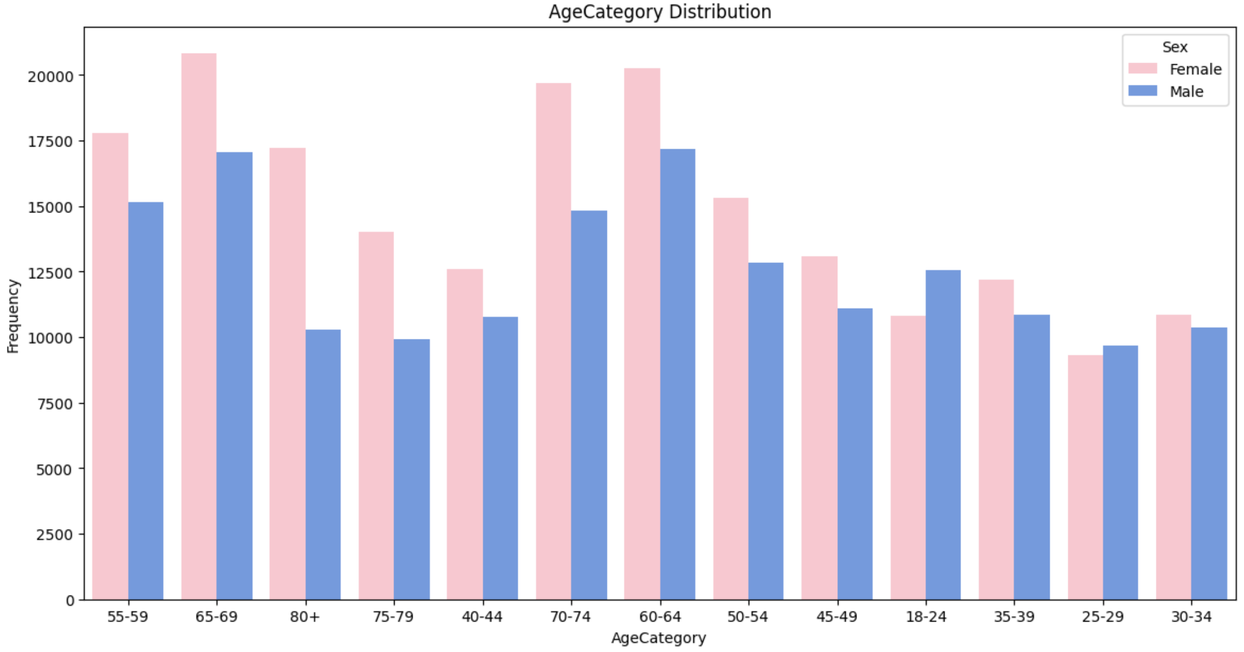
Heart Disease Ratio

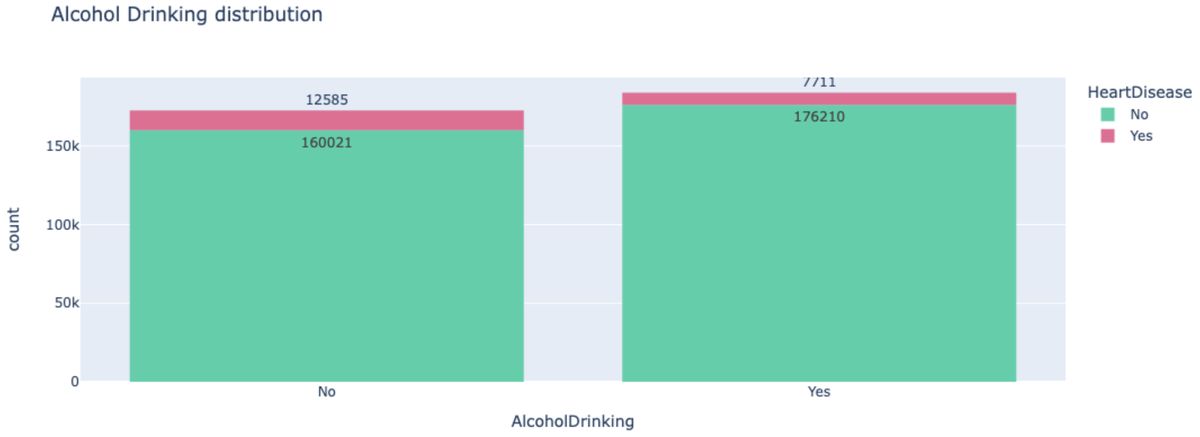


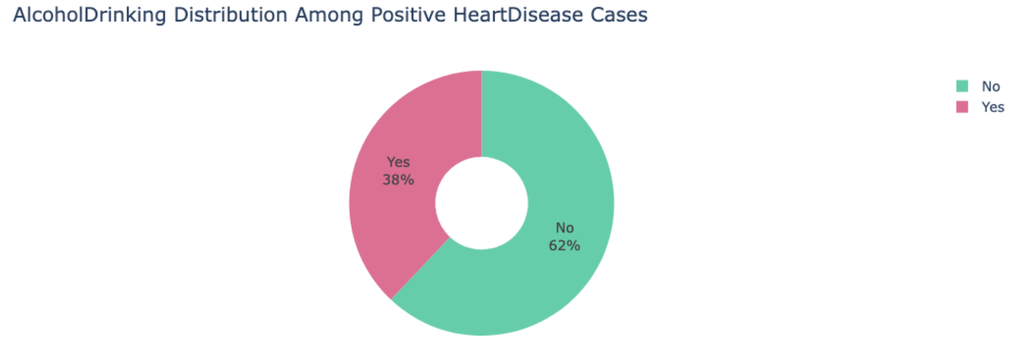
Sex Ratio with Heart Disease Percentage



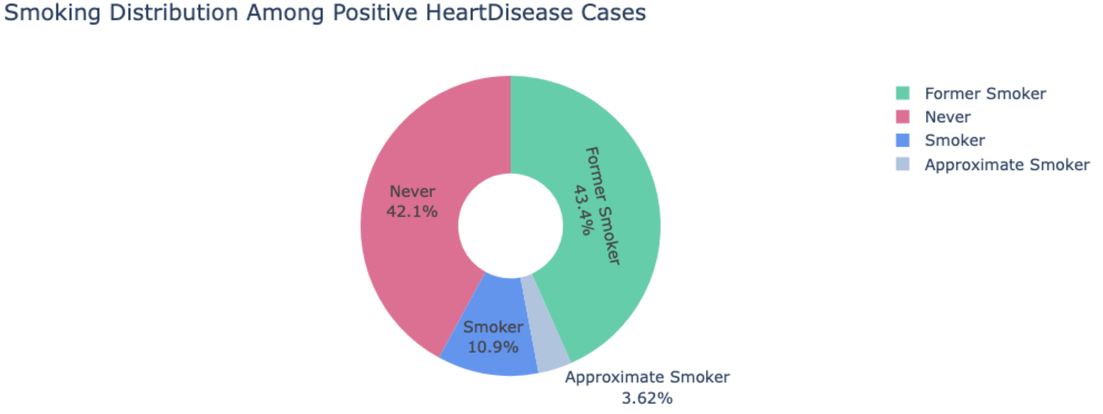

AgeCategory Distribution
HeartDisease
No
Yes
Frequency
0
5000
10000
15000
20000
25000
30000
35000
55-59
65-69
80+
75-79
40-44
70-74
60-64
50-54
45-49
18-24
35-39
25-29
30-34
AgeCategory

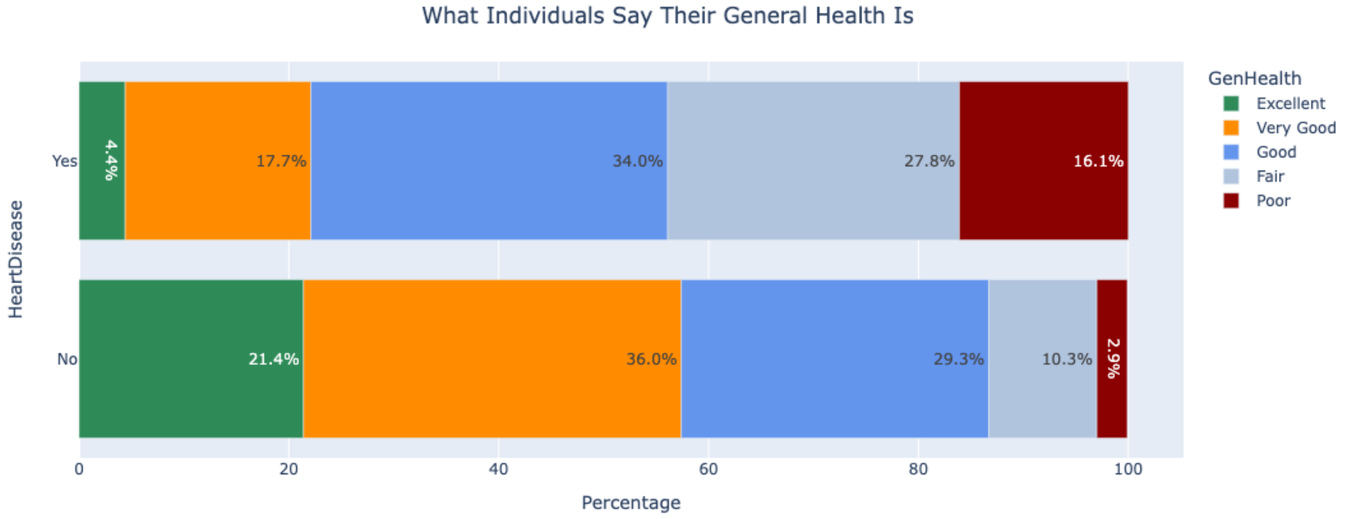

AgeCategory Distribution
Sex
Female
Male
Frequency
0
2500
5000
7500
10000
12500
15000
17500
20000
55-59
65-69
80+
75-79
40-44
70-74
60-64
50-54
45-49
18-24
35-39
25-29
30-34
AgeCategory

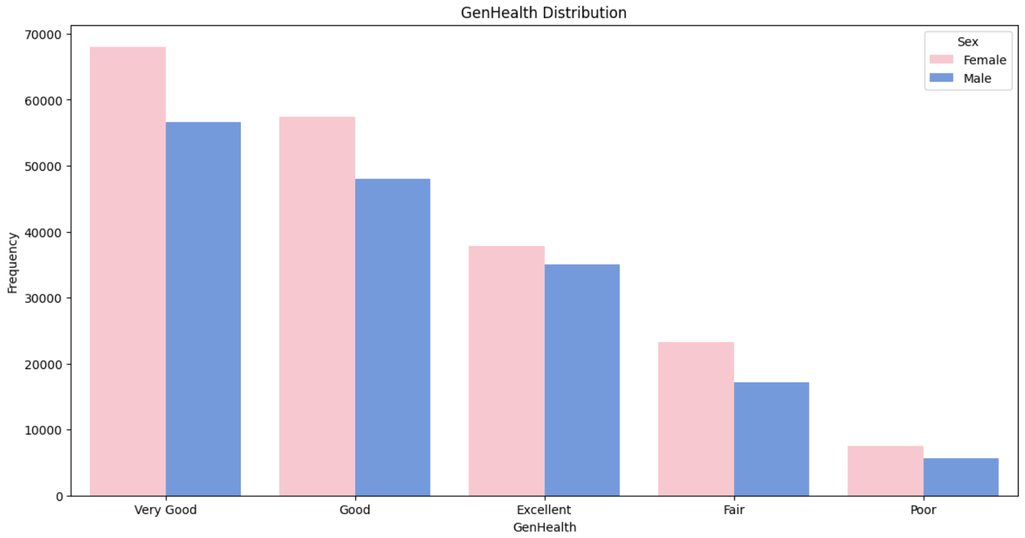

Alcohol Drinking distribution
12585
160021
7711
176210
150k
100k
50k
0
count
No
Yes
AlcoholDrinking
HeartDisease
No
Yes

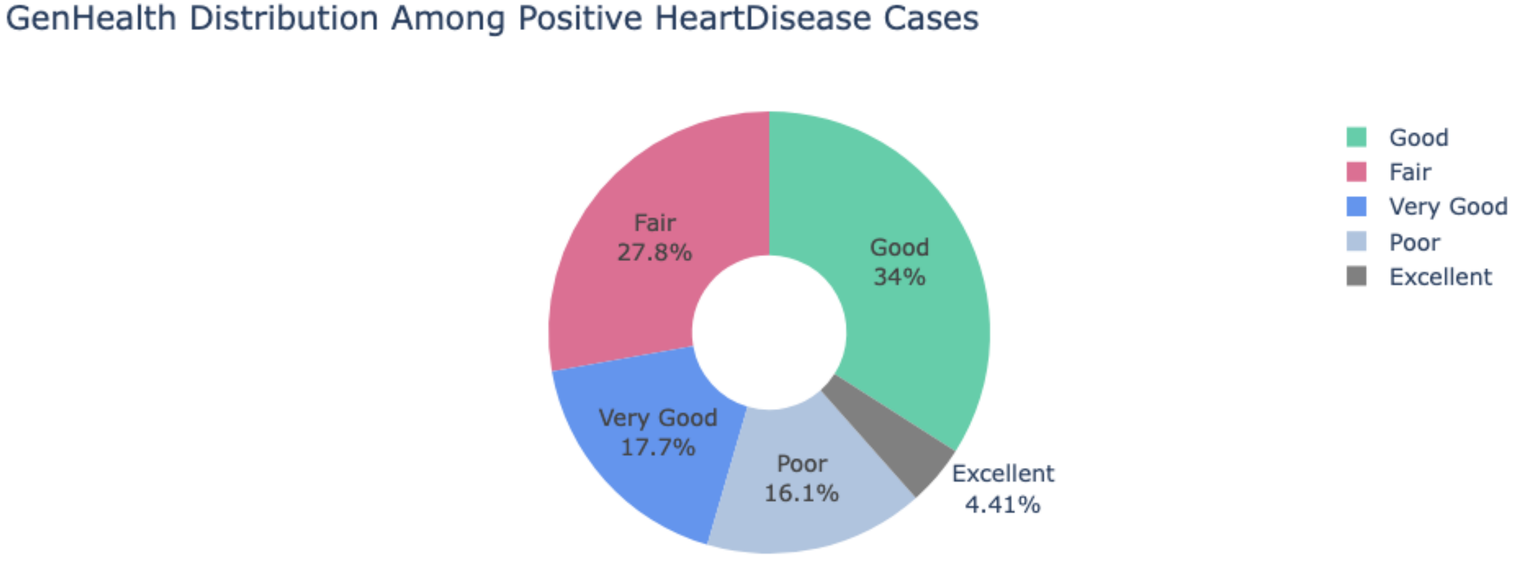

AlcoholDrinking Distribution Among Positive HeartDisease Cases
Yes
38%
No
62%
No
Yes


Smoking distribution
2216
33262
8539
202140
8806
88286
735
12543
200k
150k
100k
50k
0
count
Smoker
Never
Former Smoker
Approximate Smoker
Smoking
HeartDisease
No
Yes

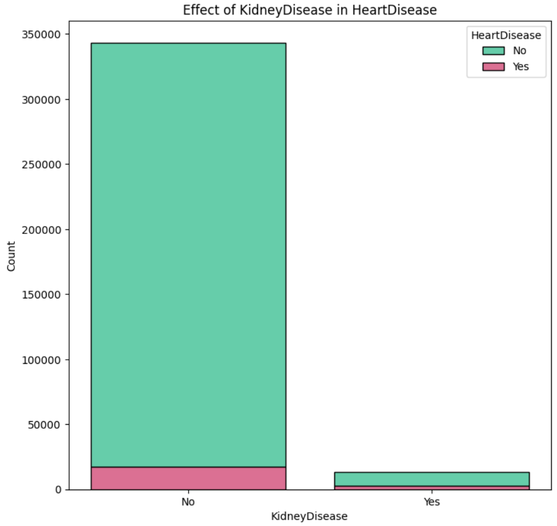

Smoking Distribution Among Positive HeartDisease Cases
Never
42.1%
Former Smoker
43.4%
Smoker
10.9%
Approximate Smoker
3.62%
Former Smoker
Never
Smoker
Approximate Smoker

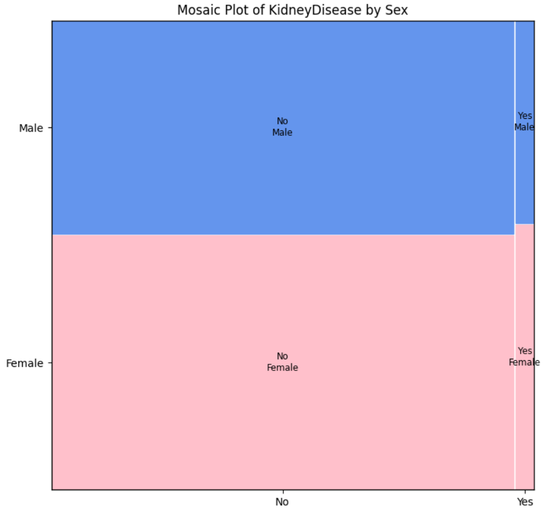

What Individuals Say Their General Health Is
HeartDisease
Yes
No
4.4%
17.7%
34.0%
27.8%
16.1%
21.4%
36.0%
29.3%
10.3%
2.9%
0
20
40
60
80
100
Percentage
GenHealth
Excellent
Very Good
Good
Fair
Poor

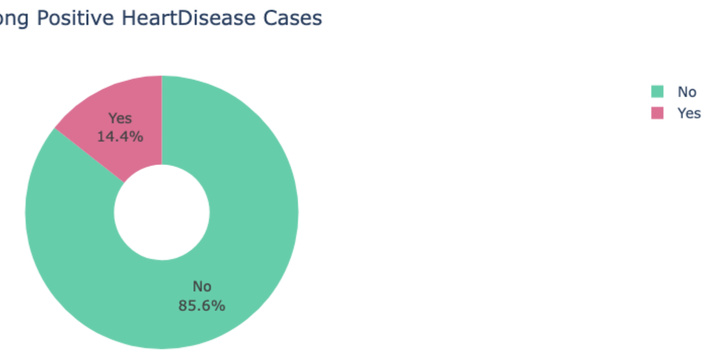

GenHealth Distribution
Frequency
70000
60000
50000
40000
30000
20000
10000
0
Very Good
Good
Excellent
Fair
Poor
GenHealth
Sex
Female
Male


GenHealth Distribution Among Positive HeartDisease Cases
Fair
27.8%
Good
34%
Very Good
17.7%
Poor
16.1%
Excellent
4.41%
Good
Fair
Very Good
Poor
Excellent

