## 1. Problem:

In the past few years more people have been getting Diabetes which is a big worry we will study and analyze patients  data to understand why. Our project goal to study this data carefully and analyze it to find out what the possible factors and risks that leads to Diabetes and how to predict if someone might get it. By doing this, we hope to help people act early to avoid these serious health issues.

## 2. Data Mining Task:

Our The data mining task is classification and clustering We will analyze the Diabetes production

## 3. Data:

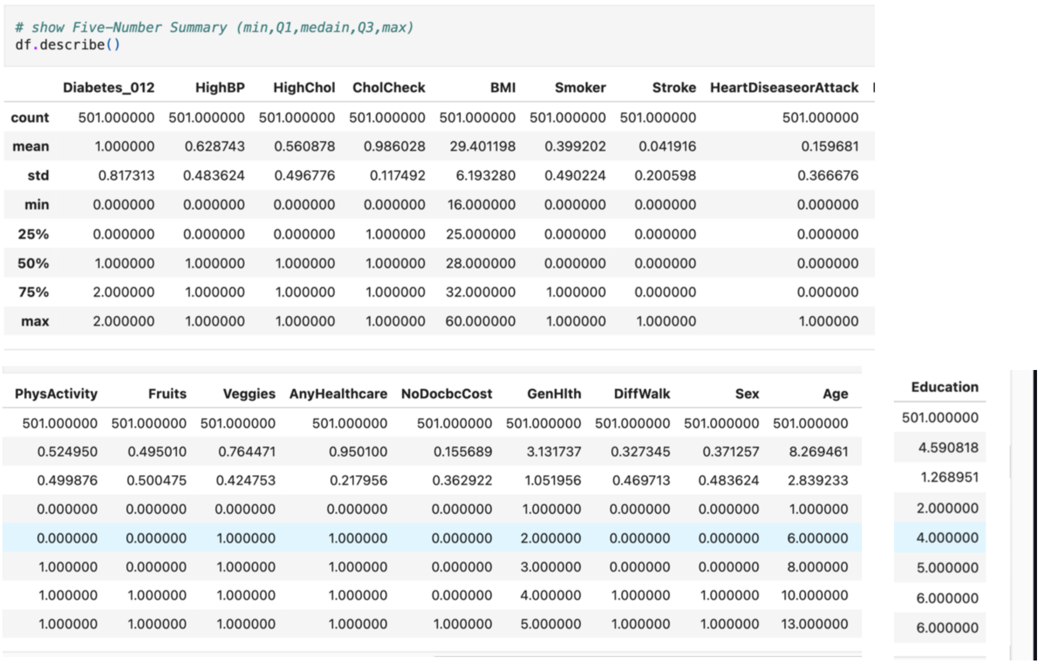
dataset description: This dataset contains information about various attributes related to health and lifestyle that provide a broad perspective on factors that can influence diabetes.

The source: https://www.kaggle.com/datasets/alexteboul/diabetes-health-indicators-dataset/data

Number of objects =501
Number of attributes= 18

| Attributes | Data Type | possible values | Missing values |
|---|---|---|---|
| Diabetes_012 | Categorical | 0, 1, 2 | NONE |
| HighBP | Binary | 1 (yes), 0 (No) | NONE |
| HighChol | Binary | 1 (yes), 0 (No) | NONE |
| CholCheck | Binary | 1 (yes), 0 (No) | NONE |
| BMI | Numeric (Ratio) | N/A | NONE |
| Smoker | Binary | 1 (yes), 0 (No) | NONE |
| Stroke | Binary | 1 (yes), 0 (No) | NONE |
| HeartDiseaseorAtta | Binary | 1 (yes), 0 (No) | NONE |
| PhysActivity | Binary | 1 (yes), 0 (No) | NONE |
| Fruits | Binary | 1 (yes), 0 (No) | NONE |
| Veggies | Binary | 1 (yes), 0 (No) | NONE |
| AnyHealthcare | Binary | 1 (yes), 0 (No) | NONE |
| NoDocbcCost | Binary | 1 (yes), 0 (No) | NONE |
| GenHlth | Ordinal | 1-5 | NONE |
| DiffWalk | Binary | 1 (yes), 0 (No) | NONE |
| Sex | Binary | 1 (yes), 0 (No) | NONE |
| Age | Categorical | 13 levels | NONE |
| Education | Categorical | 1-6 | NONE |

## - Statistical measures

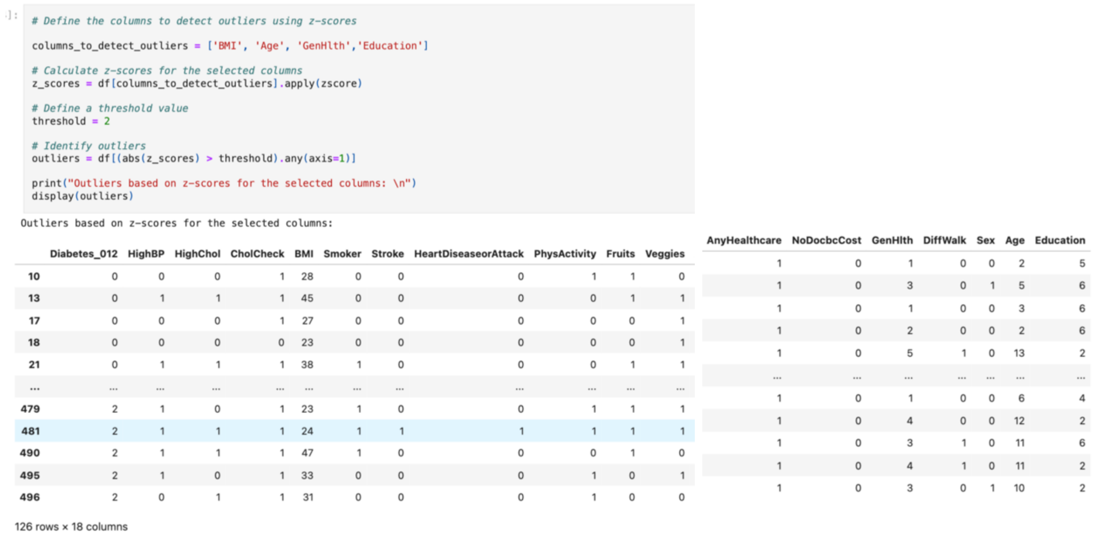


```
# show Five-Number Summary (min,Q1,medain,Q3,max)
df.describe()
```

| | Diabetes_012 | HighBP | HighChol | CholCheck | BMI | Smoker | Stroke | HeartDiseaseorAttack |
|---|---|---|---|---|---|---|---|---|
| count | 501.000000 | 501.000000 | 501.000000 | 501.000000 | 501.000000 | 501.000000 | 501.000000 | 501.000000 |
| mean | 1.000000 | 0.628743 | 0.560878 | 0.986028 | 29.401198 | 0.399202 | 0.041916 | 0.159681 |
| std | 0.817313 | 0.483624 | 0.496776 | 0.117492 | 6.193280 | 0.490224 | 0.200598 | 0.366676 |
| min | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 16.000000 | 0.000000 | 0.000000 | 0.000000 |
| 25% | 0.000000 | 0.000000 | 0.000000 | 1.000000 | 25.000000 | 0.000000 | 0.000000 | 0.000000 |
| 50% | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 28.000000 | 0.000000 | 0.000000 | 0.000000 |
| 75% | 2.000000 | 1.000000 | 1.000000 | 1.000000 | 32.000000 | 1.000000 | 0.000000 | 0.000000 |
| max | 2.000000 | 1.000000 | 1.000000 | 1.000000 | 60.000000 | 1.000000 | 1.000000 | 1.000000 |

