HEART ATTACK RISK PREDICTION

ABOUT

- This project aims to predict the likelihood of a heart attack using a data set that includes various health and lifestyle factors.
- The prediction is based on the identification of key features that contribute significantly to the assessment of heart attack risk.

FEATURES: The dataset includes the following features:

- 1. **Patient ID:** Unique identifier for each patient
- 2. **Age:** Age of the patient
- 3. **Sex:** Gender of the patient (Male/Female)
- 4. **Cholesterol:** Cholesterol levels of the patient
- 5. **Blood Pressure:** Blood pressure of the patient (systolic/diastolic)
- 6. **Heart Rate:** Heart rate of the patient
- 7. **Diabetes:** Whether the patient has diabetes (Yes/No)
- 8. **Family History:** Family history of heart-related problems (1: Yes, 0: No)
- 9. **Smoking:** Smoking status of the patient (1: Smoker, 0: Non-smoker)
- 10. **Obesity:** Obesity status of the patient (1: Obese, 0: Not obese)
- 11. **Alcohol Consumption:** Level of alcohol consumption by the patient (None/Light/Moderate/Heavy)
- 12. **Exercise Hours Per Week:** Number of exercise hours per week
- 13. **Diet:** Dietary habits of the patient (Healthy/Average/Unhealthy)
- 14. **Previous Heart Problems:** Previous heart problems of the patient (1: Yes, 0: No)
- 15. **Medication Use:** Medication usage by the patient (1: Yes, 0: No)
- 16. **Stress Level:** Stress level reported by the patient (1-10)
- 17. **Sedentary Hours Per Day:** Hours of sedentary activity per day
- 18. **Income:** Income level of the patient
- 19. **BMI:** Body Mass Index (BMI) of the patient
- 20. **Triglycerides:** Triglyceride levels of the patient
- 21. **Physical Activity Days Per Week:** Days of physical activity per week
- 22. **Sleep Hours Per Day:** Hours of sleep per day
- 23. **Country:** Country of the patient
- 24. **Continent:** Continent where the patient resides
- 25. **Hemisphere:** Hemisphere where the patient resides

- 26. **Heart Attack Risk (Outcome):** Presence of heart attack risk (1: Yes, 0: No)

METHODOLOGY:

- The predictive model will focus on selecting the most informative features from the dataset to improve the accuracy of heart attack risk predictions.
- By analyzing and prioritizing key factors, the model aims to provide valuable information to identify individuals at higher risk of having a heart attack.

NUMPY:

- NumPy stands for Numerical Python & also known by the alias array .
- NumPy is a Python library used for working with arrays.
- It also has functions for working in domain of linear algebra, fourier transform, and matrices.
- It is an open source project and you can use it freely.
- It's the universal standard for working with numerical data in Python, At the core of the scientific Python & PyData ecosystems.
- NumPy functions are used to create, manipulate, and analyze NumPy arrays.
- The syntax of NumPy functions generally involves calling the function and passing one or more parameters, such as the shape of the array, the range of values to generate or the type of data to use.
- The list of most commonly used numeric data types in NumPy: int8 , int16 , int32 , int64 - signed integer types with different bit sizes. uint8 , uint16 , uint32 , uint64 - unsigned integer types with different bit sizes.

PANDAS:

- Pandas is a software library written for the Python programming language for functions: analyzing, cleaning, exploring & manipulating data.
- Also used for working with data sets.
- The name "Pandas" has a reference to both "Panel Data", and "Python Data Analysis".
- In particular, it offers data structures and operations for manipulating numerical tables and time series.
- It is free software released under the three-clause BSD license.
- Pandas is built on top of the NumPy package, meaning a lot of the structure of NumPy is used or replicated in Pandas.
- Data in pandas is often used to feed statistical analysis in SciPy, plotting functions from Matplotlib & ML algorithms in Scikit-learn.
- It provides many functions & methods to expedite the data analysis process.
- What makes pandas so common is its functionality, flexibility & simple syntax.

MATPLOTLIB.PYPLOT:

- matplotlib.pyplot is a collection of functions that make matplotlib work like MATLAB.
- Matplotlib is a powerful library which is used to visualize data in form of plots.
- We'll see that, with the help of this library one can create many types of plots like line, bar, histogram, contour, scatter, violin, and many more.
- It is used for creating 2D plots of Arrays.
- Each pyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc.
- The Matplotlib architecture is composed of three main layers: 1.Backend Layer — Handles all the heavy works via communicating to the drawing toolkits in your machine. It is the most complex layer. 2.Artist Layer — Allows full control and fine-tuning of the Matplotlib figure — the top-level container for all plot elements.