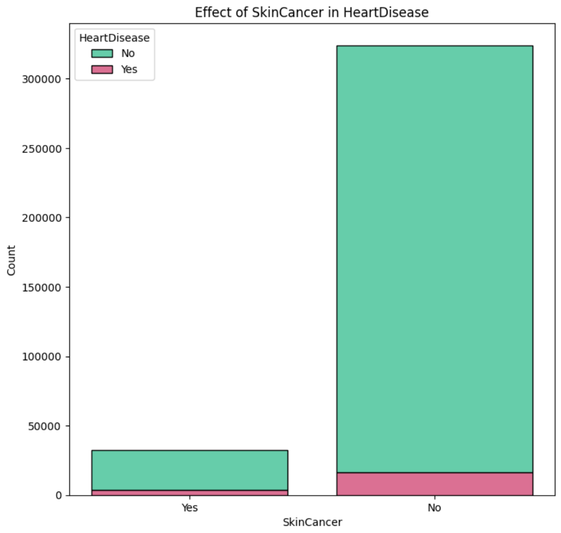
KidneyDisease Distribution

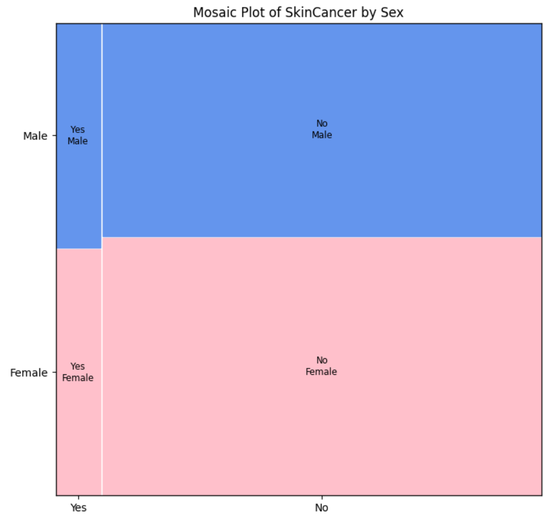


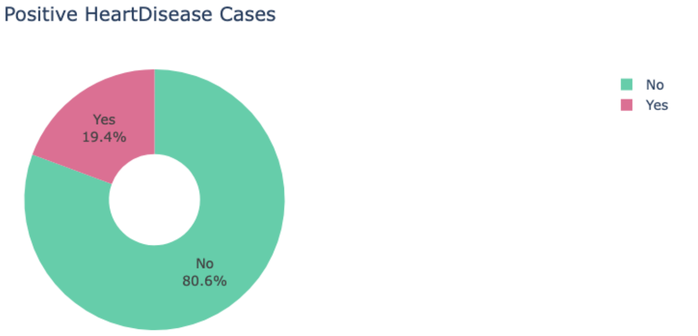




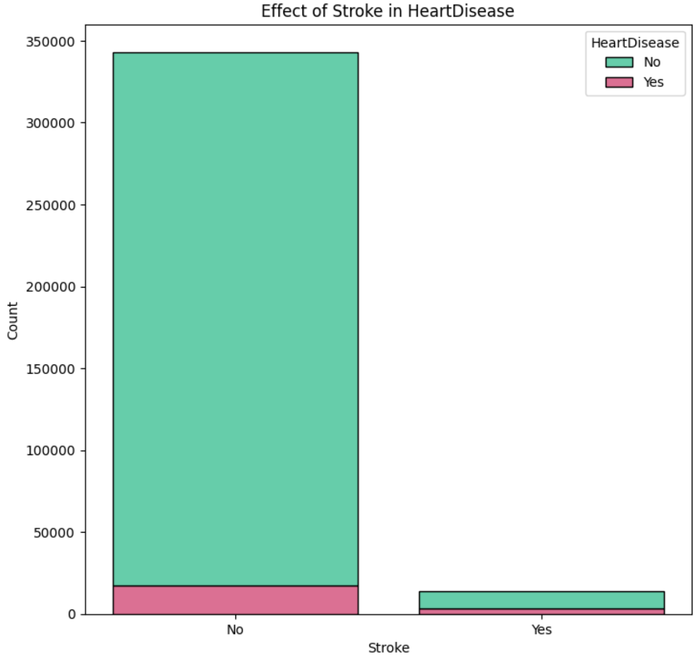
SkinCancer Distribution

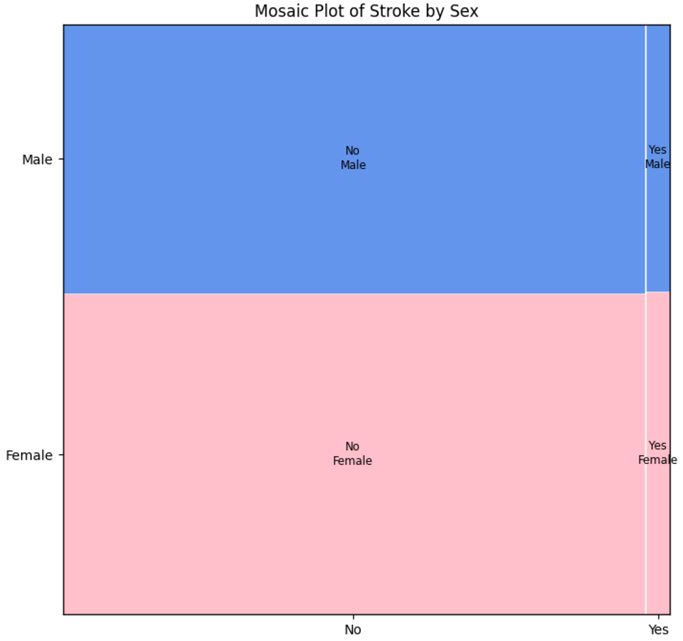


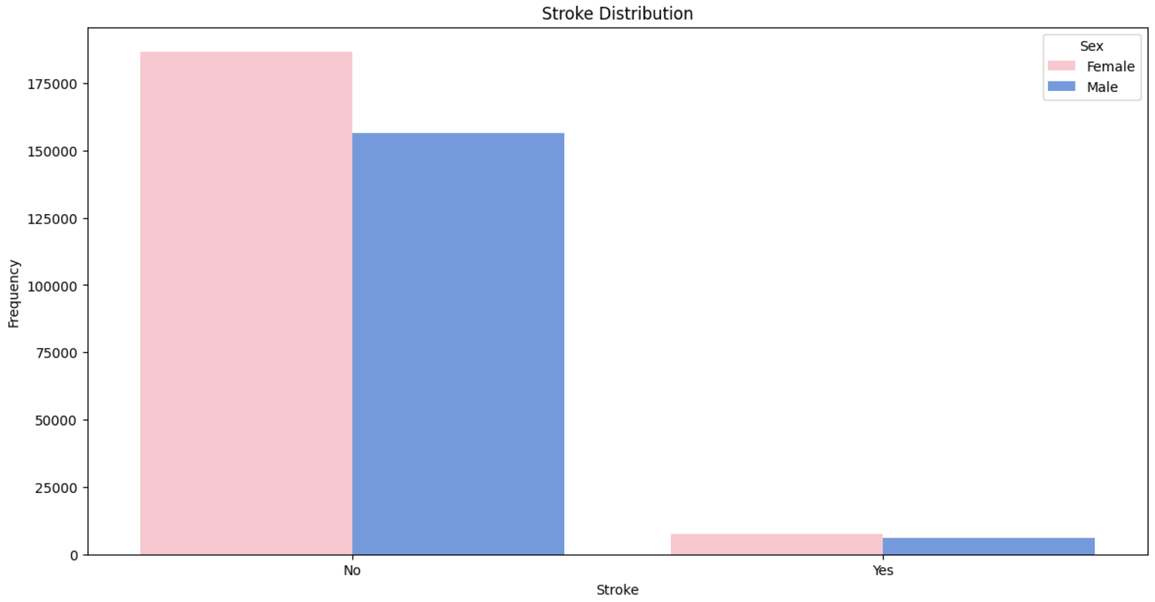


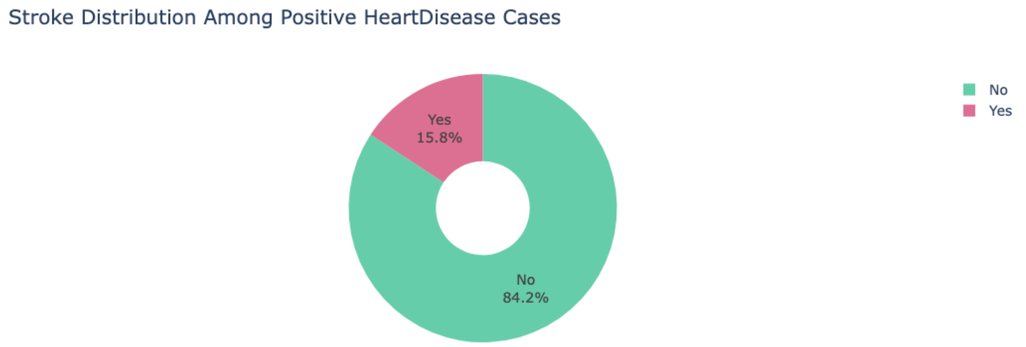


Stroke Distribution

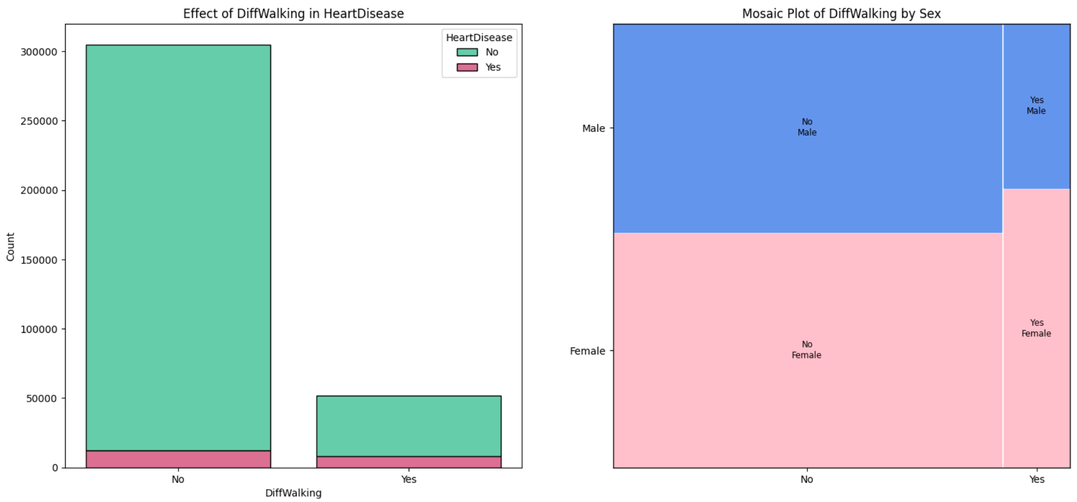


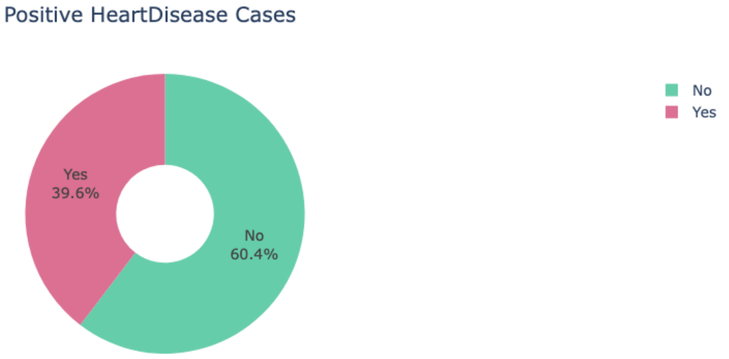




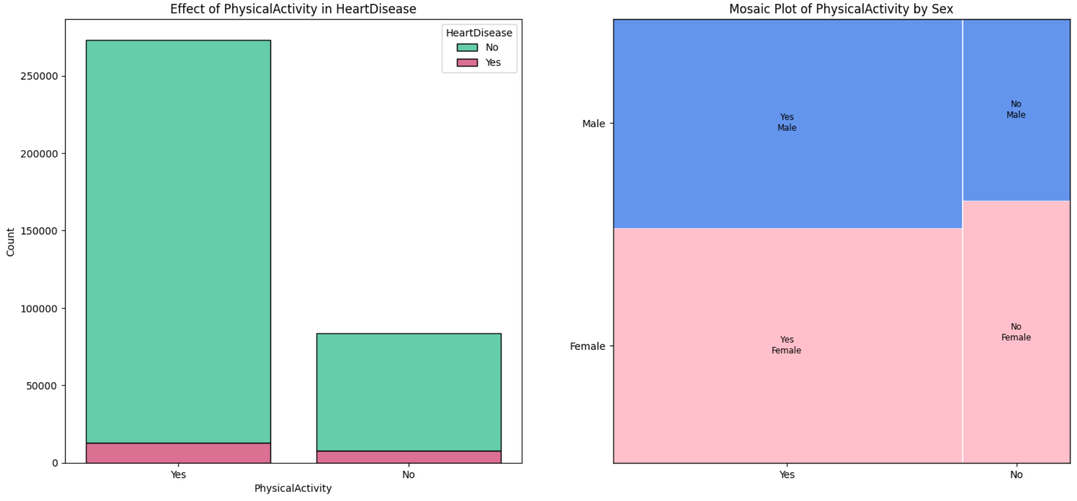


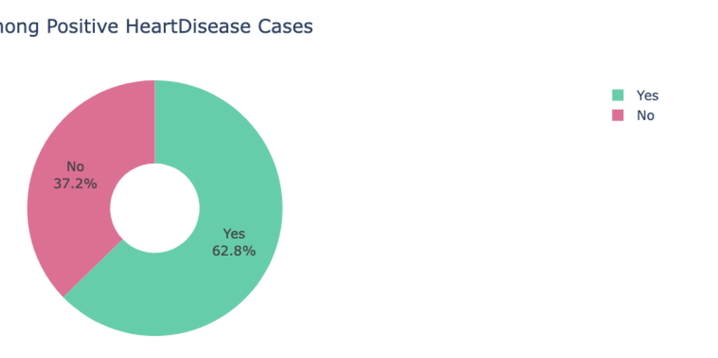
DiffWalking Distribution

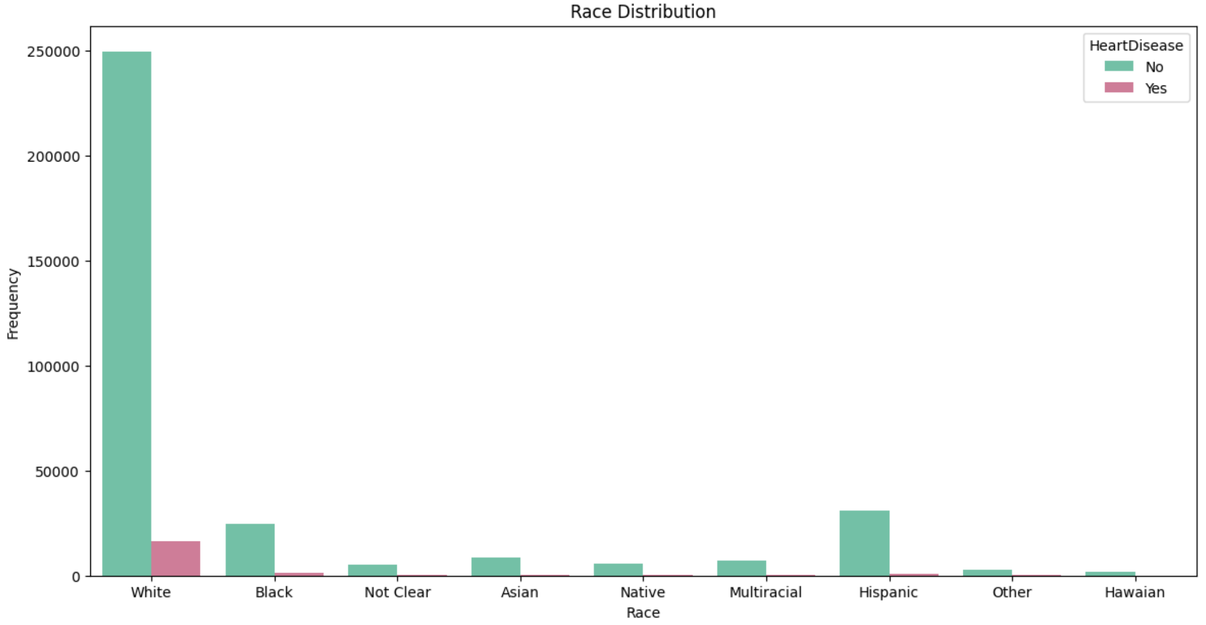


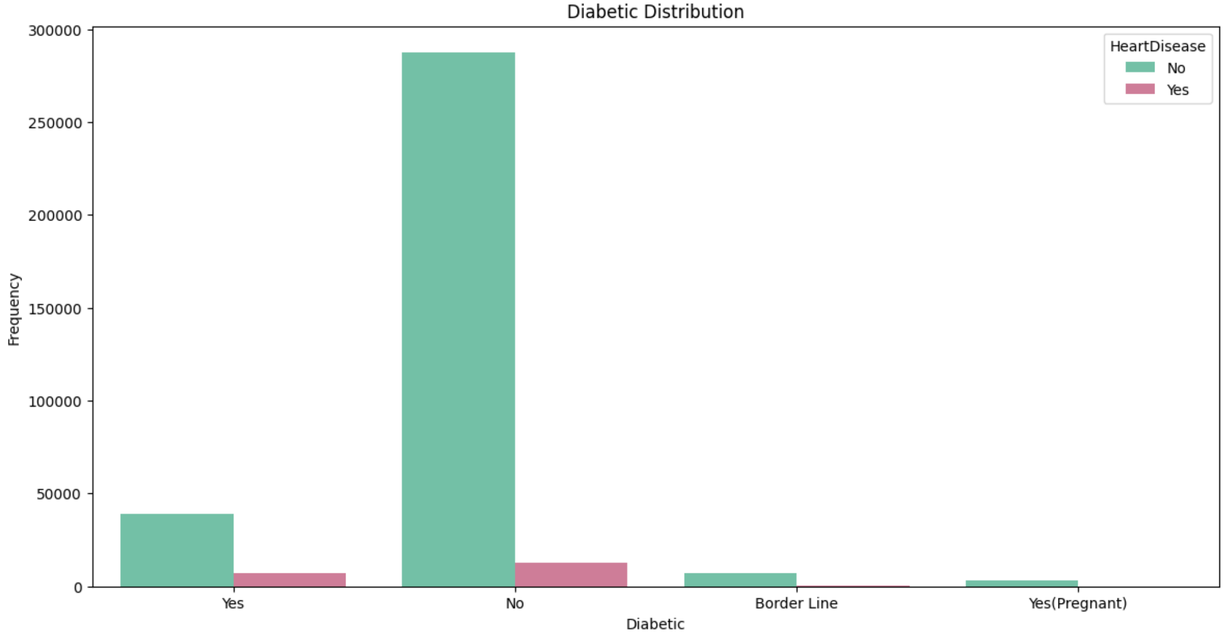


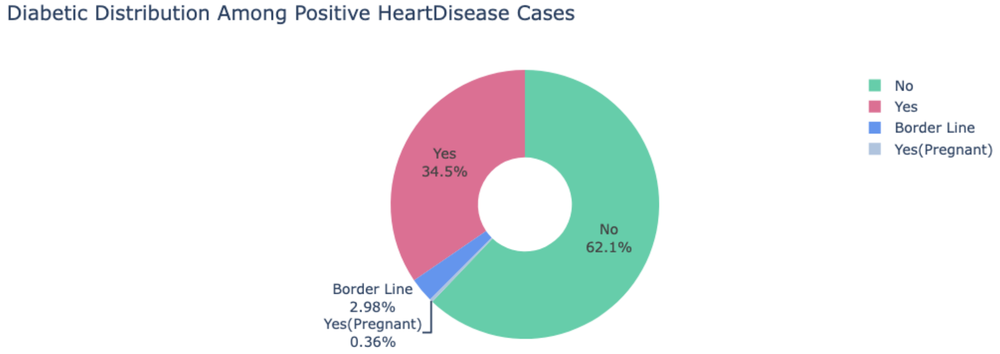
PhysicalActivity Distribution






Race Distribution
HeartDisease
No
Yes
Frequency
250000
200000
150000
100000
50000
0
White
Black
Not Clear
Asian
Native
Multiracial
Hispanic
Other
Hawaian
Race


Diabetic Distribution
HeartDisease
No
Yes
Frequency
300000
250000
200000
150000
100000
50000
0
Yes
No
Border Line
Yes(Pregnant)
Diabetic


Diabetic Distribution Among Positive HeartDisease Cases
Yes
34.5%
No
62.1%
Border Line
2.98%
Yes(Pregnant)
0.36%
No
Yes
Border Line
Yes(Pregnant)

Missing Data Imputation

Types of Missing data

Missing Completely at Random (MCAR):

We mean that the probability that an observation is missing is unrelated to the value of the variable or to the value of any other variables.
If the data are MCAR then missing values cannot be predicted any better with all information observed or not.