| | PhysActivity | Fruits | Veggies | AnyHealthcare | NoDocbcCost | GenHlth | DiffWalk | Sex | Age | Education |
|---|---|---|---|---|---|---|---|---|---|---|
| count | 501.000000 | 501.000000 | 501.000000 | 501.000000 | 501.000000 | 501.000000 | 501.000000 | 501.000000 | 501.000000 | 501.000000 |
| mean | 0.524950 | 0.495010 | 0.764471 | 0.950100 | 0.155689 | 3.131737 | 0.327345 | 0.371257 | 8.269461 | 4.590818 |
| std | 0.499876 | 0.500475 | 0.424753 | 0.217956 | 0.362922 | 1.051956 | 0.469713 | 0.483624 | 2.839233 | 1.268951 |
| min | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 1.000000 | 0.000000 | 0.000000 | 1.000000 | 2.000000 |
| 25% | 0.000000 | 0.000000 | 1.000000 | 1.000000 | 0.000000 | 2.000000 | 0.000000 | 0.000000 | 6.000000 | 4.000000 |
| 50% | 1.000000 | 0.000000 | 1.000000 | 1.000000 | 0.000000 | 3.000000 | 0.000000 | 0.000000 | 8.000000 | 5.000000 |
| 75% | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 0.000000 | 4.000000 | 1.000000 | 1.000000 | 10.000000 | 6.000000 |
| max | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 5.000000 | 1.000000 | 1.000000 | 13.000000 | 6.000000 |

## - Outliers



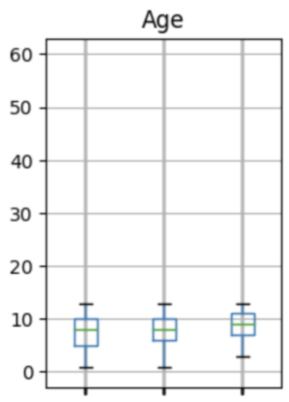
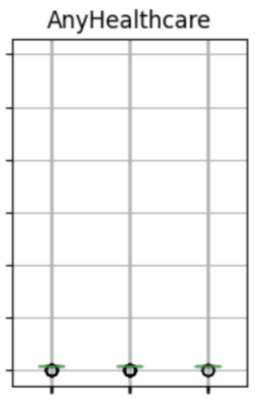
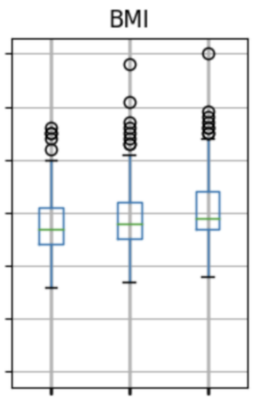
```
# Define the columns to detect outliers using z-scores

columns_to_detect_outliers = ['BMI', 'Age', 'GenHlth','Education']

# Calculate z-scores for the selected columns
z_scores = df[columns_to_detect_outliers].apply(zscore)

# Define a threshold value
threshold = 2

# Identify outliers
outliers = df[(abs(z_scores) > threshold).any(axis=1)]

print("Outliers based on z-scores for the selected columns: \n")
display(outliers)
```

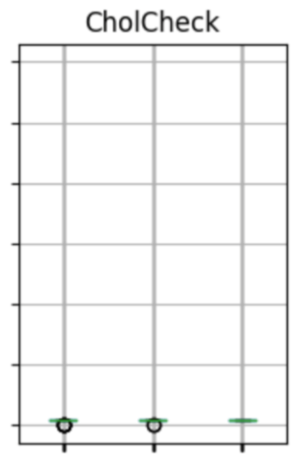
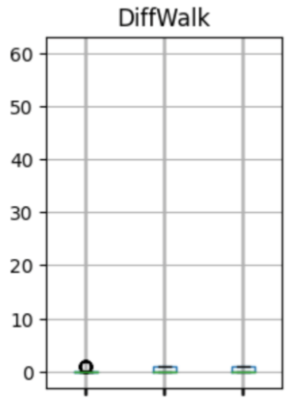
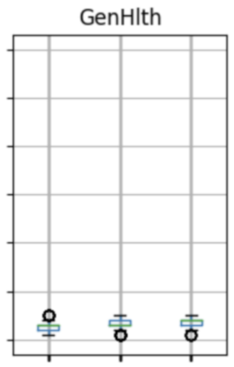
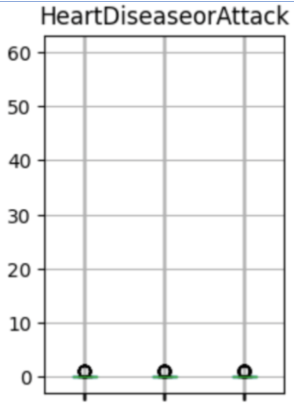
```
Outliers based on z-scores for the selected columns:
```

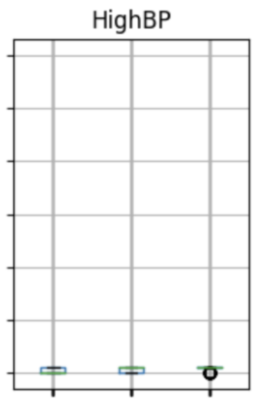
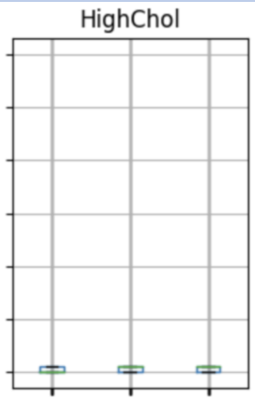
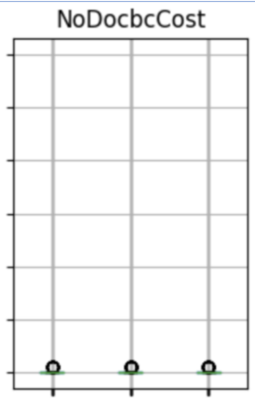
| | Diabetes_012 | HighBP | HighChol | CholCheck | BMI | Smoker | Stroke | HeartDiseaseorAttack | PhysActivity | Fruits | Veggies | AnyHealthcare | NoDocbcCost | GenHlth | DiffWalk | Sex | Age | Education |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 10 | 0 | 0 | 0 | 1 | 28 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 2 | 5 |
| 13 | 0 | 1 | 1 | 1 | 45 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 3 | 0 | 1 | 5 | 6 |
| 17 | 0 | 0 | 0 | 1 | 27 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 3 | 6 |
| 18 | 0 | 0 | 0 | 0 | 23 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 2 | 0 | 0 | 2 | 6 |
| 21 | 0 | 1 | 1 | 1 | 38 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 5 | 1 | 0 | 13 | 2 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 479 | 2 | 1 | 0 | 1 | 23 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 6 | 4 |
| 481 | 2 | 1 | 1 | 1 | 24 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 4 | 0 | 0 | 12 | 2 |
| 490 | 2 | 1 | 1 | 1 | 47 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 3 | 1 | 0 | 11 | 6 |
| 495 | 2 | 1 | 0 | 1 | 33 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 4 | 1 | 0 | 11 | 2 |
| 496 | 2 | 0 | 1 | 1 | 31 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 3 | 0 | 1 | 10 | 2 |

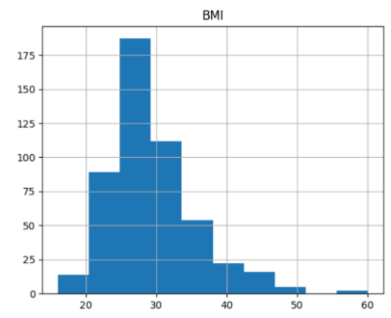
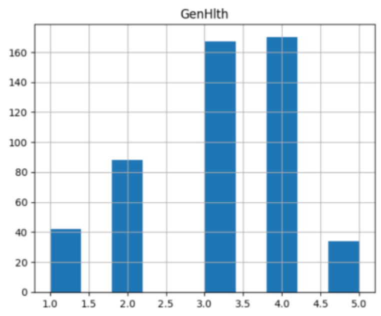
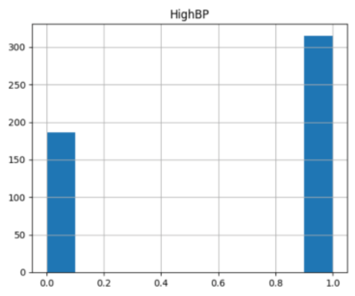
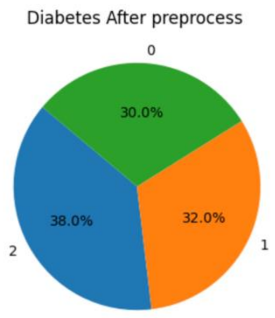
126 rows × 18 columns

- boxplots

| Graph | Description |
|---|---|
|  | The boxplot illustrates that the distribution of age suggests a diverse range of respondents, with a median age around 8 and a wide range of ages represented. This diversity in ages within the dataset indicates a broad demographic spectrum among the participants. |
|  | Boxplot illustrates that the majority of respondents have access to healthcare, with a significant portion reporting no out-of-pocket costs for doctor visits |
|  | Boxplot illustrates that the distribution of BMI values appears fairly symmetrical, with a few outliers on the higher end. Most individuals seem to have a BMI within the 25th and 75th percentiles. |

| | |
|---|---|
|  | Boxplot illustrates that most observations fall within the upper quartiles, indicating frequent cholesterol checks among respondents. |
|  | Boxplot illustrates that difficulty walking seems relatively low among respondents, with most reporting no difficulty. |
|  | Boxplot illustrates that respondents generally perceive their general health positively, with most reporting moderate to good health. |
|  | Boxplot illustrates that there's a noticeable prevalence of heart disease or heart attacks, with the median and mean indicating a significant portion of the sample reporting such conditions. |

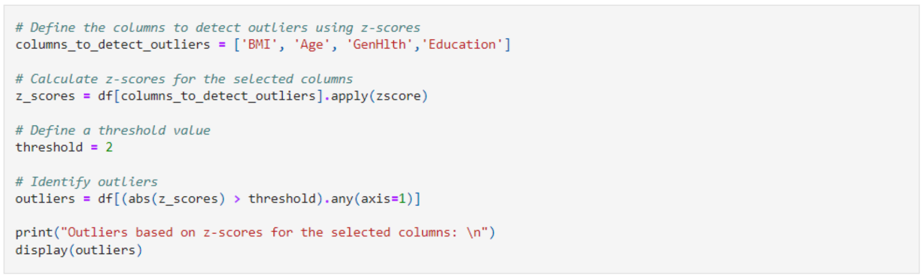
| | |
|---|---|
|  | Boxplot illustrates that the distribution seems slightly positively skewed, with the median and mean indicating a moderate prevalence of high blood pressure. |
|  | Boxplot illustrates that the distribution is similar to that of high blood pressure, indicating a moderate prevalence of high cholesterol. |
|  | Boxplot illustrates that the majority of respondents have access to healthcare, with a significant portion reporting no out-of-pocket costs for doctor visits." |

| Graph | Description |
|---|---|
|  | illustrated in the graph the most frequent BMI are between 25 - 29 we can conclude from that this BMI falls within the overweight range. |
|  | The histogram represents the frequency of general health for Participants in dataset. After observation, we noticed that the most values lie in(3,4) which is around (Fair – Good) |
|  | The histogram represents the frequency of High blood pressure for Participants in dataset. After observation, we noticed that the most of Participants got High blood pressure |
|  | The pie chart graph represents that the highest value is for Participants who got diabetes, While the lowest value represents Participants who got no diabetes. |

## 4. Data preprocessing :

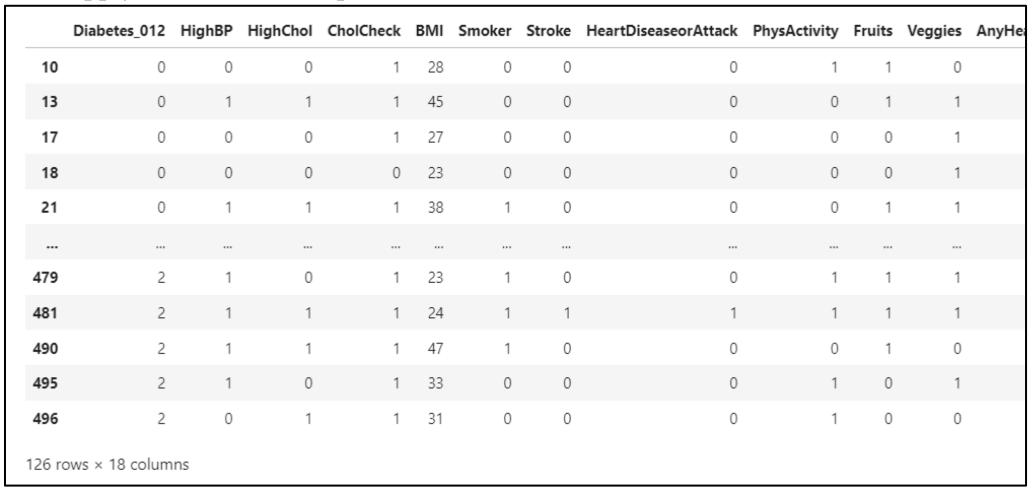
Encoding:
we did not use this process since we do not have nominal or large numbers attributes so we do not need to use it .

Cleaning:
We apply this process on four attributes (BMI, Age, GenHlth, Education)
Since they are the only numeric data in our dataset .
First we try to detect the outliers for them using z-score  :



```python
# Define the columns to detect outliers using z-scores
columns_to_detect_outliers = ['BMI', 'Age', 'GenHlth','Education']

# Calculate z-scores for the selected columns
z_scores = df[columns_to_detect_outliers].apply(zscore)

# Define a threshold value
threshold = 2

# Identify outliers
outliers = df[(abs(z_scores) > threshold).any(axis=1)]

print("Outliers based on z-scores for the selected columns: \n")
display(outliers)
```