Missing at Random (MAR):

If data meet the requirement that missingness (i.e., the manner in which data are missing from a sample of a population) does not depend on the value of the variable after controlling for another variable.

The assumption of Missing at Random(MAR) is satisfied when the probability of a value's being missing in one variable is unrelated to the probability of missing data in another variable, but may be related to the value of the variable itself.

It can be considered as a weaker assumption than MCAR.
MAR would be satisfied if the probability of missing income was related to marital status, but unrelated to income within a marital status.

If MAR is satisfied,the mechanism causing the missing data may be considered to be "ignorable." That is, it doesn't matter why MAR occurred, only that it occurred.

Missing Not at Random (MNAR):

The reason for observations being missing still depends on the unseen observations themselves.

Methodology

The incremental tree-based methodology was introduced by Siciliano and Conversano, 2002 (see also Conversano and Siciliano, 2009).

Incremental means that each column presenting missing values, at each turn, plays the role of target variable to be imputed by the complete set of variables with no missing values playing the role of predictors variables. Once that this variable is imputed it concurs to form the complete set of predictors used for the subsequent imputation.

The incremental imputation of each variable at time(instead of each single data at time) allows a more efficient algorithm, thus reducing the computational cost of the overall incremental procedure.

Decision Tree Imputation

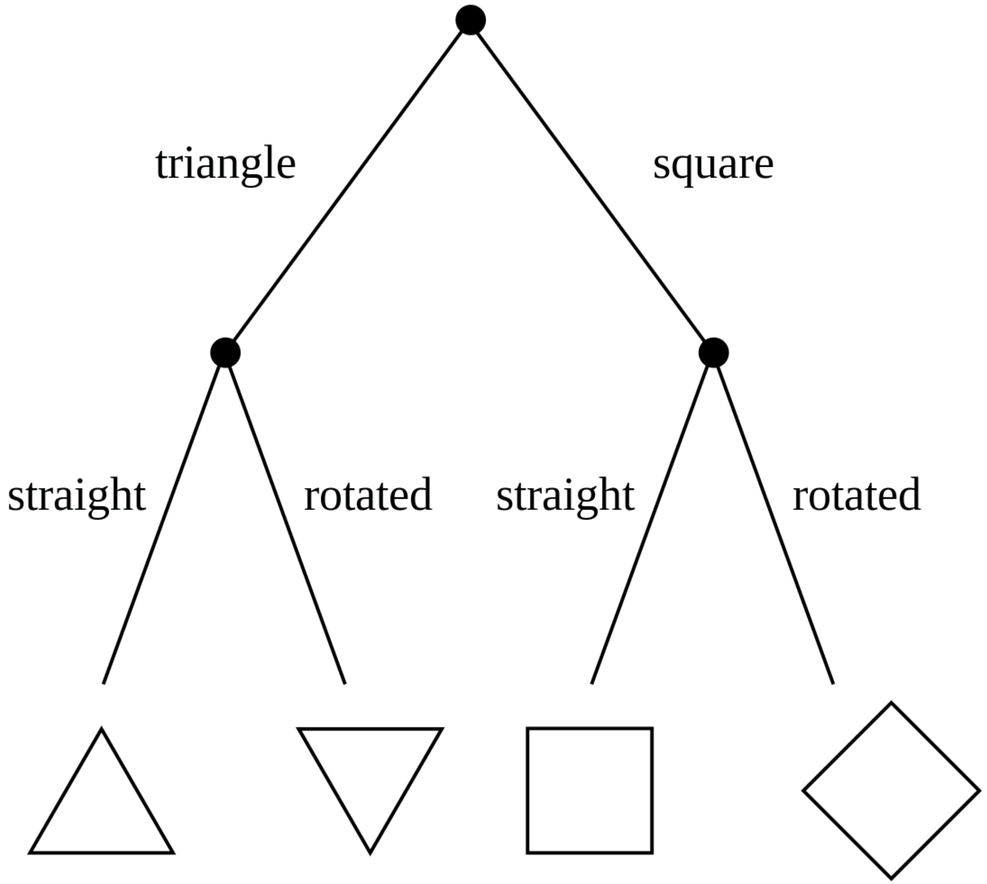
Imputing missing data is a crucial step in data preprocessing. It helps ensure that the dataset is complete and ready for machine learning modeling. Various imputation techniques are available, and the choice of technique depends on the nature of the data and the type of missing data.

One imputation method that we applied to the heart disease dataset is decision tree imputation. Decision tree imputation involves using a decision tree model to predict and fill in missing values based on other observed variables. This technique offers several advantages:

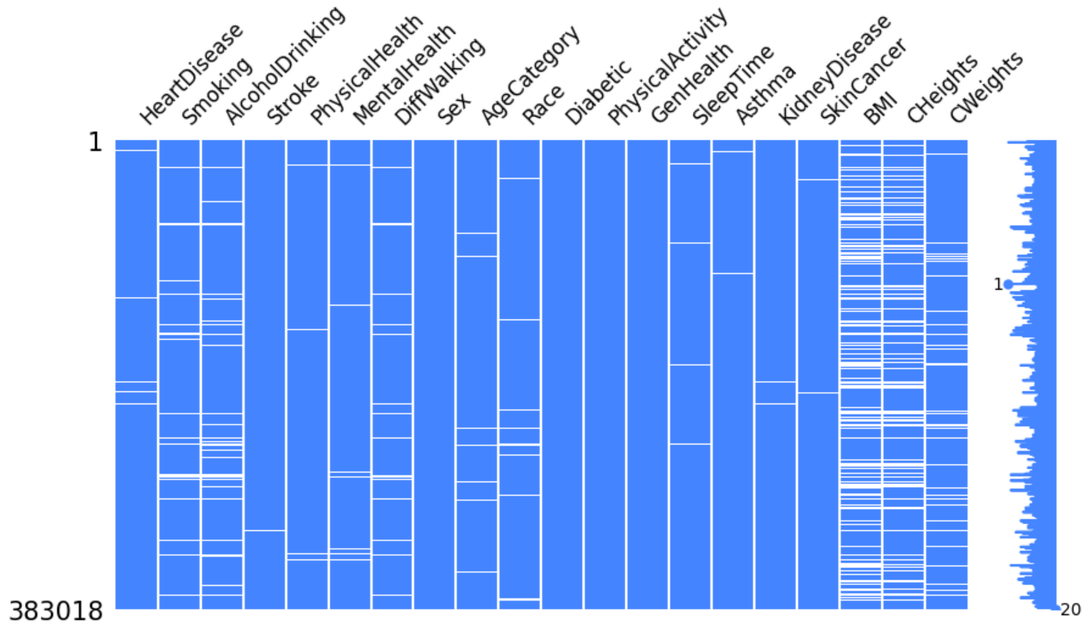
Non-linearity: Decision trees can capture complex, non-linear relationships in the data. They are well-suited for handling non-linear relationships between variables, which is common in real-world datasets.

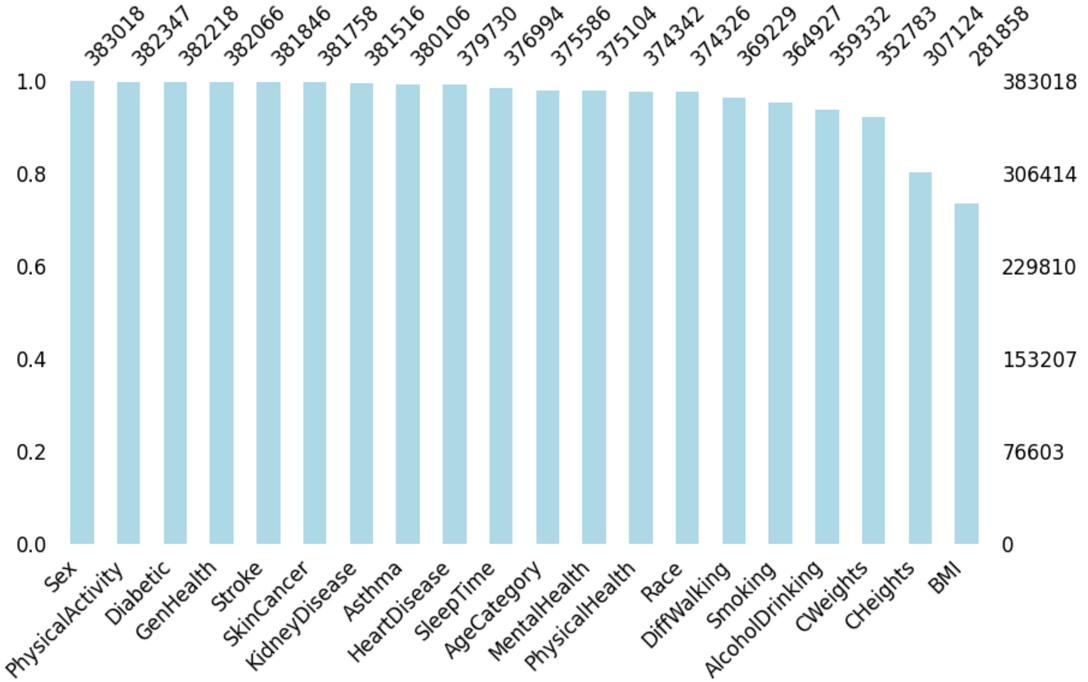
Handling Categorical Data: Decision trees can naturally handle both categorical and numerical data, making them suitable for datasets with a mix of variable types.

Variable Importance: Decision tree models can provide insights into the importance of each feature for imputing missing values. This information can be valuable for understanding which variables play a key role in imputation.
Robustness to Outliers: Decision trees are robust to outliers in the data, making them a suitable choice when missing data is related to outliers.



Missingness





Imputation Functions

```python
def numerical_imputer(df, column_name: str):

    # Split the data into two parts: known values and missing values for the specified numerical column
    var_known = df.dropna(subset=[column_name])
    var_missing = df[df[column_name].isna()]

    # Prepare features (X) and target (y) for the known values
    X_known = var_known.drop(columns=[column_name])
    y_known = pd.DataFrame(var_known[column_name])

    # Encode categorical features using LabelEncoder (assuming there are categorical features in X)
    X_known = X_known.apply(LabelEncoder().fit_transform)

    # Create a DecisionTreeRegressor and fit it to the known values
    Tree = DecisionTreeRegressor(random_state=0)
    Tree.fit(X_known, y_known)

    # Predict the missing numerical values using the fitted decision tree
    missed_values = var_missing.drop(columns=[column_name])
    missed_values[column_name] = Tree.predict(missed_values)

    # Round the predicted values to avoid decimals
    missed_values[column_name] = missed_values[column_name].round()  # This rounds to the nearest whole number

    # Print the count of NaN values before imputation
    print("Count of NaN's in ", column_name, " were: ", missed_values[column_name].shape[0])

    # Update the original DataFrame with the imputed values
    df.loc[var_missing.index, column_name] = missed_values[column_name]

    # Print a message indicating that the imputation is complete
    print(column_name, ' has been imputed')
    print('Imputation Score: ', Tree.score(X_known, y_known), '\n')
```

```python
def categorical_imputer(df, column_name: str):
    # Split the data into two parts: known values and missing values for the specified column
    var_known = df.dropna(subset=[column_name])
    var_missing = df[df[column_name].isna()]

    # Prepare features (X) and target (y) for the known values
    X_known = var_known.drop(columns=[column_name])
    y_known = pd.DataFrame(var_known[column_name])

    # Encode the categorical features using LabelEncoder
    X_known = X_known.apply(LabelEncoder().fit_transform)

    # Create a DecisionTreeClassifier and fit it to the known values
    Tree = DecisionTreeClassifier(random_state=0)
    Tree.fit(X_known, y_known)

    # Predict the missing values using the fitted decision tree
    missed_values = var_missing.drop(columns=[column_name])
    missed_values[column_name] = Tree.predict(missed_values)

    # Print the count of NaN values before imputation
    print("Count of NaN's in ", column_name, " were: ", missed_values[column_name].shape[0])

    # Update the original DataFrame with the imputed values
    df.loc[var_missing.index, column_name] = missed_values[column_name]

    # Print a message indicating that the imputation is complete
    print(column_name, ' has been imputed')
    print('Imputation Score: ', Tree.score(X_known, y_known), '\n')
```

Imputation Result

```
Count of NaN's in  HeartDisease  were:  3288
HeartDisease  has been imputed
Imputation Score:  0.9981399421551198

Count of NaN's in  AlcoholDrinking  were:  23310
AlcoholDrinking  has been imputed
Imputation Score:  0.9751434784558132

Count of NaN's in  Stroke  were:  1172
Stroke  has been imputed
Imputation Score:  0.9989181658425377

Count of NaN's in  Smoking  were:  17717
Smoking  has been imputed
Imputation Score:  0.9689273564524985

Count of NaN's in  PhysicalActivity  were:  671
PhysicalActivity  has been imputed
Imputation Score:  0.9899390492954963

Count of NaN's in  DiffWalking  were:  13417
DiffWalking  has been imputed
Imputation Score:  0.998211271714916

Count of NaN's in  SkinCancer  were:  1260
SkinCancer  has been imputed
Imputation Score:  0.9917893470010398
```

```
df.isna().any()
```

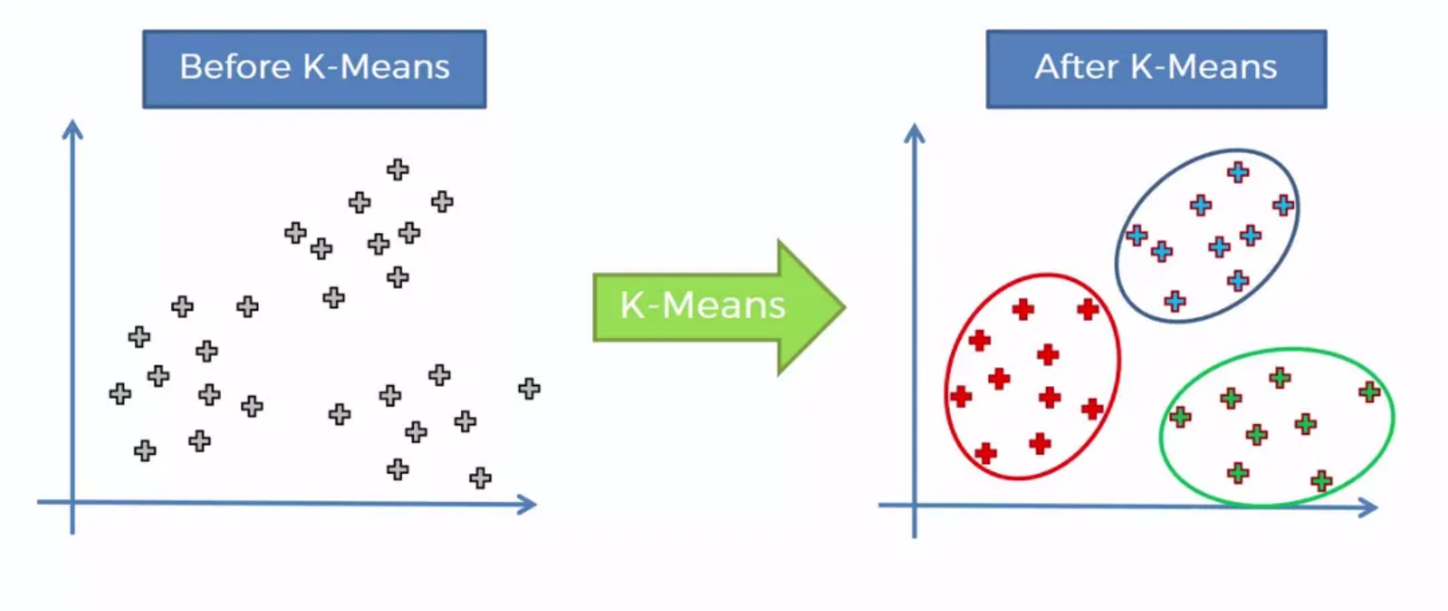
```
HeartDisease        False
Smoking             False
AlcoholDrinking     False
Stroke              False
PhysicalHealth      False
MentalHealth        False
DiffWalking         False
Sex                 False
AgeCategory         False
Race                False
Diabetic            False
PhysicalActivity    False
GenHealth           False
SleepTime           False
Asthma              False
KidneyDisease       False
SkinCancer          False
BMI                 False
```

```
HeartDisease : 2
Smoking : 4
AlcoholDrinking : 2
Stroke : 2
PhysicalHealth : 31
MentalHealth : 31
DiffWalking : 2
Sex : 2
AgeCategory : 13
Race : 8
Diabetic : 3
PhysicalActivity : 2
GenHealth : 5
SleepTime : 12
Asthma : 3
KidneyDisease : 2
SkinCancer : 2
BMI : 1946
```

Cluster Analysis

Methodology

Lets do the cluster analysis for those who have heart disease



K-means is an iterative algorithm. It starts with a predefined number of clusters, 'K', and assigns each data point to the nearest cluster centroid. Then, it recalculates the cluster centroids as the mean of the data points assigned to each cluster. This process is repeated until convergence, where the assignment of data points to clusters remains unchanged.

Elbow Method

1 - Purpose of the Elbow Method:
The elbow method is used to determine the optimal number of clusters in unsupervised machine learning algorithms, such as K-Means.
It helps strike a balance between overfitting (using too many clusters) and underfitting (using too few clusters).

2 - How It Works:
The method involves running the clustering algorithm for a range of K values (e.g., from 1 to a maximum K).
For each K, the **sum of squared distances (inertia)** of data points to their assigned cluster centers is calculated. In K-Means, this is called the "within-cluster sum of squares."
The plot of K values against the corresponding inertia is created.

3 - Elbow Point:
When you plot the inertia as a function of K, the graph often resembles an "elbow."
The point at which the inertia starts to decrease more slowly and forms an "elbow" in the plot is the optimal K value.

4 - Choosing K:
Selecting K is somewhat subjective, but a common approach is to look for the point where the rate of decrease in inertia sharply changes, or where the curve begins to flatten out.
The optimal K is typically the point just before the inertia starts to level off.