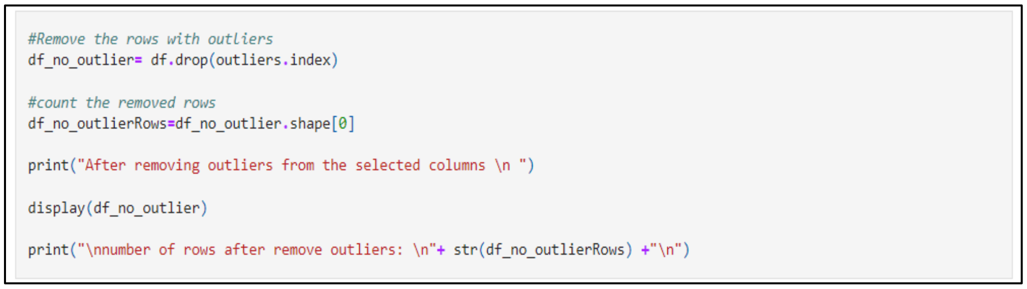
After apply this code the output will be a number of detected outliers rows “126 rows”  :



|  | Diabetes_012 | HighBP | HighChol | CholCheck | BMI | Smoker | Stroke | HeartDiseaseorAttack | PhysActivity | Fruits | Veggies | AnyHe |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **10** | 0 | 0 | 0 | 1 | 28 | 0 | 0 | 0 | 1 | 1 | 0 |  |
| **13** | 0 | 1 | 1 | 1 | 45 | 0 | 0 | 0 | 0 | 1 | 1 |  |
| **17** | 0 | 0 | 0 | 1 | 27 | 0 | 0 | 0 | 0 | 0 | 1 |  |
| **18** | 0 | 0 | 0 | 0 | 23 | 0 | 0 | 0 | 0 | 0 | 1 |  |
| **21** | 0 | 1 | 1 | 1 | 38 | 1 | 0 | 0 | 0 | 1 | 1 |  |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |  |
| **479** | 2 | 1 | 0 | 1 | 23 | 1 | 0 | 0 | 1 | 1 | 1 |  |
| **481** | 2 | 1 | 1 | 1 | 24 | 1 | 1 | 1 | 1 | 1 | 1 |  |
| **490** | 2 | 1 | 1 | 1 | 47 | 1 | 0 | 0 | 0 | 1 | 0 |  |
| **495** | 2 | 1 | 0 | 1 | 33 | 0 | 0 | 0 | 1 | 0 | 1 |  |
| **496** | 2 | 0 | 1 | 1 | 31 | 0 | 0 | 0 | 1 | 0 | 0 |  |

126 rows × 18 columns

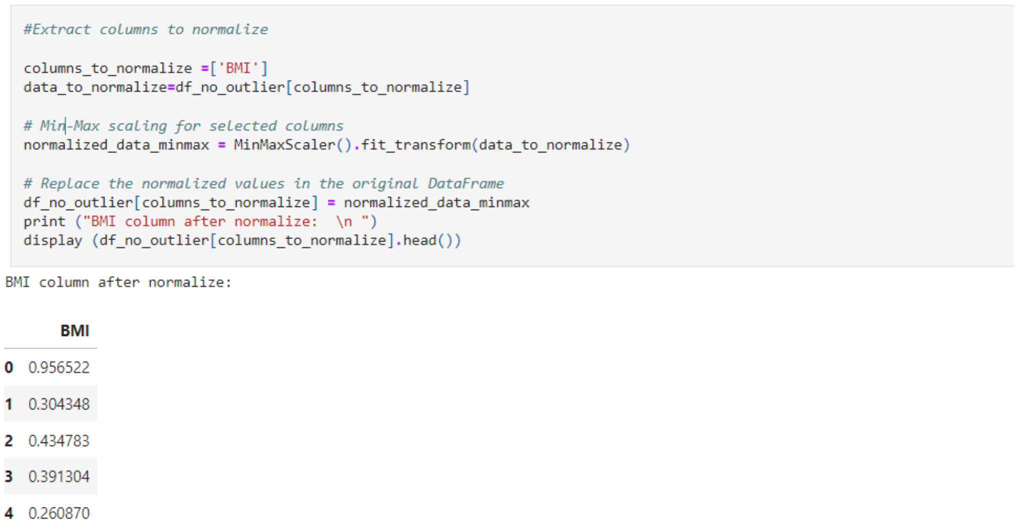
Second we removed all detected outliers rows from our dataset and print the new row number and save the new dataset in  df_no_outlier:



```python
#Remove the rows with outliers
df_no_outlier= df.drop(outliers.index)

#count the removed rows
df_no_outlierRows=df_no_outlier.shape[0]

print("After removing outliers from the selected columns \n ")

display(df_no_outlier)

print("\nnumber of rows after remove outliers: \n"+ str(df_no_outlierRows) +"\n")
```

```
number of rows after remove outliers:
375
```

Normalization :
We apply this process to (BMI) sence it is the only attribute that is not categorize :

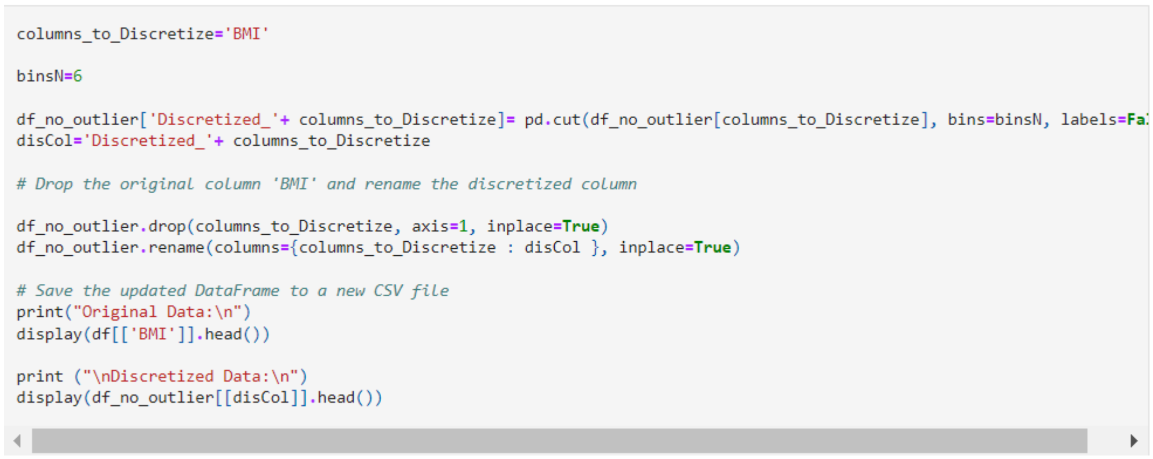


```
#Extract columns to normalize

columns_to_normalize =['BMI']
data_to_normalize=df_no_outlier[columns_to_normalize]

# Min-Max scaling for selected columns
normalized_data_minmax = MinMaxScaler().fit_transform(data_to_normalize)

# Replace the normalized values in the original DataFrame
df_no_outlier[columns_to_normalize] = normalized_data_minmax
print ("BMI column after normalize:  \n ")
display (df_no_outlier[columns_to_normalize].head())
```

```
BMI column after normalize:
```

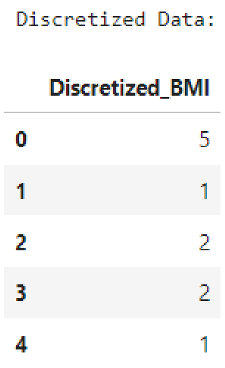
| | BMI |
|---|---|
| 0 | 0.956522 |
| 1 | 0.304348 |
| 2 | 0.434783 |
| 3 | 0.391304 |
| 4 | 0.260870 |

Discretization:
 categorize the 'BMI' values into a smaller number of bins for easier interpretation and analysis:



```
columns_to_Discretize='BMI'

binsN=6

df_no_outlier['Discretized_'+ columns_to_Discretize]= pd.cut(df_no_outlier[columns_to_Discretize], bins=binsN, labels=Fa
disCol='Discretized_'+ columns_to_Discretize

# Drop the original column 'BMI' and rename the discretized column

df_no_outlier.drop(columns_to_Discretize, axis=1, inplace=True)
df_no_outlier.rename(columns={columns_to_Discretize : disCol }, inplace=True)

# Save the updated DataFrame to a new CSV file
print("Original Data:\n")
display(df[['BMI']].head())

print ("\nDiscretized Data:\n")
display(df_no_outlier[[disCol]].head())
```

after Discretize BMI we change its column to Discretized_BMI with the new values :

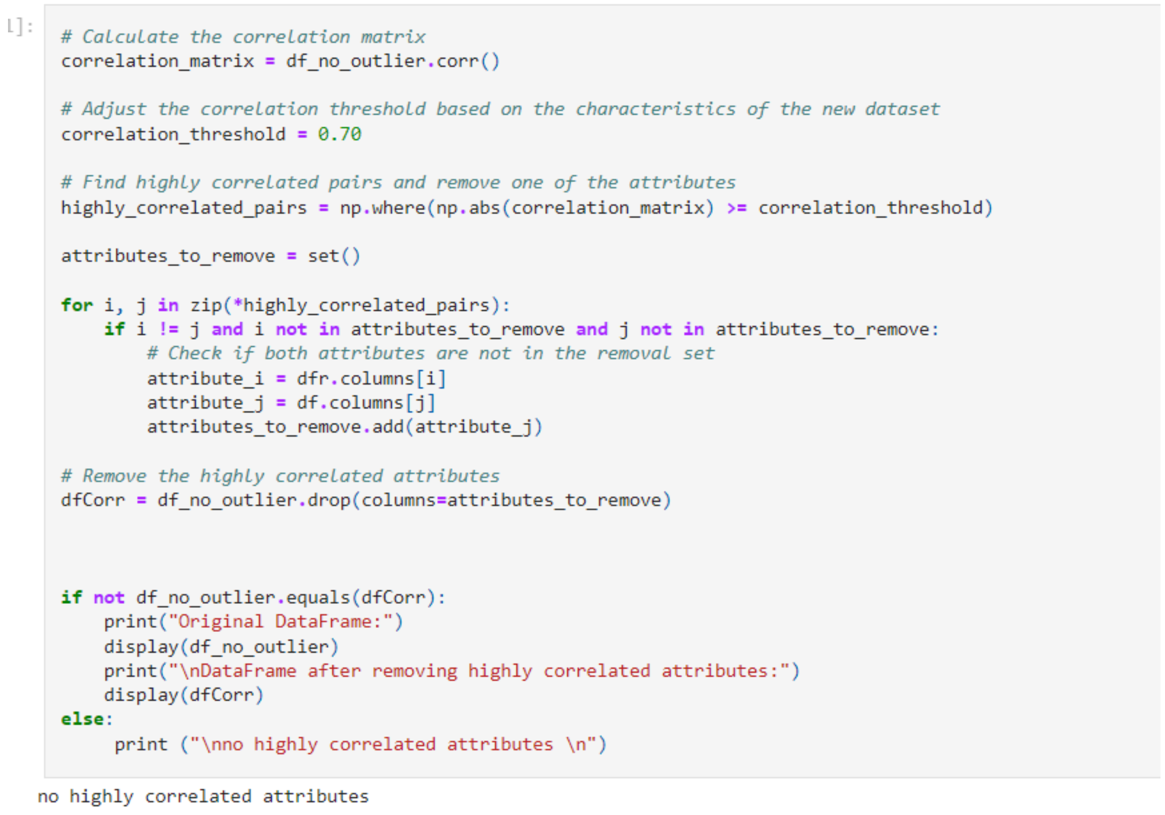


```
Discretized Data:
```

| | Discretized_BMI |
|---|---|
| 0 | 5 |
| 1 | 1 |
| 2 | 2 |
| 3 | 2 |
| 4 | 1 |

Correlation:

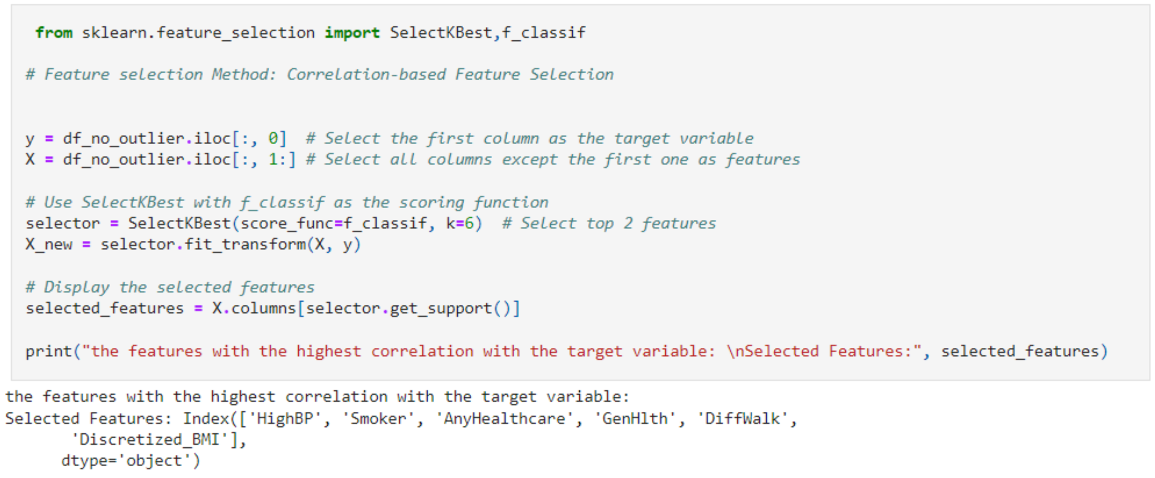
In this process we tried to remove the highly correlated attributes to prevent multicollinearity issues and enhance the efficiency of predictive models by reducing redundancy in the dataset.
But there is no highly correlated attributes:



```
[ ]:
# Calculate the correlation matrix
correlation_matrix = df_no_outlier.corr()

# Adjust the correlation threshold based on the characteristics of the new dataset
correlation_threshold = 0.70

# Find highly correlated pairs and remove one of the attributes
highly_correlated_pairs = np.where(np.abs(correlation_matrix) >= correlation_threshold)

attributes_to_remove = set()

for i, j in zip(*highly_correlated_pairs):
    if i != j and i not in attributes_to_remove and j not in attributes_to_remove:
        # Check if both attributes are not in the removal set
        attribute_i = dfr.columns[i]
        attribute_j = df.columns[j]
        attributes_to_remove.add(attribute_j)

# Remove the highly correlated attributes
dfCorr = df_no_outlier.drop(columns=attributes_to_remove)



if not df_no_outlier.equals(dfCorr):
    print("Original DataFrame:")
    display(df_no_outlier)
    print("\nDataFrame after removing highly correlated attributes:")
    display(dfCorr)
else:
     print ("\nno highly correlated attributes \n")
```

```
no highly correlated attributes
```