Creating new variables for Cluster Analysis

Assuming that 0 in 'AlcoholDrinking' means not drinking and 1 means drinking

And that 1 in 'PhysicalActivity' means active and 0 means inactive

Not drinking alcohol: +1 point

Being physically active: +1 point

7-9 hours of sleep: +1 point, scaling down to 0 as you move away from this range

Sleep Score:

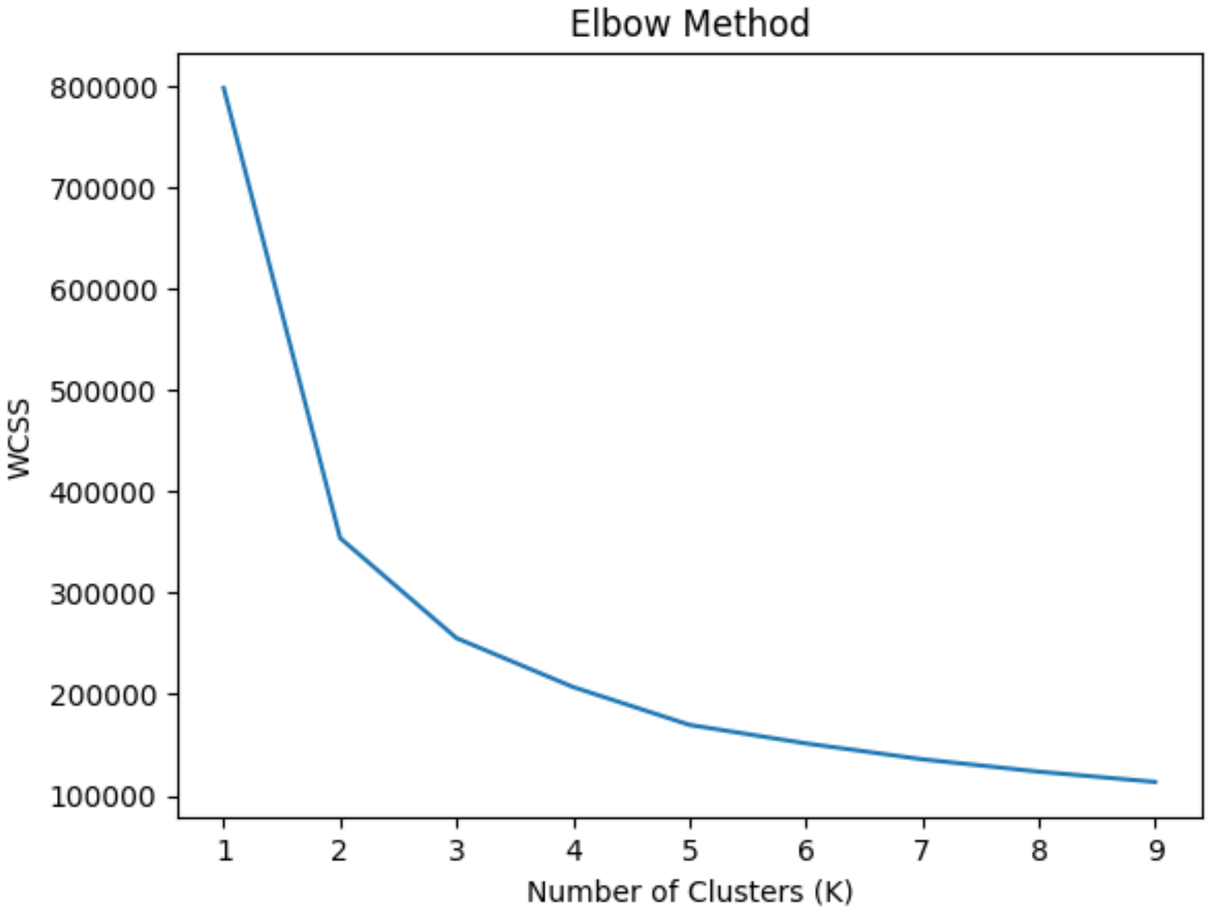
Create a normalized sleep score that peaks at 7-9 hours and decreases towards 0 or 24 hours

The simplest way could be to use a Gaussian-like function or triangular function

Smoking Scores: 1: -1, # Smoker 2: 0, # Approximate Smoker 3: 0.5, # Former Smoker 4: 1 # Never

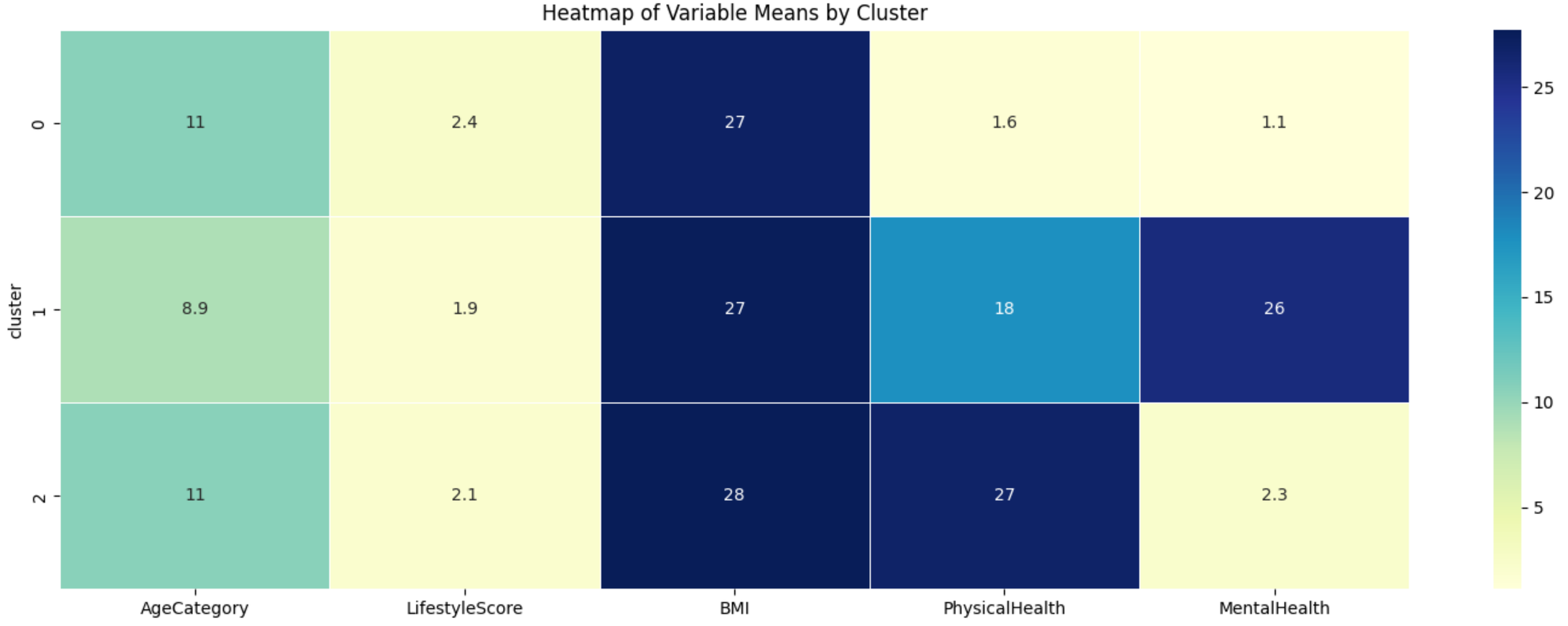
LifestyleScore = 'AlcoholScore', 'PhysicalActivityScore', 'SleepScore', 'SmokingScore'

Elbow method result



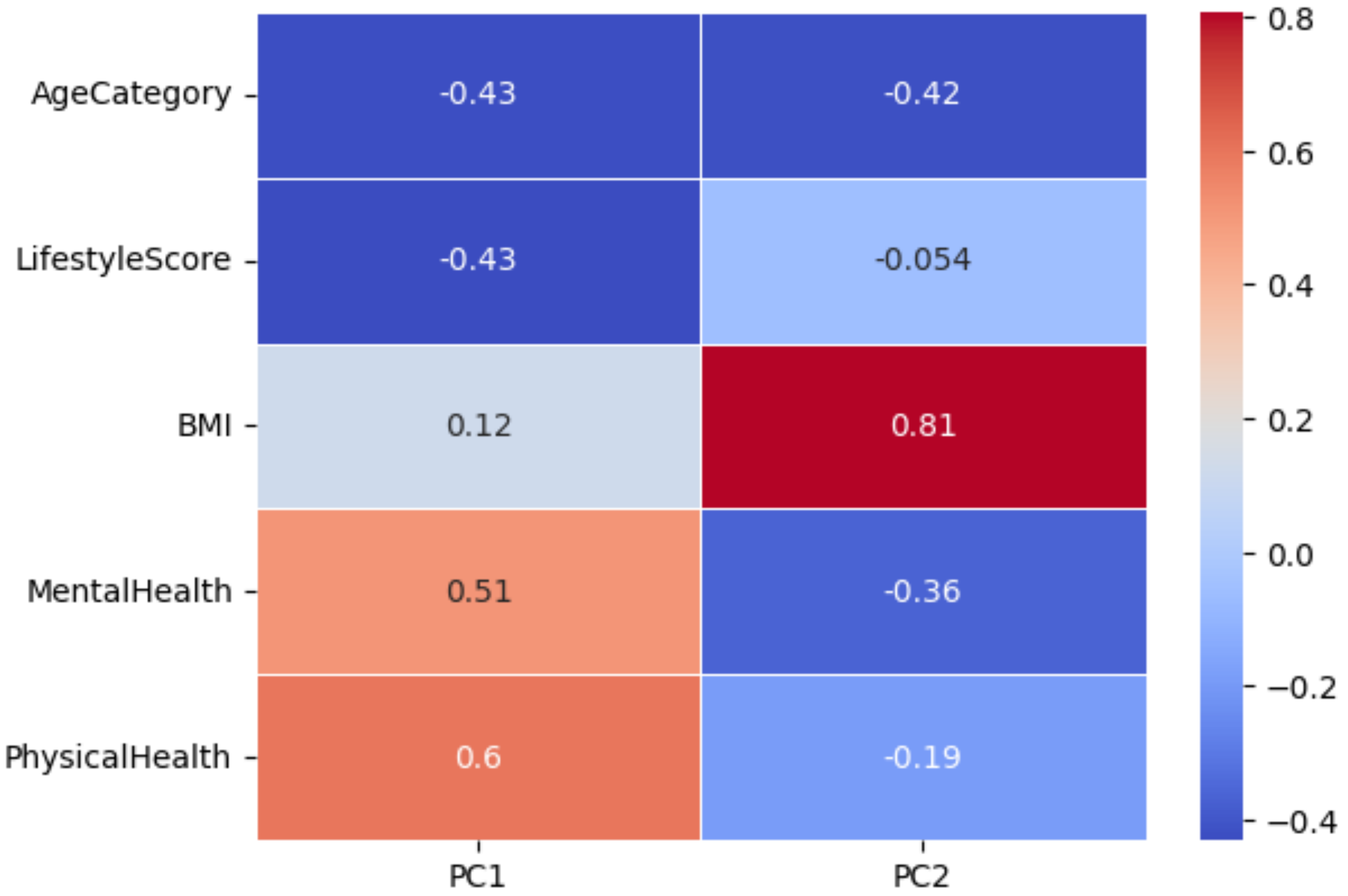
Mean of features in each cluster

cluster	AgeCategory	LifestyleScore	BMI	PhysicalHealth	MentalHealth
0	10.606436	2.375207	27.003950	1.622395	1.121670
1	8.912827	1.887139	27.487772	17.828588	25.685941
2	10.629417	2.144333	27.686776	26.756625	2.257509


Heatmap of Variable Means by Cluster
cluster
0
1
2
11
2.4
27
1.6
1.1
8.9
1.9
27
18
26
11
2.1
28
27
2.3
AgeCategory
LifestyleScore
BMI
PhysicalHealth
MentalHealth
25
20
15
10
5

Principal Component Analysis

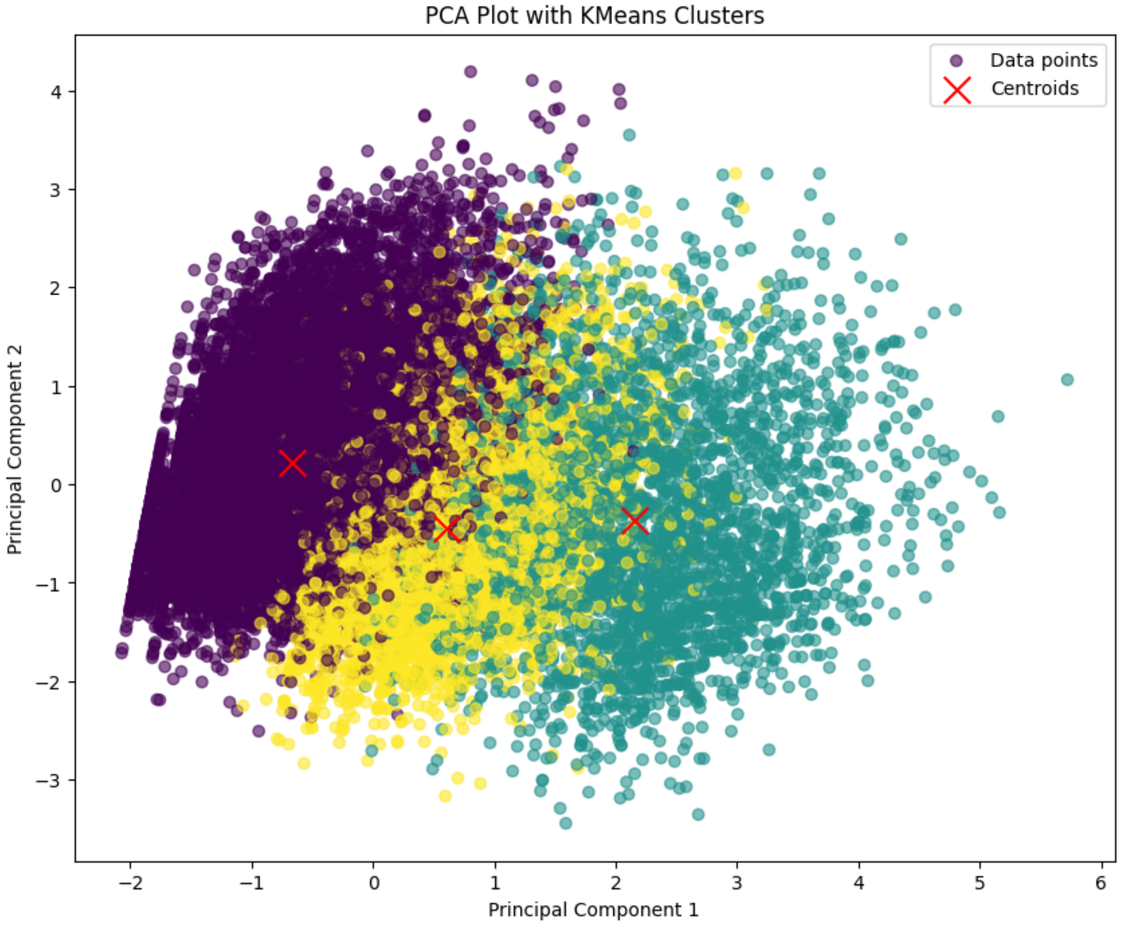
PCA Loadings

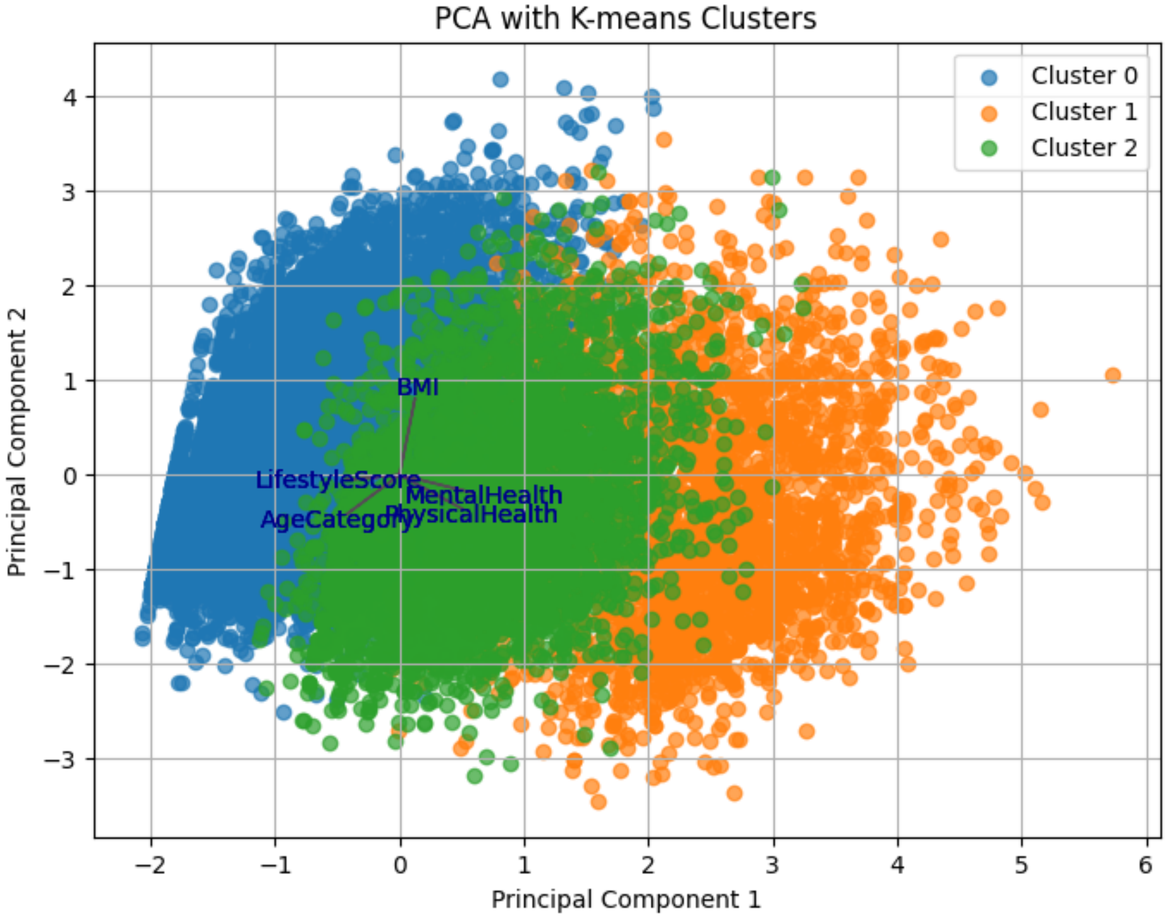


Overal Insights

The first principal component seems to contrast age and lifestyle factors with BMI, potentially capturing an aspect of health risk or status that increases with higher BMI and lower lifestyle scores

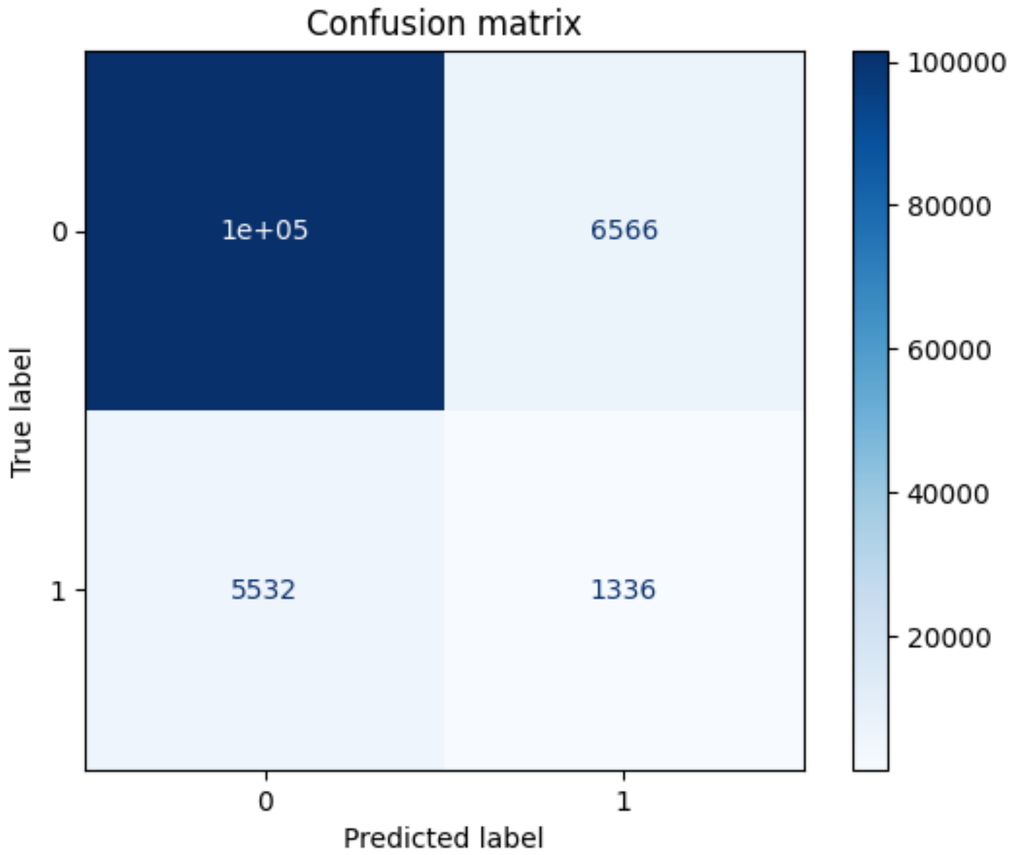
The second principal component appears to be a contrast between physical and mental health, with a particular emphasis on the ratio of poor physical to mental health days.


PCA Plot with KMeans Clusters
Data points
Centroids
Principal Component 2
Principal Component 1


PCA with K-means Clusters
Cluster 0
Cluster 1
Cluster 2
BMI
LifestyleScore
MentalHealth
PhysicalHealth
AgeCategory
Principal Component 2
Principal Component 1

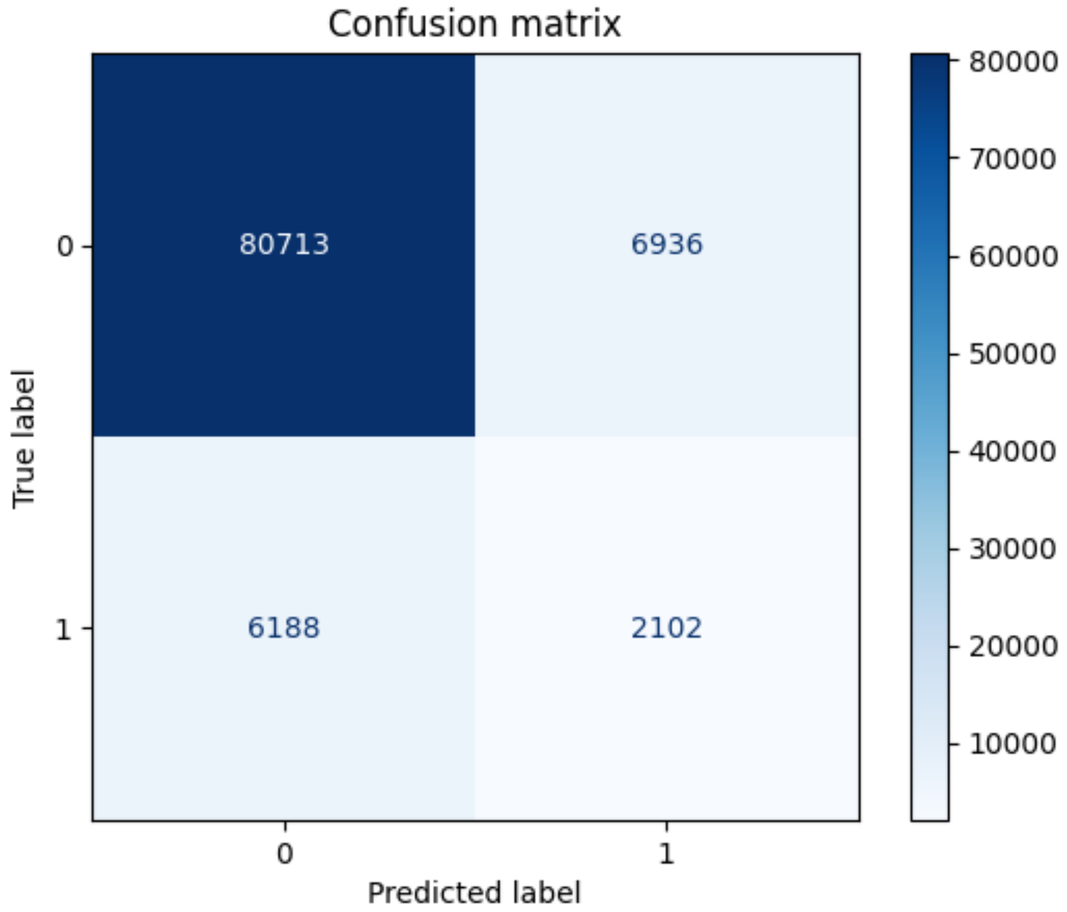
Classification and Regression

Decision Tree for Imputed Dataset




```
Accuracy(Test Set Score):  0.8947139400901607

Classification Score:  0.9665237665070571

Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.94      0.94    108038
           1       0.17      0.19      0.18      6868

    accuracy                           0.89    114906
   macro avg       0.56      0.57      0.56    114906
weighted avg       0.90      0.89      0.90    114906
```

Decision Tree for Kaggle Dataset




```
Accuracy(Test Set Score):  0.8632047446815164

Classification Score:  0.9569568004502884

Classification Report:
              precision    recall  f1-score   support

           0       0.93      0.92      0.92     87649
           1       0.23      0.25      0.24      8290

    accuracy                           0.86     95939
   macro avg       0.58      0.59      0.58     95939
weighted avg       0.87      0.86      0.87     95939
```

OverSampling

Oversampling is a technique used in the field of machine learning and data analysis to address class imbalance in a dataset. Class imbalance occurs when one or more classes or categories in a classification problem have significantly fewer instances than others. In the context of oversampling, it typically refers to the practice of increasing the number of instances in the minority class(es) to balance the class distribution.

Here's how oversampling works:

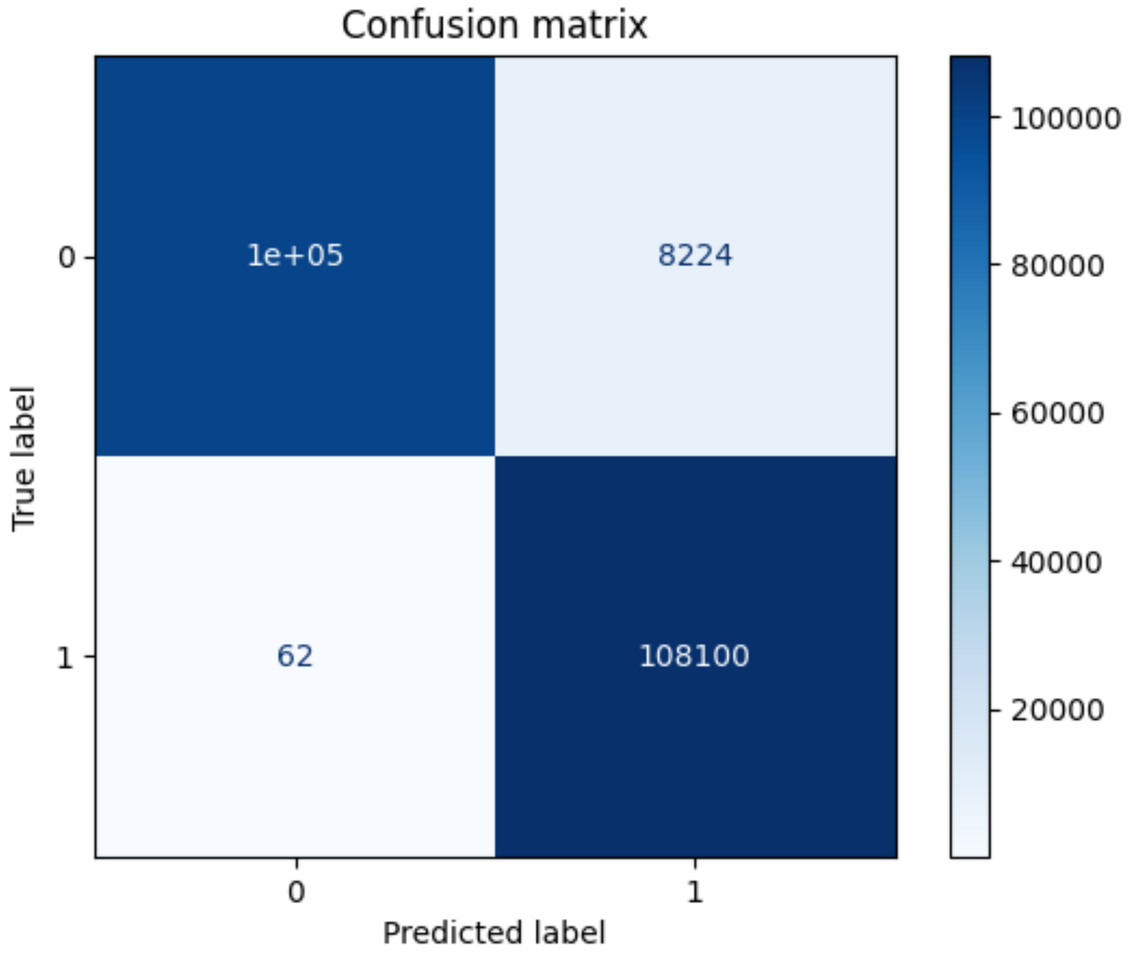
Class Imbalance Problem:

In many real-world datasets, especially in fields like fraud detection, medical diagnosis, and rare event prediction, one class may be underrepresented, making it difficult for a machine learning model to learn from the minority class.

Random Oversampling:

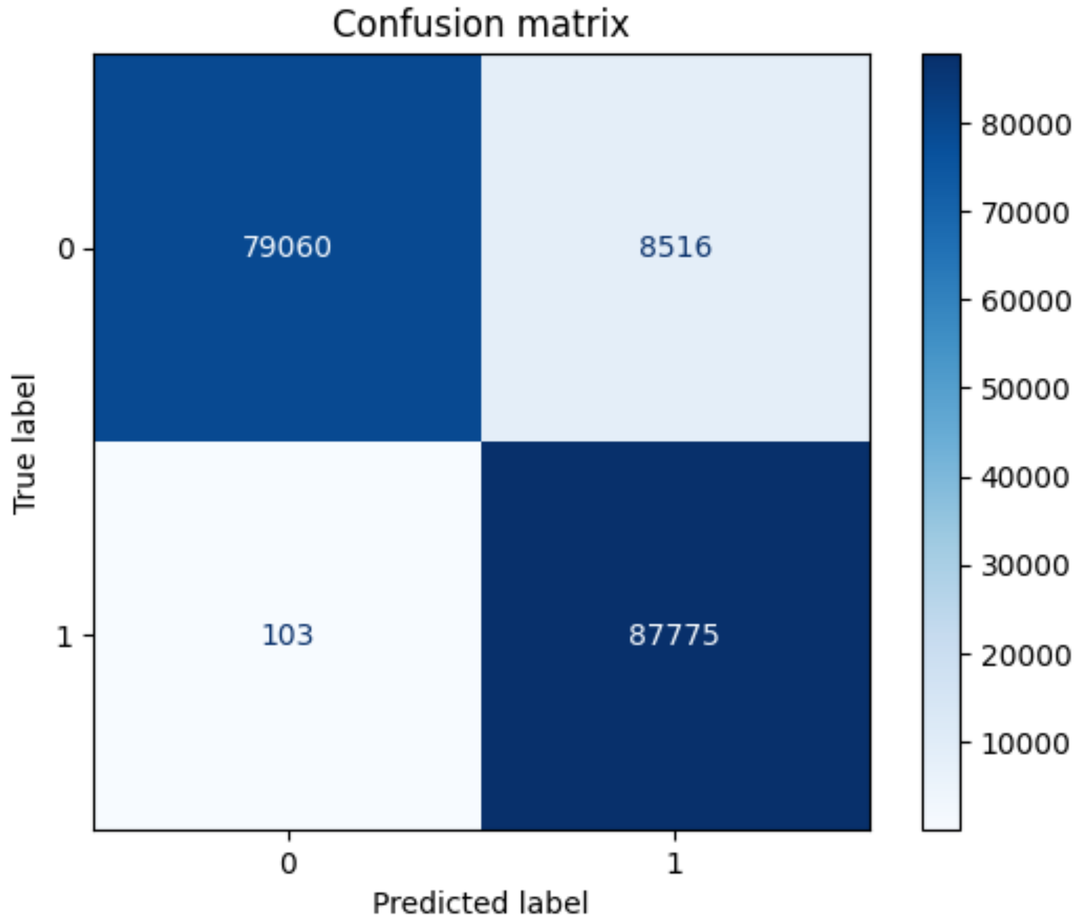
One simple oversampling technique is random oversampling, where new instances are randomly sampled (with replacement) from the existing instances of the minority class. This effectively increases the number of minority class instances.

Decision Tree for OverSampled Imputed Dataset




```
Test Set Score after Sampling:  96.16%
Train Set Score after Sampling:  99.61%
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.92      0.96    107790
           1       0.93      1.00      0.96    108162

    accuracy                           0.96    215952
   macro avg       0.96      0.96      0.96    215952
weighted avg       0.96      0.96      0.96    215952
```

Decision Tree for OverSampled Kaggle Dataset

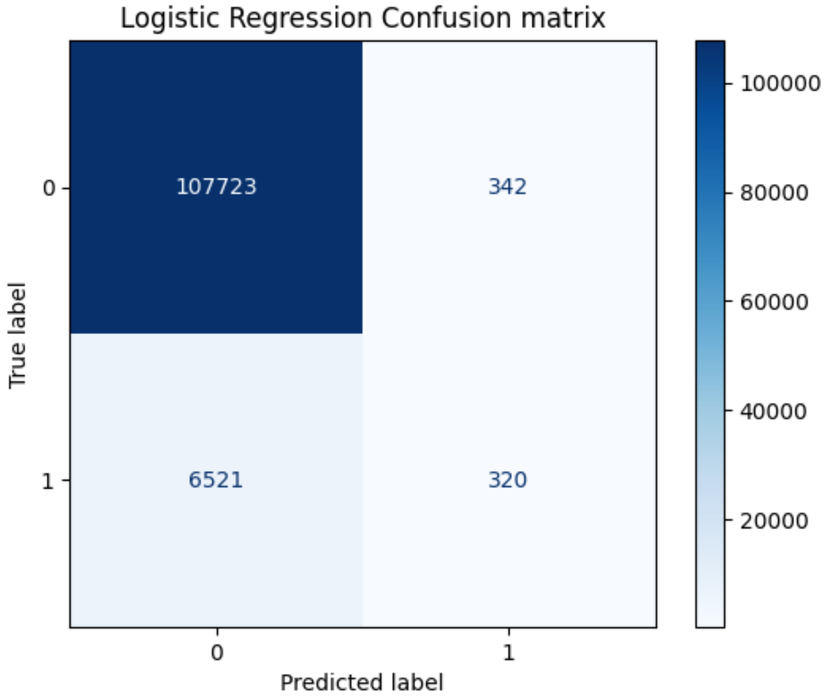



```
Test Set Score after Sampling:  95.09%
Train Set Score after Sampling:  99.66%
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.90      0.95     87576
           1       0.91      1.00      0.95     87878

    accuracy                           0.95    175454
   macro avg       0.96      0.95      0.95    175454
weighted avg       0.96      0.95      0.95    175454
```

Logistic Regression

Logistic Regression for Imputed Dataset

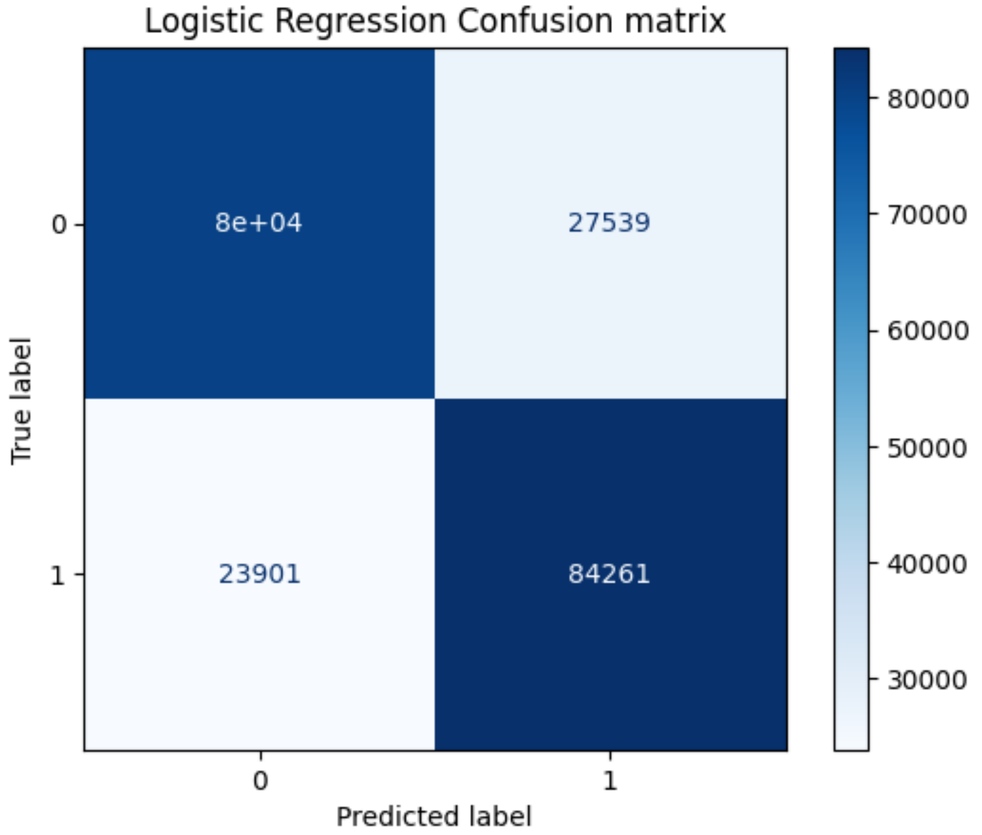
	Feature	Coefficient
0	Smoking	0.123332
1	AlcoholDrinking	-0.044368
2	Stroke	0.157294
3	PhysicalHealth	0.037959
4	MentalHealth	0.021044
5	DiffWalking	0.086105
6	Sex	-0.329169
7	AgeCategory	0.871916
8	Race	-0.092439
9	Diabetic	0.160186
10	PhysicalActivity	0.038029
11	GenHealth	0.511005
12	SleepTime	-0.040945
13	Asthma	0.090967
14	KidneyDisease	0.119193
15	SkinCancer	0.060615
16	BMI	0.069471




```
Train Set Shape:  (268112, 17) (268112,)
Test Set Shape:  (114906, 17) (114906,)

Train set Accuracy:  93.91%
Test set Accuracy:  94.03%
Logistic Regression's f1 Score:  91.65%
```

Logistic Regression for OverSampled Imputed Dataset

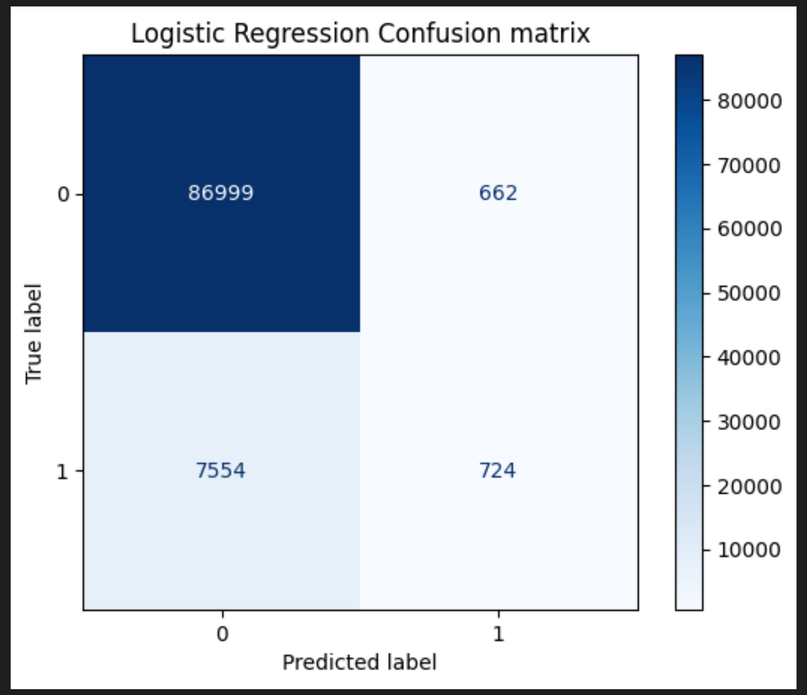
	Feature	Coefficient
0	Smoking	0.135064
1	AlcoholDrinking	-0.044017
2	Stroke	0.293695
3	PhysicalHealth	0.057240
4	MentalHealth	0.048383
5	DiffWalking	0.115925
6	Sex	-0.374192
7	AgeCategory	0.973128
8	Race	-0.056215
9	Diabetic	0.187934
10	PhysicalActivity	0.047318
11	GenHealth	0.629734
12	SleepTime	-0.051918
13	Asthma	0.118958
14	KidneyDisease	0.188328
15	SkinCancer	0.066587
16	BMI	0.067204




```
Train set Accuracy:  76.12%
Test set Accuracy:  76.18%
Logistic Regression's f1 Score:  76.17%
```