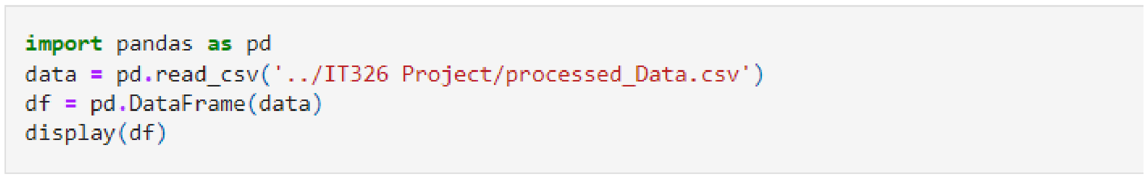
Feature selection :
We apply this process to identify the most relevant 6 features that have the highest correlation with the target variable, aiding in model performance improvement.



```
 from sklearn.feature_selection import SelectKBest,f_classif

# Feature selection Method: Correlation-based Feature Selection


y = df_no_outlier.iloc[:, 0]  # Select the first column as the target variable
X = df_no_outlier.iloc[:, 1:] # Select all columns except the first one as features

# Use SelectKBest with f_classif as the scoring function
selector = SelectKBest(score_func=f_classif, k=6)  # Select top 2 features
X_new = selector.fit_transform(X, y)

# Display the selected features
selected_features = X.columns[selector.get_support()]

print("the features with the highest correlation with the target variable: \nSelected Features:", selected_features)
```

```
the features with the highest correlation with the target variable:
Selected Features: Index(['HighBP', 'Smoker', 'AnyHealthcare', 'GenHlth', 'DiffWalk',
       'Discretized_BMI'],
      dtype='object')
```

Save processed data :
The last processing step is saving the data set after all processes to use it in



```python
import pandas as pd
data = pd.read_csv('../IT326 Project/processed_Data.csv')
df = pd.DataFrame(data)
display(df)
```

Explain why did you (or didn’t you) choose to apply data preprocessing. If your data requires preprocessing, explain your process. Provide justification for the techniques you applied. In your submission, include your raw dataset, as well as your processed dataset. You can apply either data cleaning or data transformation or both. For each, you should describe why you applied it and how and on which attributes. You need to provide a snapshot of the raw dataset, as well as preprocessed dataset. If your data did not require preprocessing, you should justify why and illustrate that using code or pictures. NOTE: you should submit at least three preprocessing different tasks (not including: removing attributes or splitting the data set).

## 5. Data Mining Technique:

Describe the data mining techniques (classification and clustering) that you will apply to your dataset, why and how (i.e. specify the Python packages and methods).

**1- Split the data Technique**

Before applying the Classification Techniques, we will divide the dataset into 2 distinct parts: a training set and a testing set.
**training set:** used to train the classification model.
**testing set:** used to evaluate the model's performance on unseen data.

**2- Classification Techniques**

Classification is the process of categorizing data into predefined classes or labels. Here's how we will apply classification techniques to our dataset:

Decision Trees

**Why**: Decision trees are intuitive and easy to interpret. They work well with both numerical and categorical data.

**How**: use the 'scikit-learn' library in Python, specifically the "DecisionTreeClassifier" class.

**3- Clustering Techniques**

Clustering is the process of grouping similar data points together. Here's how we will apply clustering techniques

K-Means Clustering

**Why**: K-Means is one of the most popular clustering algorithms, efficient for large datasets.

**How**: Python's 'scikit-learn' library offers K-Means clustering through the 'KMeans' class.

**4- Python Packages and Methods**

1- scikit-learn: A powerful library for machine learning in Python, providing implementations for various classification and clustering algorithms
2- scipy: Useful for hierarchical clustering, providing functions like 'linkage' and 'dendrogram'
3- numpy: Often used for numerical operations and data manipulation
4- pandas: Ideal for data manipulation and preprocessing
5- matplotlib and seaborn: For visualization of data and cluster

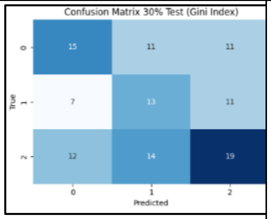
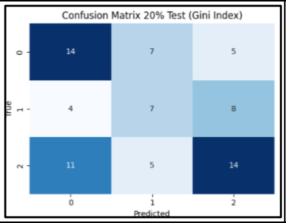
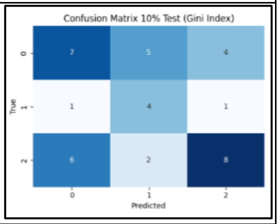
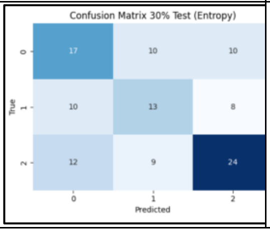
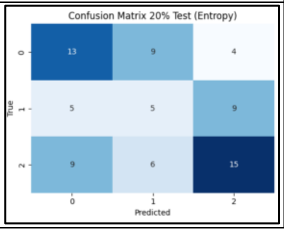
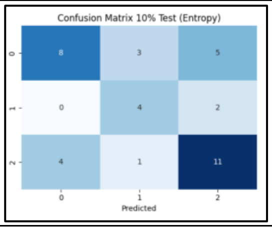
## 6. Evaluation and Comparison:

- Classification:

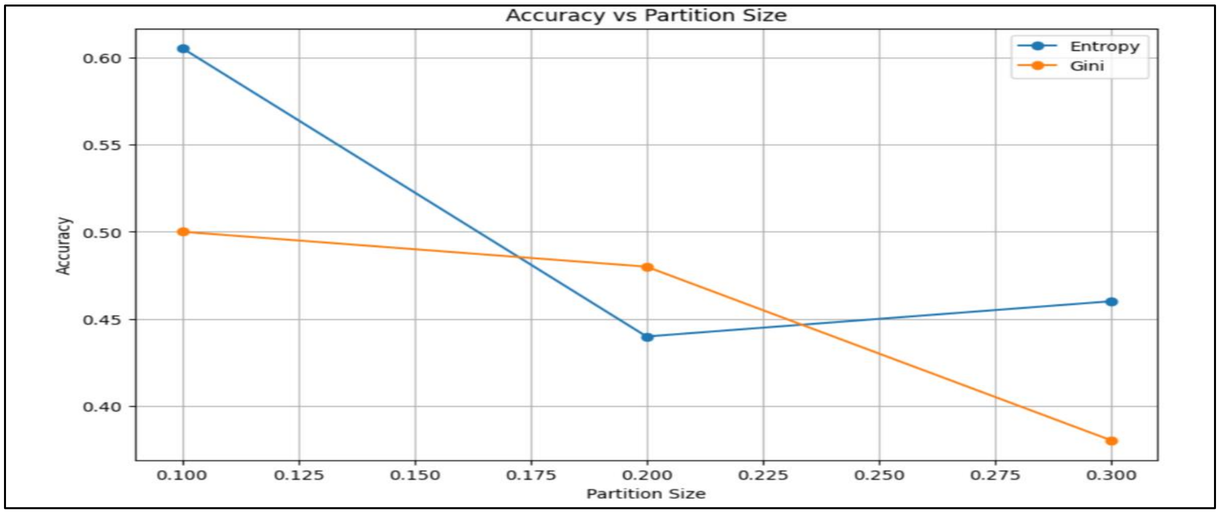
we chose three data partitions. The first partition involved allocating 70% of the data to the training set and 30% to the test set. The second partition involved allocating 80% to the training set and 20% to the validation set. Finally, the third partition involved allocating 90% to the training set and 10% to the test set.

Accuracy reflects the proportion of correctly classified examples out of the total examples presented to the model. In other words, it measures the model's ability to correctly predict the class of a given sample. So, in classfcation we try tow different selection measures (IG and Gini index) . The selection measures help determine the best attribute for splitting the data, which directly impacts the model's accuracy in correctly predicting the class of given samples.

the following table show the accuracy for each size with Gini and IG with Confusion matrixes :

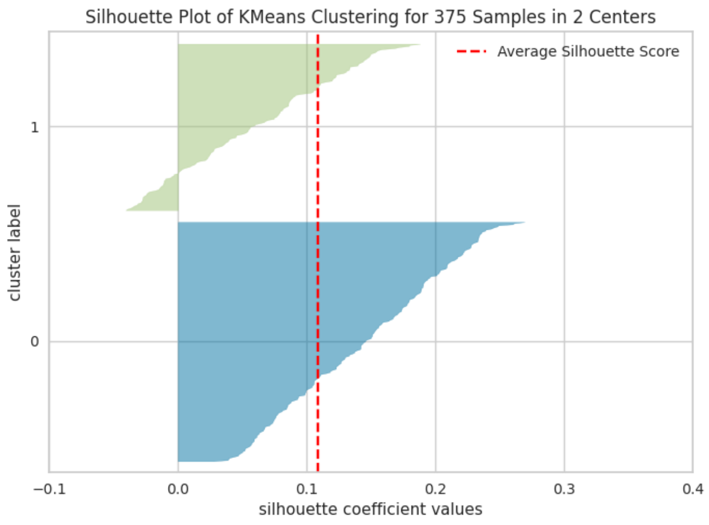
| | selection measure | 70% training set 30% test set | 80% training set 20% test set | 90% training set 10% test set |
|---|---|---|---|---|
| Accuracy | Gini index | 41.6% | 46.7% | 50% |
| | |  |  |  |
| Accuracy | IG | 47.8% | 44% | 60.5% |
| | |  |  |  |

The table shows that accuracy varies depending on the choice of splitting measure and selection metric used. We observe that the Gini Index performs better than Information Gain when the split ratio is 20%, whereas Information Gain is better in all other cases.

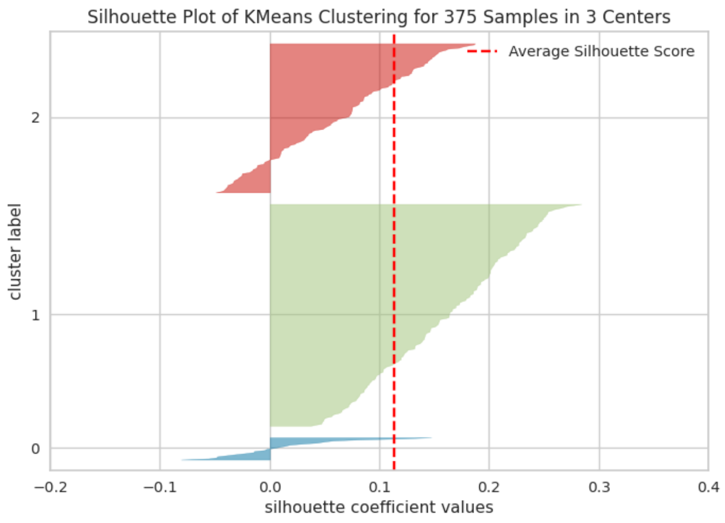


- Clustering:

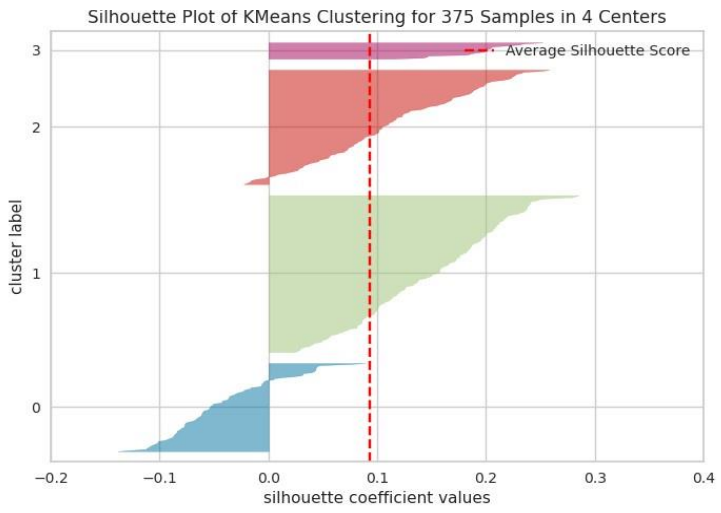
In our clustering analysis, we employed the K-means algorithm with different values of K [2,3,4,5] to determine the ideal number of clusters. By applying (WSS, Average Silhouette Score, Visualization of K-mean) for each K:



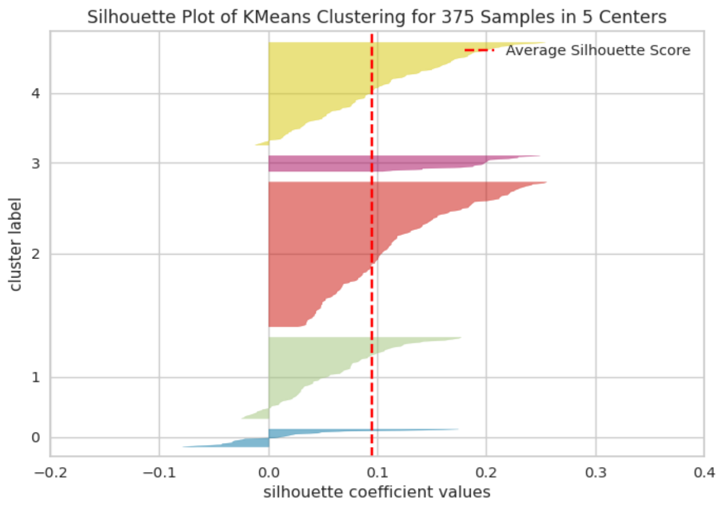
*(Figure 2 : **Number of cluster(K)= 2**)*



*(Figure 2: **Number of cluster(K)= 3**)*



*(Figure 3 : **Number of cluster(K)= 4**)*



*(Figure 4 : **Number of cluster(K)= 5**)*

*(Figure 3 : Number of cluster(K)= visualization4)*

| | K=2 | K=3 | K=4 | K=5 |
|---|---|---|---|---|
| Average Silhouette width | 0.112 | 0.115 | 0.098 | 0.099 |
| total within-cluster sum of square | 5750 | 5350 | 5050 | 4700 |

After examining the plots, we noticed a significant decrease in inertia as the number of clusters increased from 2 to 3, the decrease in inertia became less pronounced. Also, we observed that the average silhouette score was highest when k =3 compared to other values of K [2,4,5].
so, we choose K=3 as it represents a configuration where the data points are most appropriately grouped into distinct clusters, maximizing both cohesion within clusters and separation between clusters. so, we Choose K=3 as the elbow point, indicating that adding more clusters wouldn't significantly reduce the inertia.

## 7. Findings:

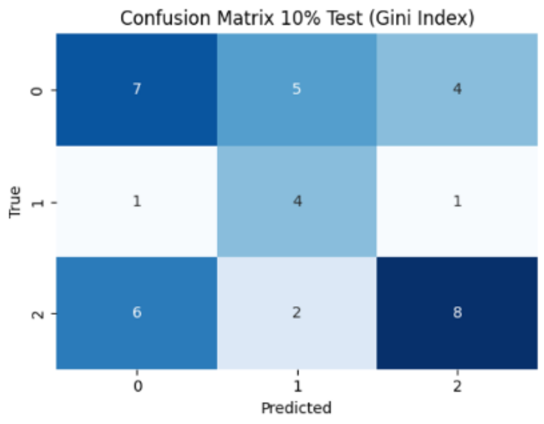
Our project focuses on a dataset about diabetes prevalence and its health indicators. We aim to analyze the data to gain insights and raise awareness about diabetes. By advocating for proactive measures in public health, we hope to contribute to better management and prevention of diabetes, ultimately improving public health outcomes.