Logistic Regression for Kaggle Dataset

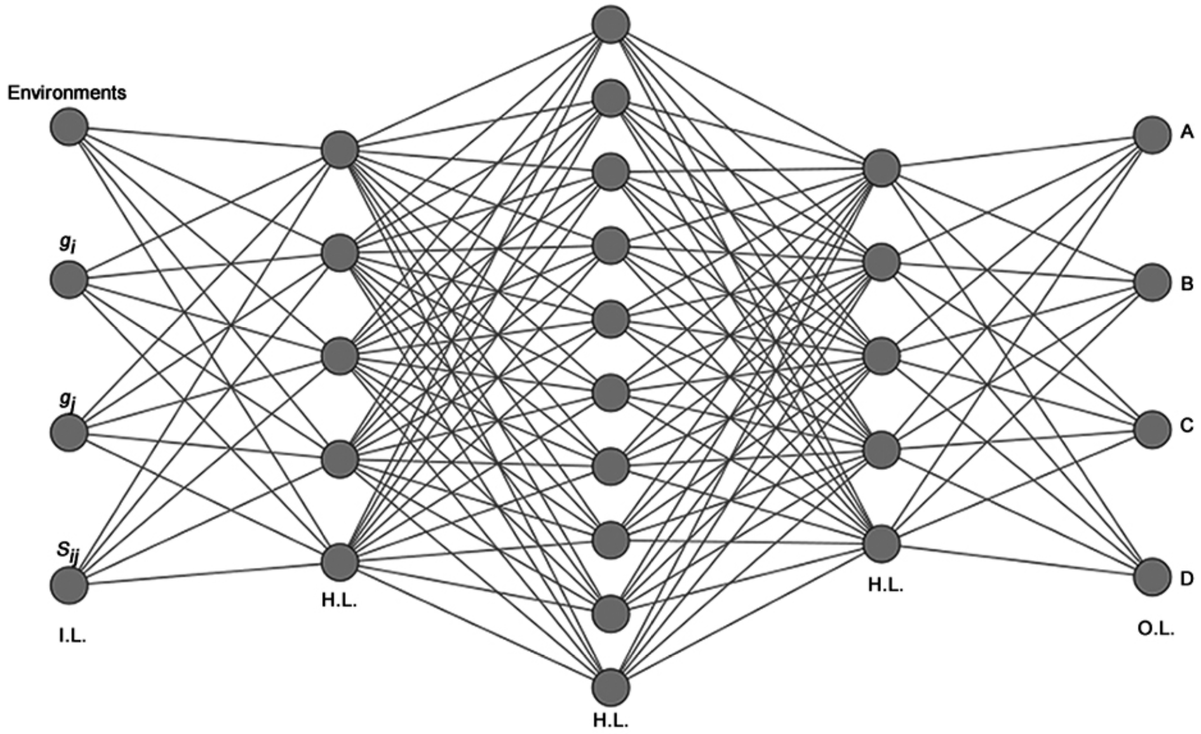
	Feature	Coefficient
0	BMI	0.084959
1	Smoking	0.222917
2	AlcoholDrinking	-0.060926
3	Stroke	0.213707
4	PhysicalHealth	0.170281
5	MentalHealth	0.077589
6	DiffWalking	0.137547
7	Sex	0.348633
8	AgeCategory	0.962168
9	Race	0.038585
10	Diabetic	0.214122
11	PhysicalActivity	-0.038381
12	GenHealth	-0.045929
13	SleepTime	-0.044607
14	Asthma	0.122195
15	KidneyDisease	0.132359
16	SkinCancer	0.041035




```
Train Set Shape:  (223856, 17) (223856,)
Test Set Shape:  (95939, 17) (95939,)

Train set Accuracy:  91.55%
Test set Accuracy:  91.44%
Logistic Regression's f1 Score:  88.54%
```

Multi Layer Perceptron

Multi layer perceptron (MLP) is a supplement of feed forward neural network. It consists of three types of layers the input layer, output layer and hidden layer. The input layer receives the input signal to be processed. The required task such as prediction and classification is performed by the output layer. An arbitrary number of hidden layers that are placed in between the input and output layer are the true computational engine of the MLP. Similar to a feed forward network in a MLP the data flows in the forward direction from input to output layer. The neurons in the MLP are trained with the back propagation learning algorithm. MLPs are designed to approximate any continuous function and can solve problems which are not linearly separable. The major use cases of MLP are pattern classification, recognition, prediction and approximation.



TensorFlow Feature Columns

```
# Define feature columns
feature_columns = []

# Numeric feature columns
for feature_name in num_var:
    feature_columns.append(tf.feature_column.numeric_column(feature_name, dtype=tf.float32))

# Categorical feature columns (use one-hot encoding)
for feature_name in cat_var:
    vocabulary_list = df[feature_name].unique()
    cat_column = tf.feature_column.categorical_column_with_vocabulary_list(feature_name, vocabulary_list)
    one_hot_column = tf.feature_column.indicator_column(cat_column)
    feature_columns.append(one_hot_column)

# Binary feature columns (use one-hot encoding)
for feature_name in bin_var:
    vocabulary_list = df[feature_name].unique()
    bin_column = tf.feature_column.categorical_column_with_vocabulary_list(feature_name, vocabulary_list)
    one_hot_column = tf.feature_column.indicator_column(bin_column)
    feature_columns.append(one_hot_column)
```

Model Creation

```
# Build and compile the model with the custom initializer for a specific layer
model = tf.keras.Sequential([
        tf.keras.layers.DenseFeatures(feature_columns),  # Input layer with feature columns
        tf.keras.layers.Dense(units=128, activation='relu', kernel_initializer=initializer),  # Hidden layer with custom initializer
        tf.keras.layers.Dropout(rate=0.2),  # Dropout layer for regularization
        tf.keras.layers.Dense(units=128, activation='relu'),  # Another hidden layer
        tf.keras.layers.Dense(units=1, activation='sigmoid')  # Output layer with sigmoid activation for binary classification
    ])

# Add the AUC metric to the list of metrics
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy', tf.keras.metrics.AUC()])
```

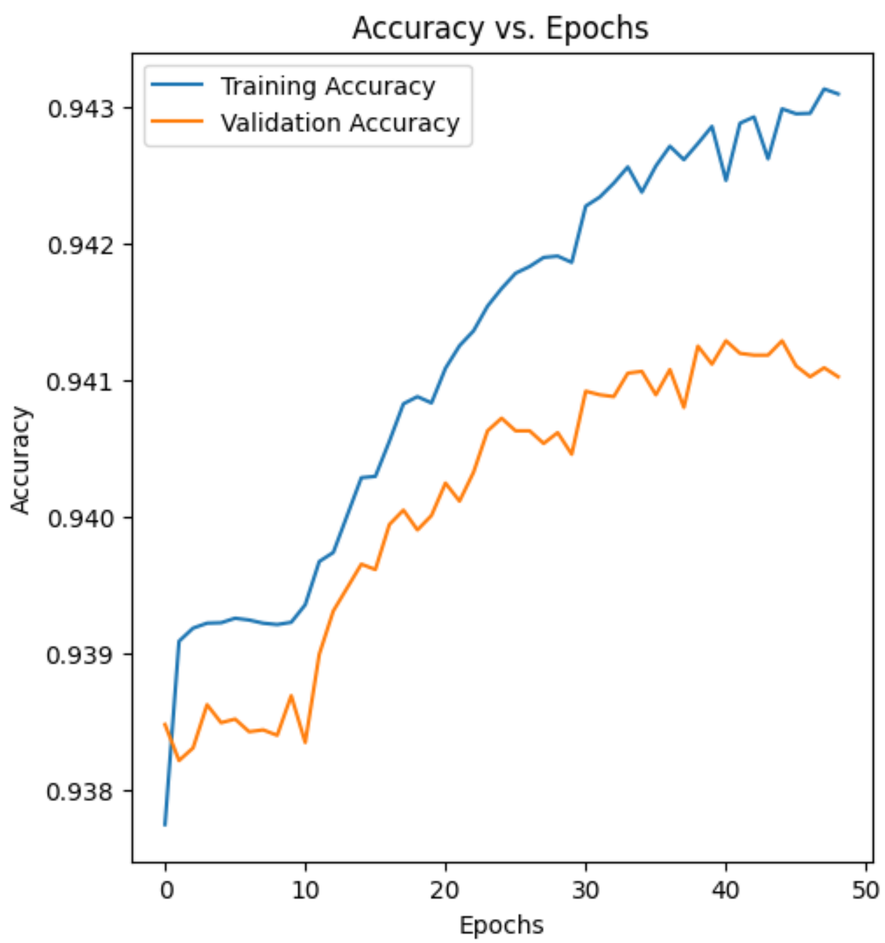
```
# Implement early stopping
early_stopping = tf.keras.callbacks.EarlyStopping(
    monitor='val_loss',
    patience=10,  # Number of epochs with no improvement after which training will be stopped.
    restore_best_weights=True  # Restore model weights from the epoch with the best value of the monitored quantity.
)
```

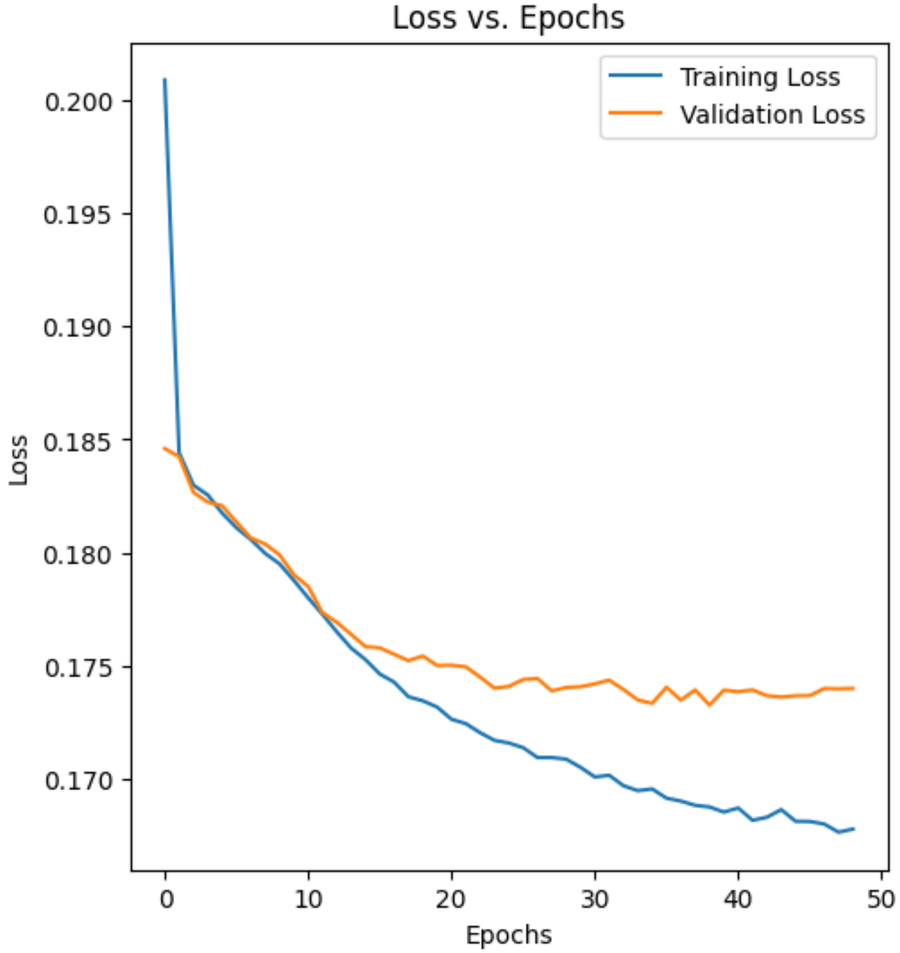
```
# Train the model
history = model.fit(train_ds, validation_data=eval_ds, epochs=100, callbacks=[early_stopping])
```

MLP for Imputed Dataset

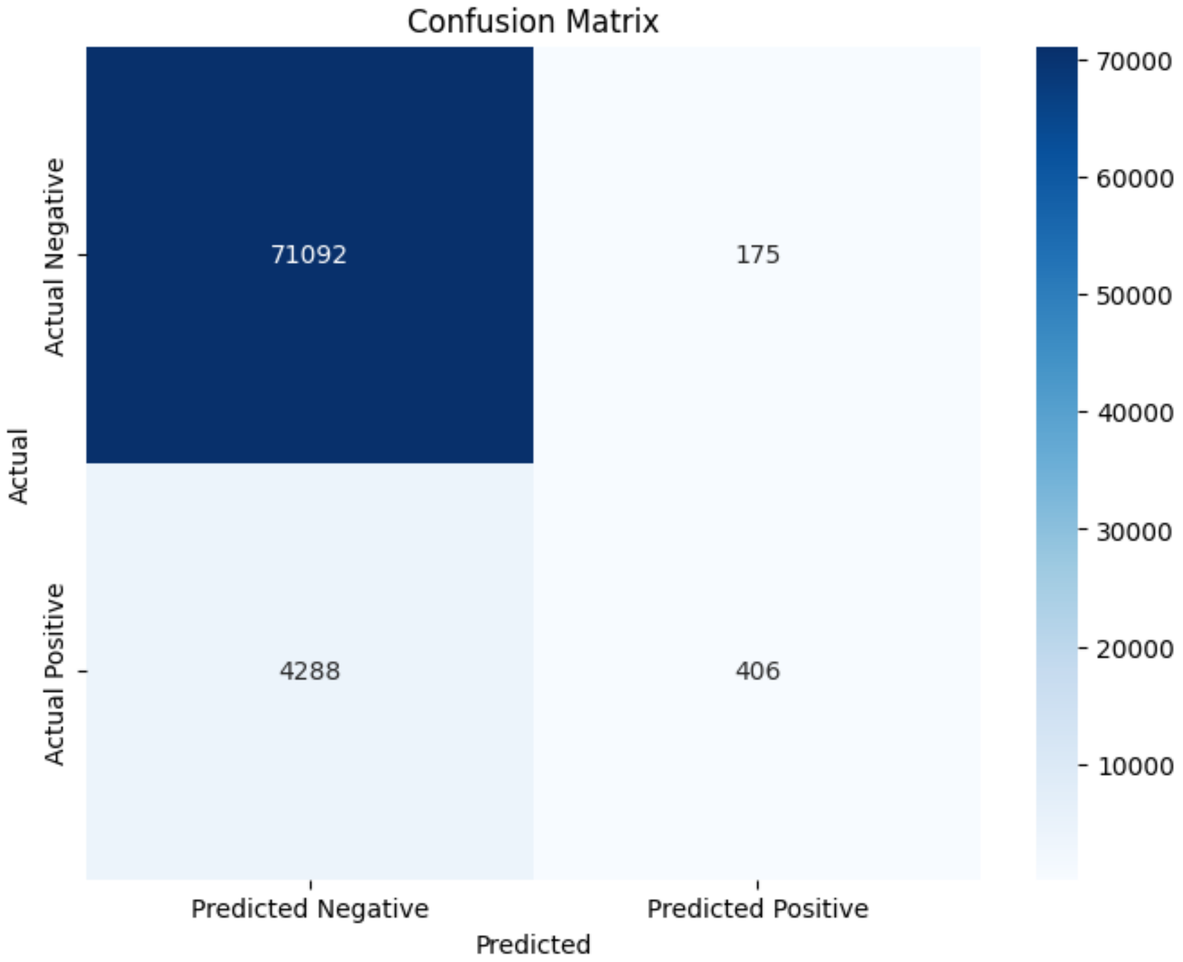
```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_features_1 (DenseFea  multiple                  0
 tures)

 dense_3 (Dense)             multiple                  7040

 dropout_1 (Dropout)         multiple                  0

 dense_4 (Dense)             multiple                  16512

 dense_5 (Dense)             multiple                  129

=================================================================
Total params: 23681 (92.50 KB)
Trainable params: 23681 (92.50 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
None
```

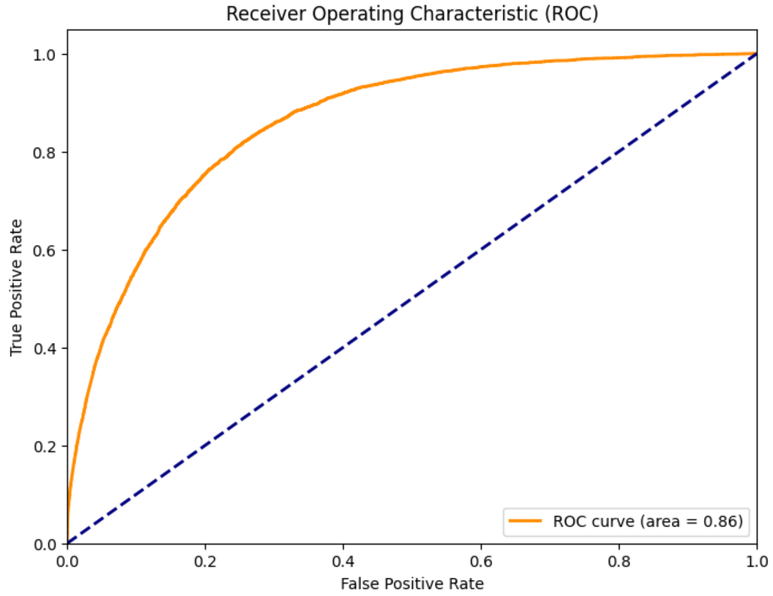
MLP for Imputed Dataset





MLP for Imputed Dataset





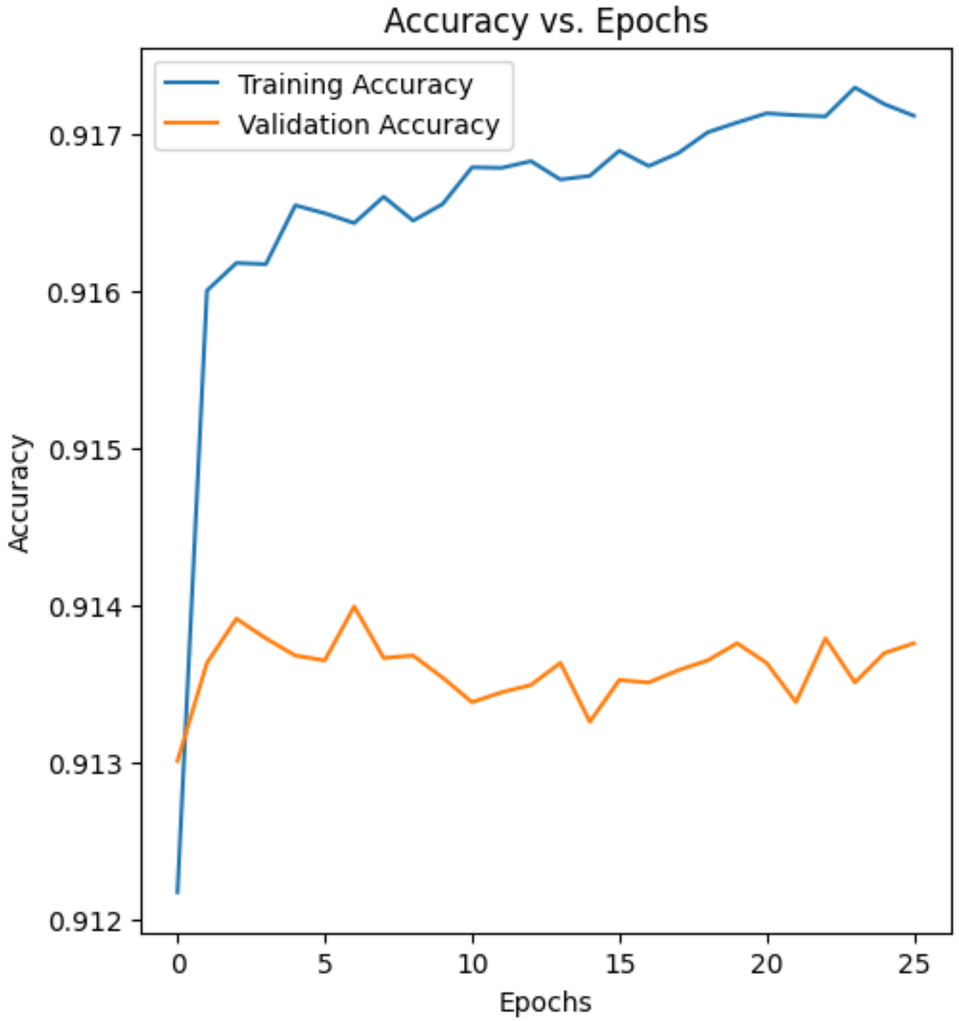
	precision	recall	f1-score	support
0	0.94	1.00	0.97	71267
1	0.70	0.09	0.15	4694
accuracy			0.94	75961
macro avg	0.82	0.54	0.56	75961
weighted avg	0.93	0.94	0.92	75961

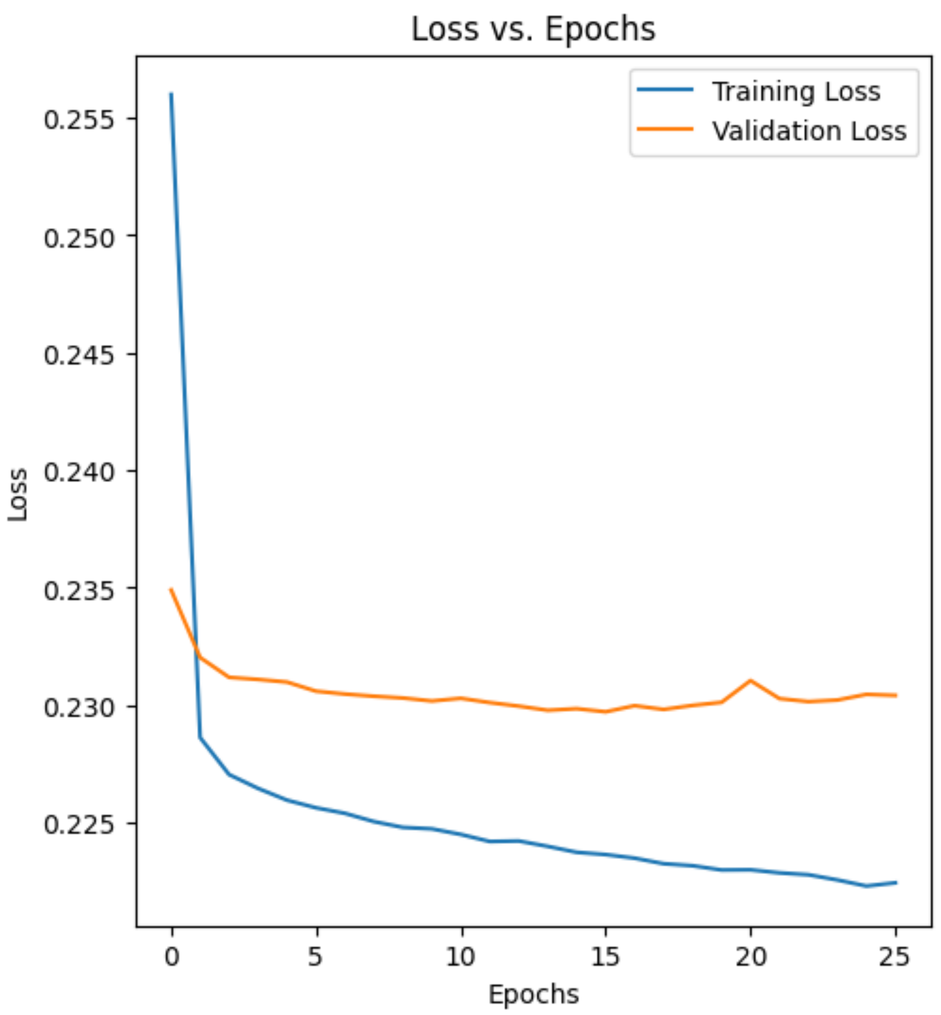
AUC Score: 86.05%

MLP for Kaggle Dataset

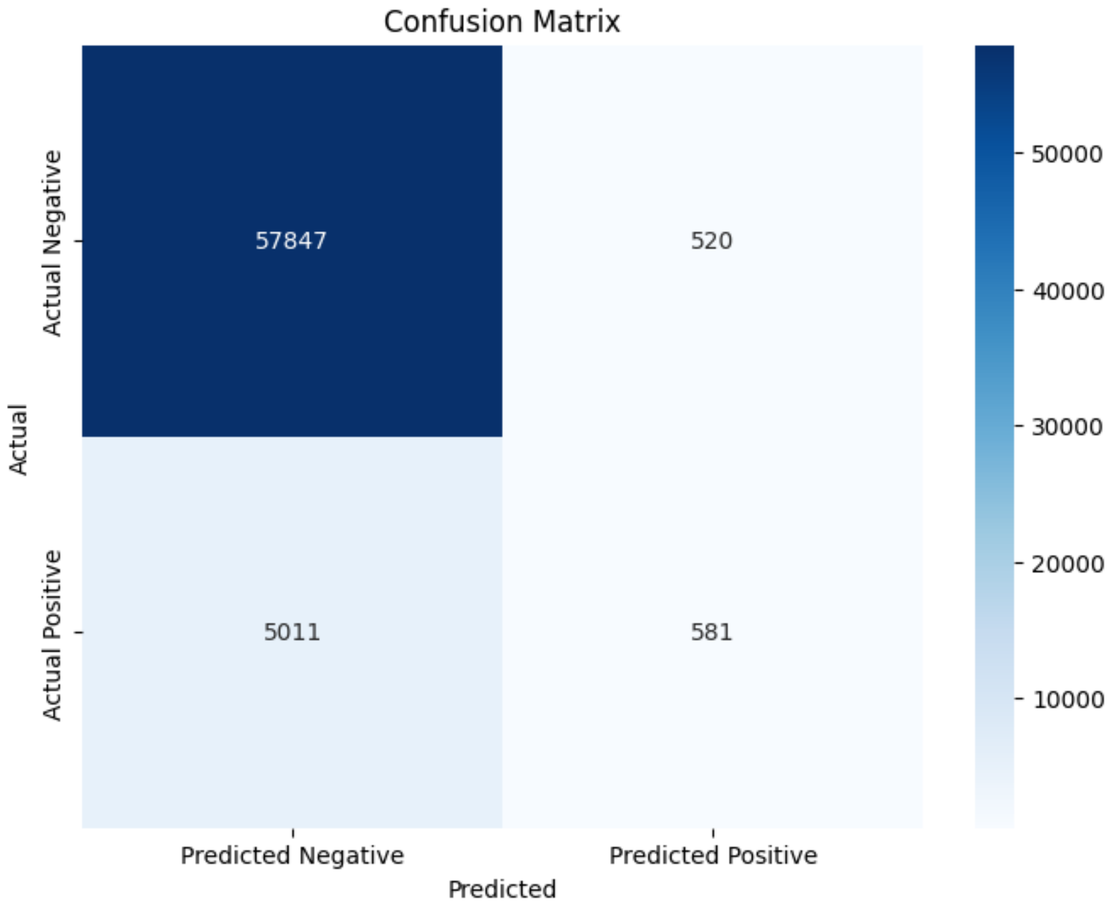
```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_features_2 (DenseFea  multiple                  0
 tures)

 dense_6 (Dense)             multiple                  6528

 dropout_2 (Dropout)         multiple                  0

 dense_7 (Dense)             multiple                  16512

 dense_8 (Dense)             multiple                  129

=================================================================
Total params: 23169 (90.50 KB)
Trainable params: 23169 (90.50 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
None
```

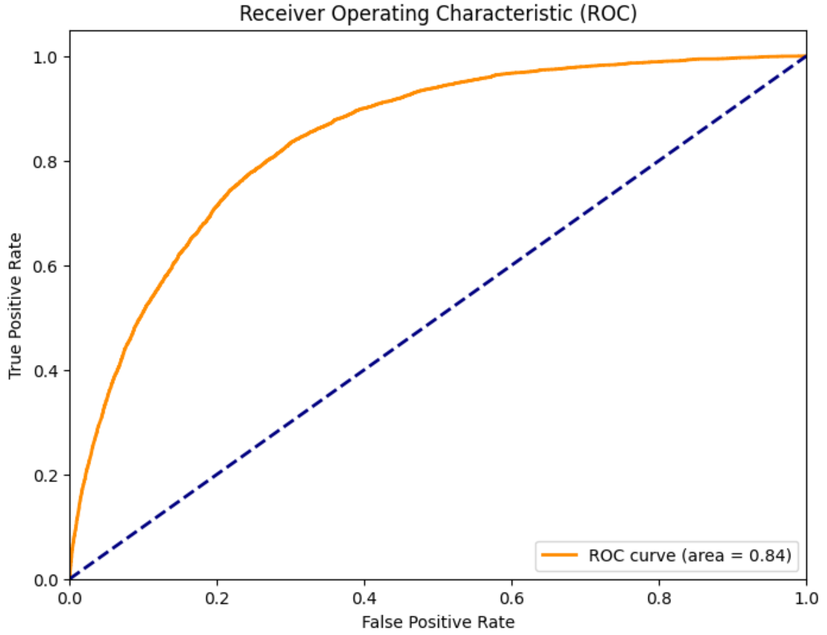
MLP for Kaggle Dataset





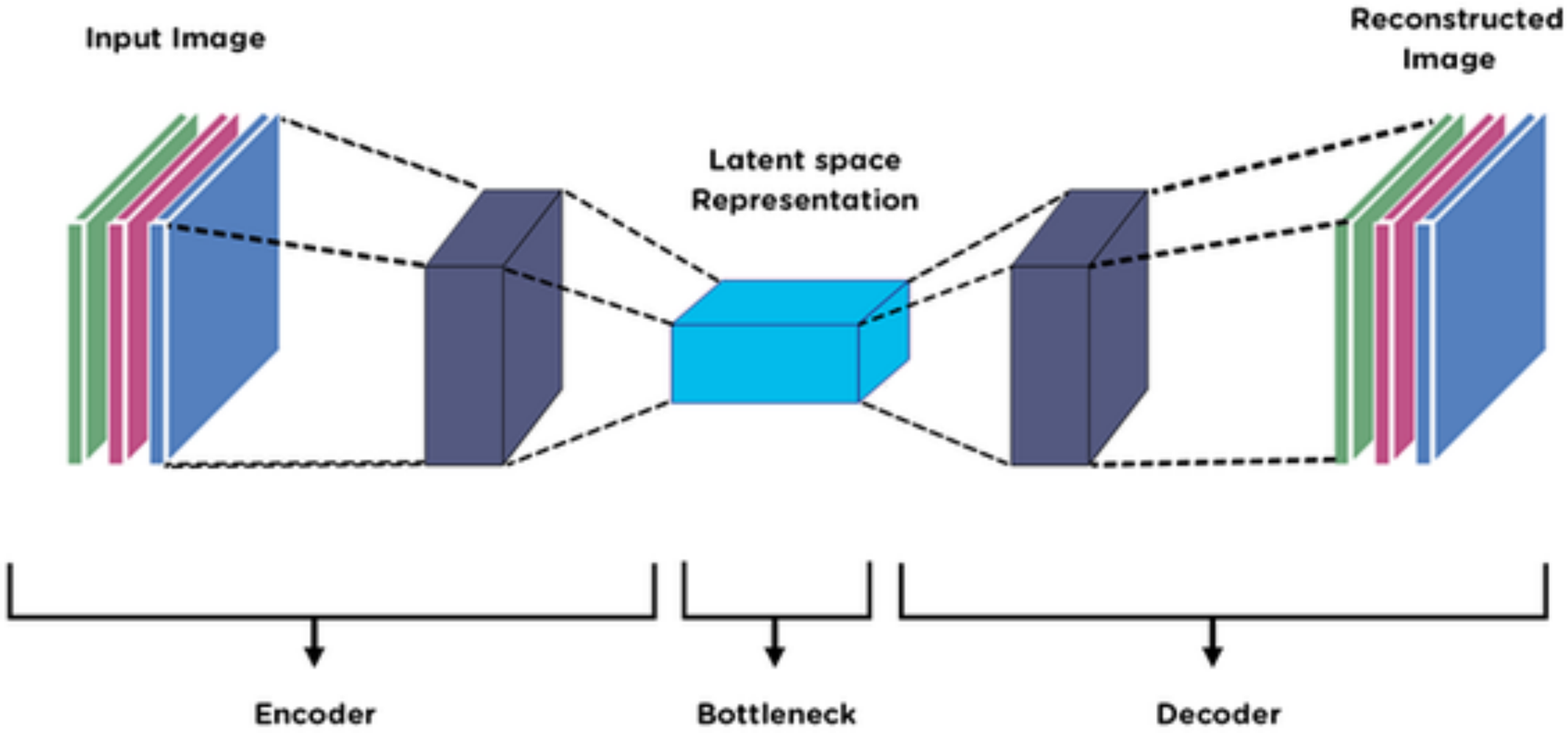
MLP for Kaggle Dataset





	precision	recall	f1-score	support
0	0.92	0.99	0.95	58367
1	0.53	0.10	0.17	5592
accuracy			0.91	63959
macro avg	0.72	0.55	0.56	63959
weighted avg	0.89	0.91	0.89	63959

AutoEncoders


Input Image
Reconstructed Image
Latent space Representation
Encoder
Bottleneck
Decoder

AutoEncoders

Autoencoders are a special kind of neural network utilized for unsupervised machine learning tasks, particularly those involving dimensionality reduction and anomaly detection. Here are the core aspects of autoencoders, synthesized from multiple sources:

Basic Functionality:
An autoencoder is designed to learn to copy its input to its output through a compressed representation. It first encodes the input into a lower-dimensional latent space, and then decodes this representation back to the original input format.

Architecture:
The architecture comprises three main parts: the encoder, the code, and the decoder. The encoder compresses the input data to form a code, the decoder then reconstructs the original data from this code.

Learning Method:
Autoencoders learn data encodings in an unsupervised manner, meaning they don't require labeled data for training. They aim to learn a lower-dimensional representation of higher-dimensional data, and are trained to capture the most significant features of the input data.

Use Cases:
They are widely used for dimensionality reduction, noise reduction, generating new data that's similar to the training data (generative modeling), and detecting anomalies in data.

Limitations:
There are limitations associated with autoencoders, such as the risk of learning too simple a function (the identity function) when the capacity of the network is too high.

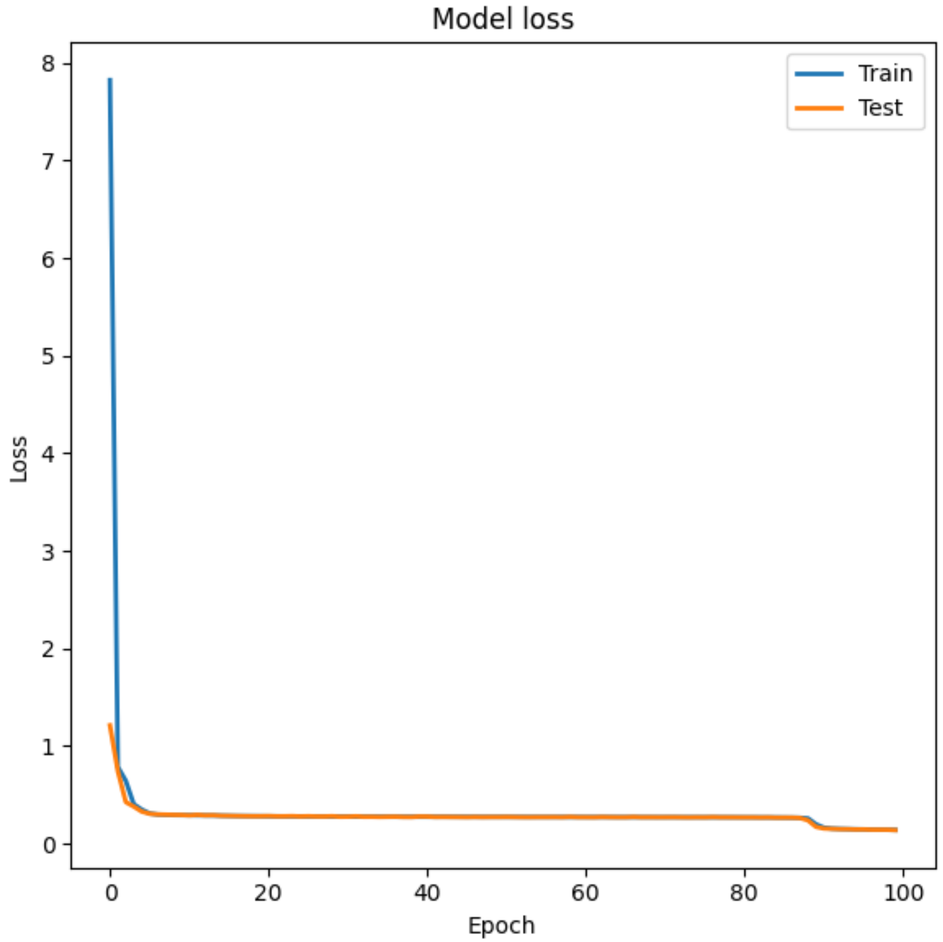
Model Architecture

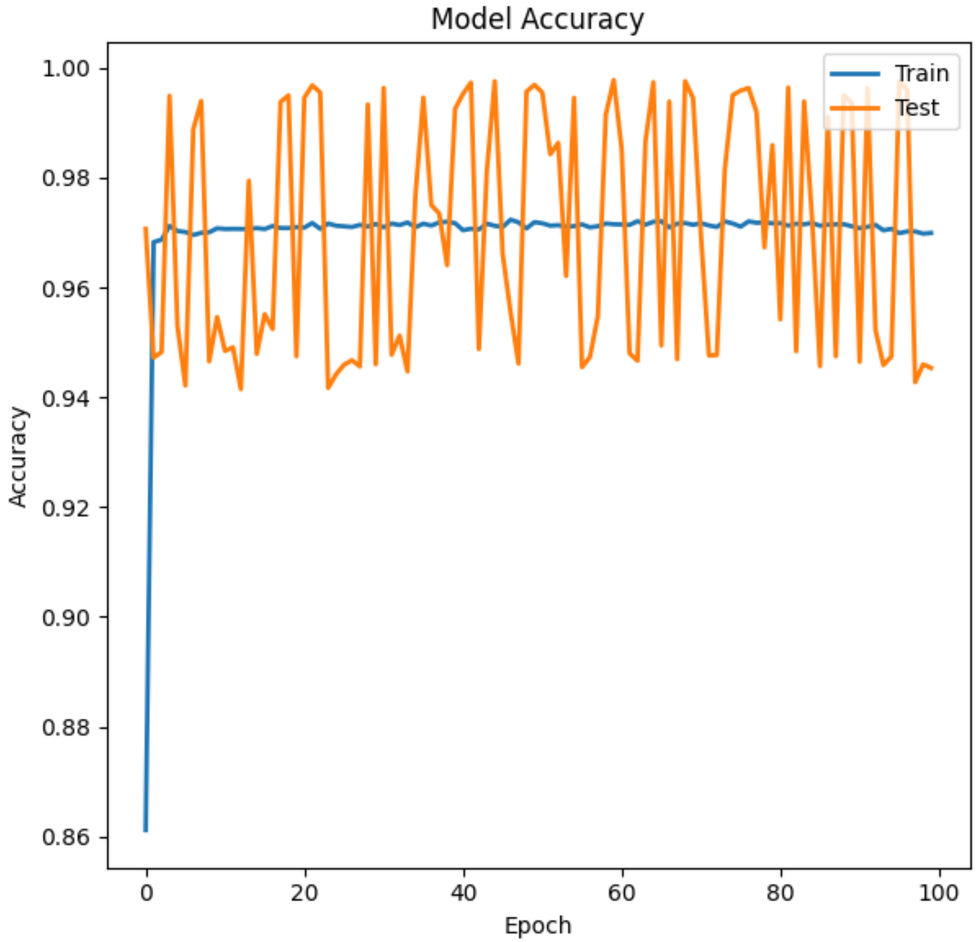
```
class Autoencoder(tf.keras.Model):
  def __init__(self, feature_count):
    self.feature_count = feature_count
    super(Autoencoder, self).__init__()
    self.encoder = tf.keras.Sequential([
      tf.keras.layers.Dense(self.feature_count, activation="relu"),
      tf.keras.layers.Dense(16, activation="relu"),
      tf.keras.layers.Dense(12, activation="relu")
      tf.keras.layers.Dense(12, activation="relu")
      ])

    self.decoder = tf.keras.Sequential([
      tf.keras.layers.Dense(12, activation="relu"),
      tf.keras.layers.Dense(12, activation="relu"),
      tf.keras.layers.Dense(16, activation="relu"),
      tf.keras.layers.Dense(self.feature_count, activation="linear")])

  def call(self, x):
    encoded = self.encoder(x)
    decoded = self.decoder(encoded)
    return decoded
```

AutoEncoder for Imputed Dataset

```
Records in Diseased Train Data =  18496
Records in Normal Train Data =  285345
Records in Diseased Test Data =  4723
Records in Normal Test Data= 71238
```

AutoEncoder for Imputed Dataset





AutoEncoder for Imputed Dataset

Anomaly Score:

The model would compute some kind of anomaly score for each data point in the dataset. This score would quantify how much of an outlier the data point is in comparison to the rest of the data. The score could be based on a variety of metrics, such as reconstruction error in autoencoders.