To ensure optimal accuracy and efficiency, we implemented various preprocessing techniques aimed at enhancing the quality of our dataset. Utilizing visualization tools like boxplots and histograms, we gained insights into the data distribution, facilitating a better understanding of its characteristics and guiding our preprocessing decisions. Subsequently, we conducted thorough data cleansing by removing null, missing, and outlier values, which could adversely impact the accuracy of our results. Furthermore, we applied data transformation methods such as normalization and discretization to standardize attribute weights and streamline data handling during subsequent mining tasks.

Following preprocessing, we proceeded with data mining tasks, specifically classification and clustering. Employing the decision tree method for classification, we rigorously experimented with varying sizes of training and testing data sets to optimize model construction and evaluation. Through meticulous analysis, we derived conclusive results aimed at enhancing the efficacy and reliability of our classification model The partition sizes used were 10%, 20%, and 30% and we concluded the following results:(Gini index)

1- 90% Training data, 10% Test data, accuracy = 50%
2- 80% Training data, 20% Test data, accuracy = 46.667%
3- 70% Training data, 30% Test data, accuracy = 41.593%

The model that has The highest accuracy achieved with the Gini index was 50% for a partition size of 10% which means that most tuples were correctly classified.

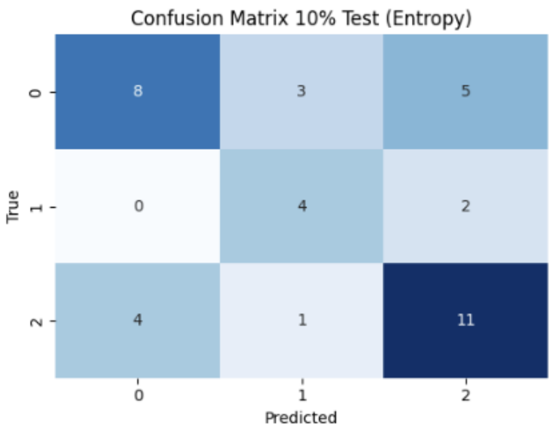


(90% Training data, 10% Test data, accuracy = 50%)

From the plot of the tree, we concluded the following results:(entropy)

1- 90% Training data, 10% Test data, accuracy = 60.5%
2- 80% Training data, 20% Test data, accuracy = 44%
3- 70% Training data, 30% Test data, accuracy = 47.7%

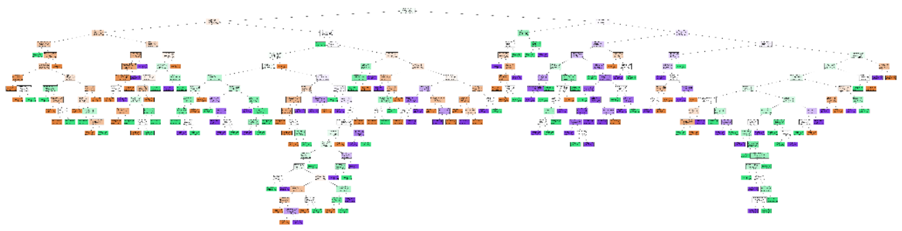
Similarly, the partition sizes used were 10%, 20%, and 30%, The highest accuracy achieved with entropy was 60.5% for a partition size of 10%. Therefore, the best partition size for entropy, based on the highest accuracy, was also 10%.



(90% Training data, 10% Test data, accuracy = 60.5%)

The model employing entropy data, boasting an accuracy rate of 60.5%, stands out as the optimal choice for detecting diabetes based on the provided variable values. With its ability to capture and quantify the uncertainty within the dataset, the entropy-based approach proves effective in discerning patterns and identifying potential indicators of diabetes. Leveraging this model not only enhances diagnostic accuracy but also empowers healthcare professionals with valuable insights for early detection and intervention, thereby contributing to improved patient outcomes and public health initiatives.

Here we choose the 60.5% accuracy “highest accuracy”tree to get the following results:



- The root of the tree is (GenHlth) attribute which has the maximum information gain that split the data into a binary sub tree the first node for tuples HighBp and the second one discrtized_BMI.

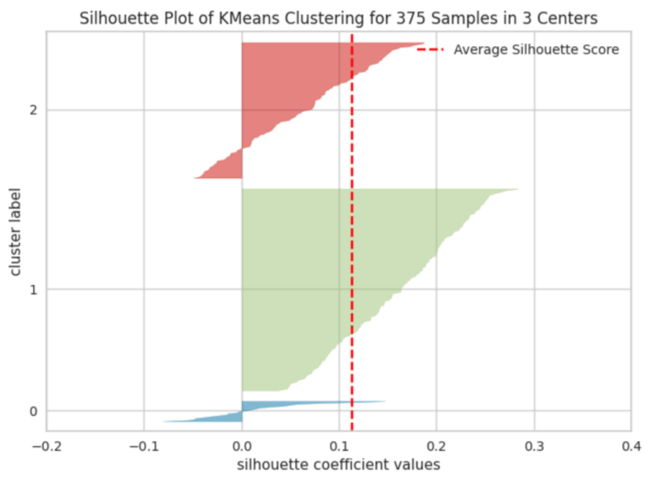
- From the first node, we can notice the (discrtized_BMI) attribute which has the maximum information gain that split the data into a binary sub tree the first node for tuples Age and the second one Education.

- From the second node, we can notice the (High Bp) attribute which has the maximum information gain that split the data into a binary sub tree the first node for tuples AnyHealthcare and the second one age.

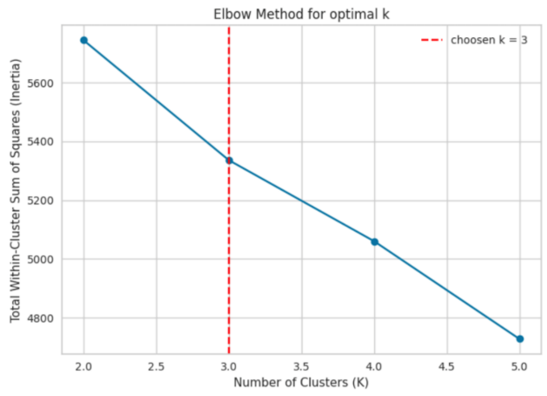
- The most important information for our problem is highest information gain witch is GenHlth.

In our clustering analysis, we employed the K-means algorithm with varying values of K to determine the ideal number of clusters. By computing the average silhouette width for each K, we drew definitive conclusions regarding the optimal clustering configuration, and we concluded the following results:
• Number of cluster(K)= 2, the average silhouette width=0.112
• Number of cluster(K)= 3, the average silhouette width=0.115
• Number of cluster(K)= 4, the average silhouette width=0.098
• Number of cluster(K)= 5, the average silhouette width=0.099

Among the models tested, the 2-Mean model emerges as the optimal choice, boasting the highest average silhouette width. This metric indicates that items within the same cluster exhibit close proximity to one another while maintaining significant distance from objects in other clusters. Furthermore, as evidenced in the evaluation and comparison figures, the 2-Mean clustering model demonstrates minimal overlap between clusters compared to its counterparts.



(2-Mean clusters)



(Optimal number of clusters)

Finally, both models are helpful for predicting whether a person can have a diabetes
and helped us to reach our goal which is helping to have an impact on promoting public health.
but Since our data contains a class label "diabetes" This makes
Supervised Learning models(classification) more accurate and suitable to apply than unsupervised
learning model(clustering), as the expected output is known beforehand this way we makes use of the class
label attribute.

References :

W3school