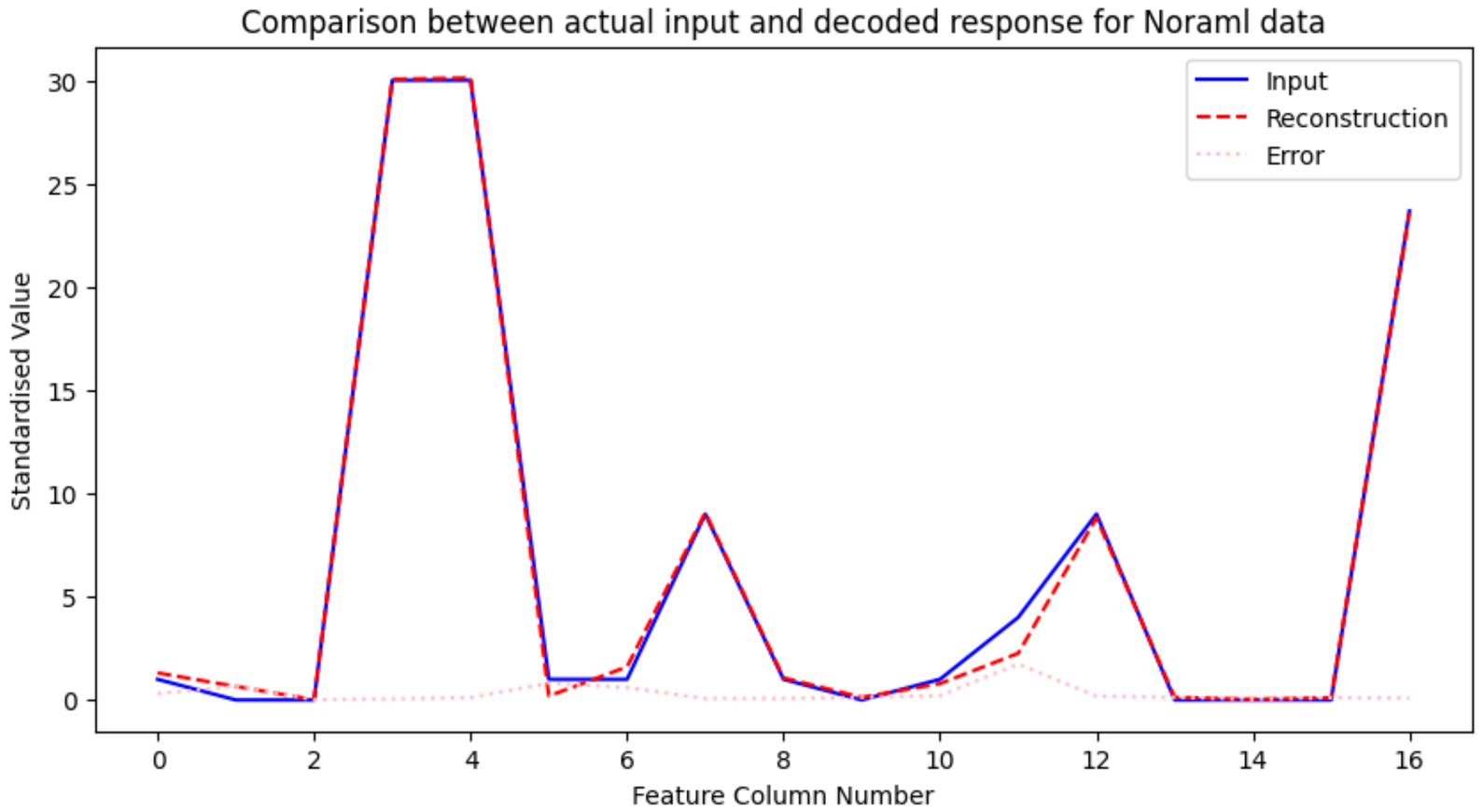
Threshold Application:

The threshold of 0.296 acts as a cutoff value. Scores above this threshold would indicate an anomaly, while scores below would suggest normal behaviour. The specific choice of threshold is crucial and would depend on the domain and the cost of false positives (normal behaviour wrongly identified as an anomaly) versus false negatives (anomalies not detected).

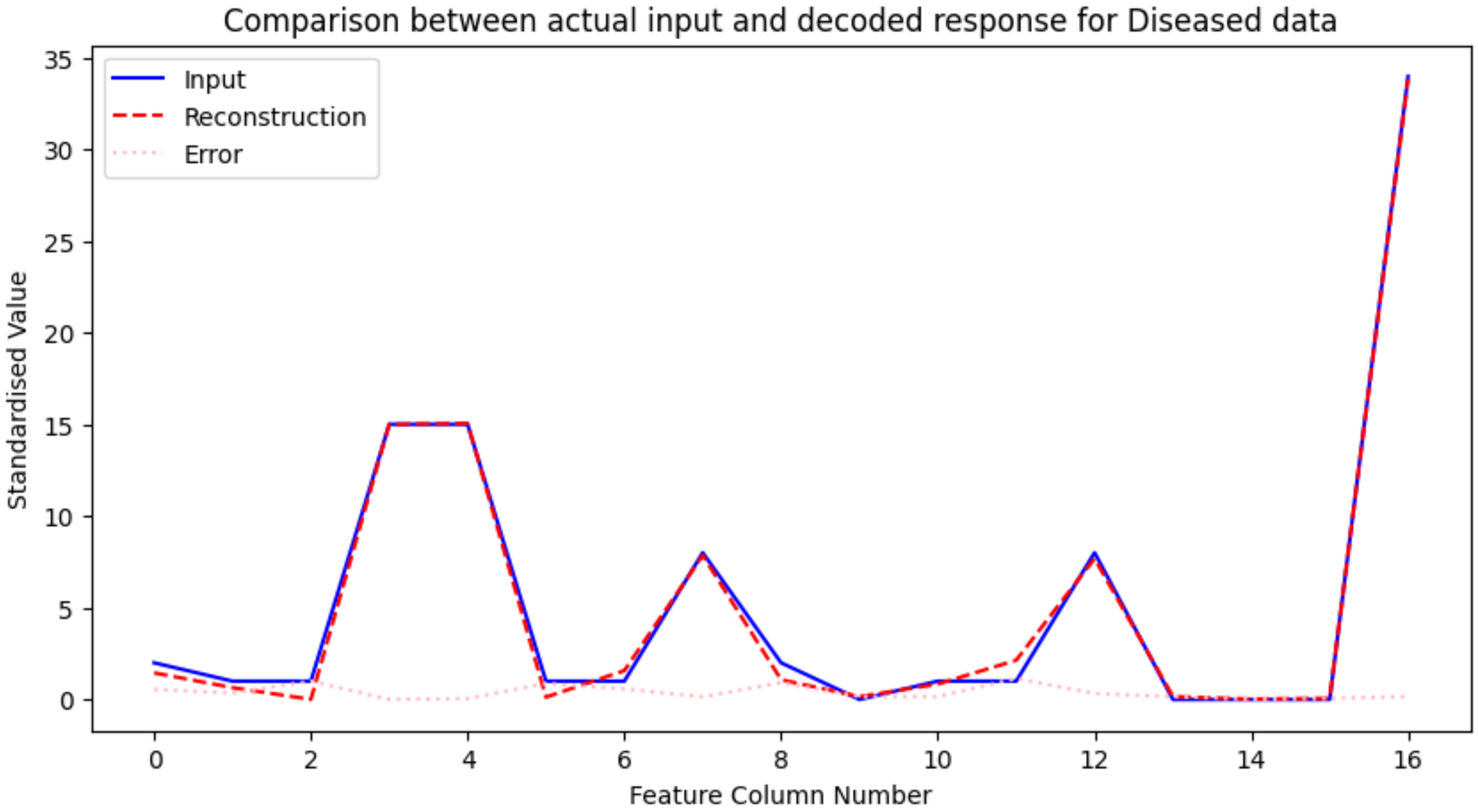
Model Performance:

The chosen threshold would significantly affect the model's performance. It's a balance between sensitivity (the ability to detect true anomalies) and specificity (the ability to not flag normal data as anomalous). The optimal threshold is usually determined by evaluating the model on a validation set and considering the trade-off between false positives and false negatives.

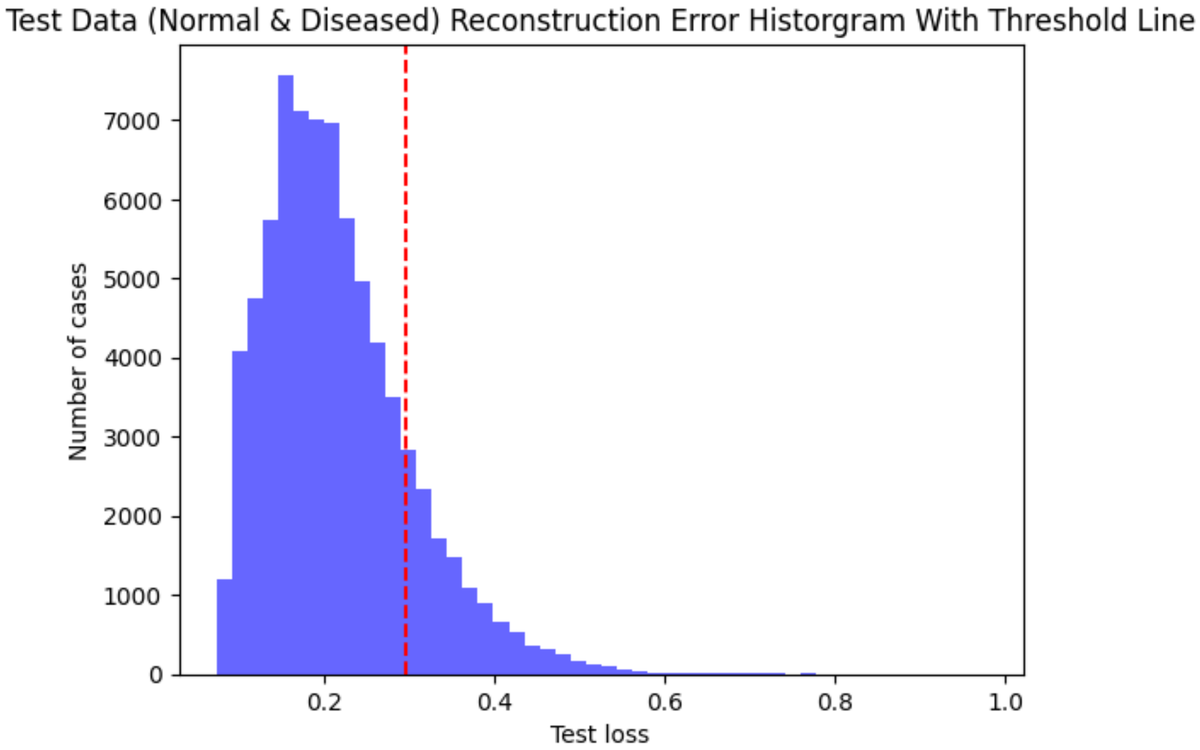
Comparison plot for a random sample from the test set



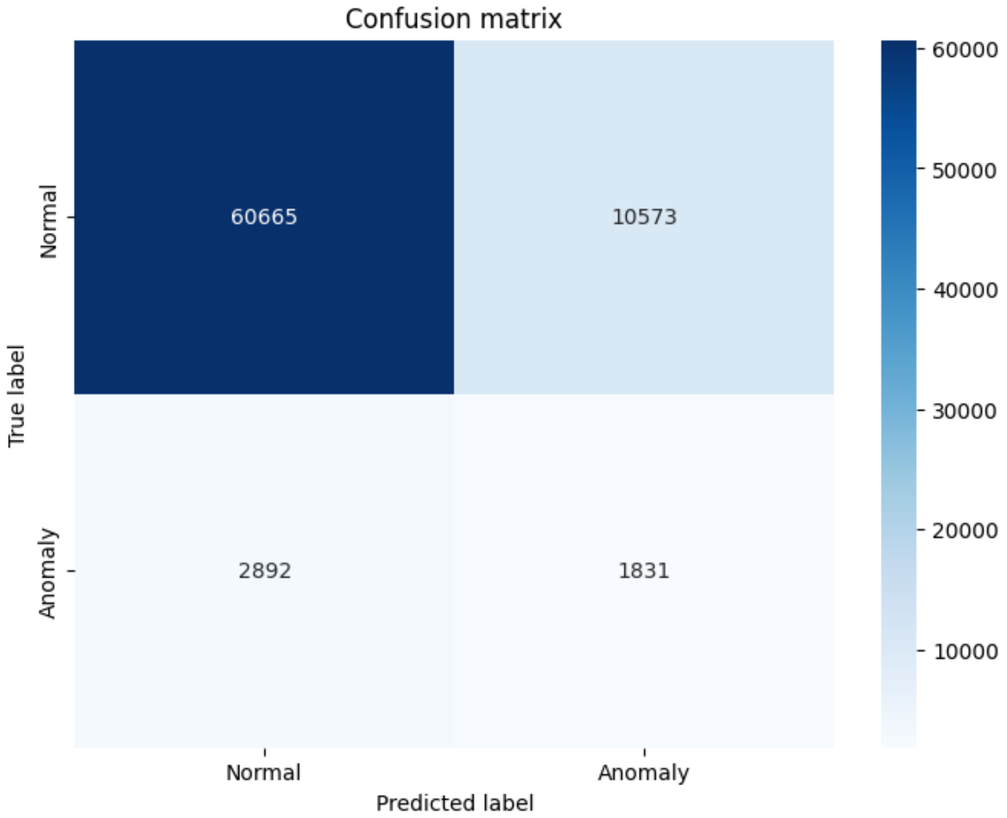
Compariosn plot for decoded data

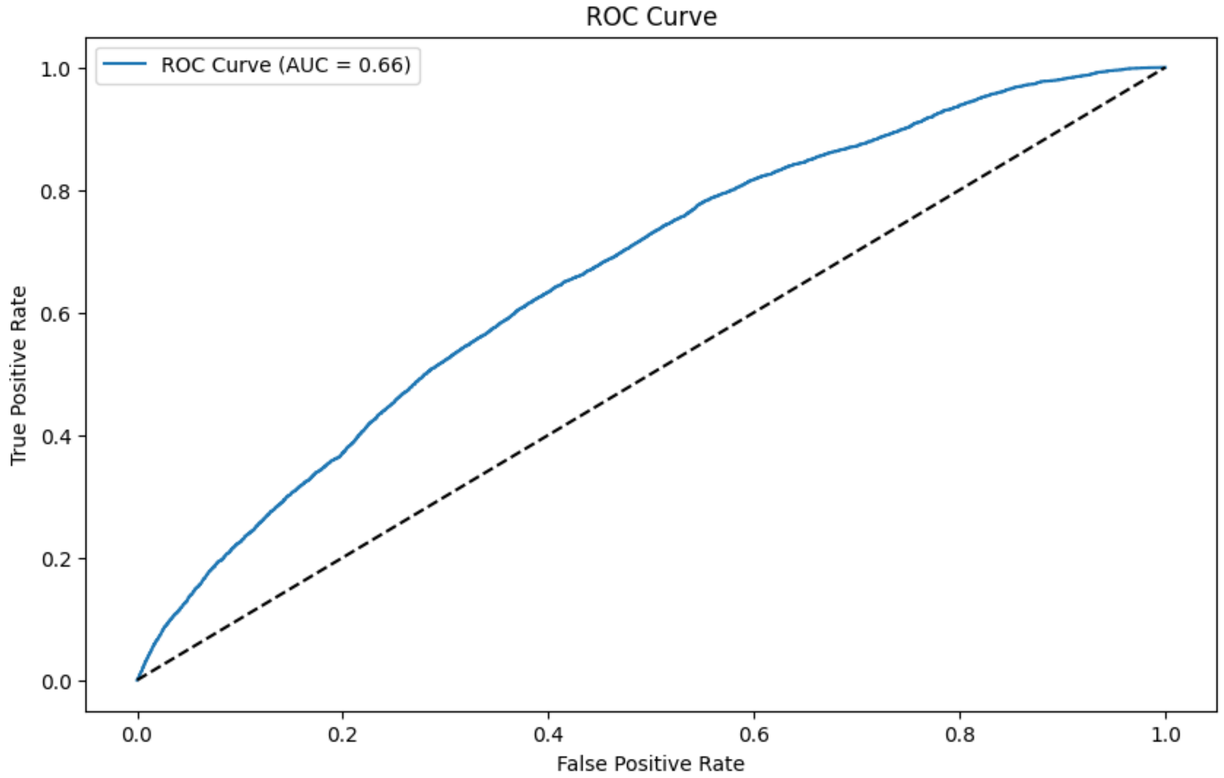


Reconstruction Error



Result


Confusion matrix
Normal
Anomaly
True label
60665
10573
2892
1831
Normal
Anomaly
Predicted label
60000
50000
40000
30000
20000
10000


ROC Curve
ROC Curve (AUC = 0.66)
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0

Final Result

	Decision Tree	Logistic Regression	MLP	AutoEncoder
Imputed Dataset	89%	93.80%	94%	82%
Oversampled	96%	75%		
Kaggle	86%	91.55%	91%	