SEABORN:

- Seaborn is a Python data visualization library based on matplotlib.
- Python Seaborn library is a widely popular data visualization library that is commonly used for data science and machine learning tasks.
- We'll build it on top of the matplotlib data visualization library and can perform exploratory analysis.
- It provides a high-level interface for drawing attractive and informative statistical graphics.
- For a brief introduction to the ideas behind the library, you can read the introductory notes.
- Provides five different built-in themes named: "darkgrid", "whitegrid", "dark", "white" & "ticks". We can select them both with the style argument of the set_style or set_style function.

What is the difference between matplotlib and seaborn?

- Matplotlib is a library in Python that enables users to generate visualizations like histograms, scatter plots, bar charts, pie charts and much more.
- Seaborn is a visualization library that is built on top of Matplotlib & It provides data visualizations that are typically more aesthetic and statistically sophisticated.

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

IMPORTING & READING THE EXCEL FILE

In [2]:
```
df = r"E:\DATASETS/heart_attack_prediction_dataset.csv"
df = pd.read_csv(df)
df
```

Out[2]:

	Patient ID	Age	Sex	Cholesterol	Blood Pressure	Heart Rate	Diabetes	Family History	Smoking	Obesity	...	Sedentary Hours Per Day	Income	BMI	Tr
0	BMW7812	67	Male	208	158/88	72	0	0	1	0	...	6.615001	261404	31.251233	
1	CZE1114	21	Male	389	165/93	98	1	1	1	1	...	4.963459	285768	27.194973	
2	BNI9906	21	Female	324	174/99	72	1	0	0	0	...	9.463426	235282	28.176571	
3	JLN3497	84	Male	383	163/100	73	1	1	1	0	...	7.648981	125640	36.464704	
4	GFO8847	66	Male	318	91/88	93	1	1	1	1	...	1.514821	160555	21.809144	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
8758	MSV9918	60	Male	121	94/76	61	1	1	1	0	...	10.806373	235420	19.655895	
8759	QSV6764	28	Female	120	157/102	73	1	0	0	1	...	3.833038	217881	23.993866	
8760	XKA5925	47	Male	250	161/75	105	0	1	1	1	...	2.375214	36998	35.406146	
8761	EPE6801	36	Male	178	119/67	60	1	0	1	0	...	0.029104	209943	27.294020	
8762	ZWN9666	25	Female	356	138/67	75	1	1	0	0	...	9.005234	247338	32.914151	

8763 rows × 26 columns

head() function:

- The head() function is used to get the first n rows.

- This function returns the first n rows for the object based on position.
- It is useful for quickly testing if your object has the right type of data in it.
- The first n rows of the caller object.

In [3]:
```
df.head()
```

Out[3]:

	Patient ID	Age	Sex	Cholesterol	Blood Pressure	Heart Rate	Diabetes	Family History	Smoking	Obesity	...	Sedentary Hours Per Day	Income	BMI	Trigly
0	BMW7812	67	Male	208	158/88	72	0	0	1	0	...	6.615001	261404	31.251233	
1	CZE1114	21	Male	389	165/93	98	1	1	1	1	...	4.963459	285768	27.194973	
2	BNI9906	21	Female	324	174/99	72	1	0	0	0	...	9.463426	235282	28.176571	
3	JLN3497	84	Male	383	163/100	73	1	1	1	0	...	7.648981	125640	36.464704	
4	GFO8847	66	Male	318	91/88	93	1	1	1	1	...	1.514821	160555	21.809144	

5 rows × 26 columns

tail() function:

- The tail() function is used to get the last n rows.
- This function returns last n rows from the object based on position.
- It is useful for quickly verifying data, for example, after sorting or appending rows.
- The last n rows of the caller object.

In [4]:
```
df.tail()
```

Out[4]:

	Patient ID	Age	Sex	Cholesterol	Blood Pressure	Heart Rate	Diabetes	Family History	Smoking	Obesity	...	Sedentary Hours Per Day	Income	BMI	Tri
8758	MSV9918	60	Male	121	94/76	61	1	1	1	0	...	10.806373	235420	19.655895	
8759	QSV6764	28	Female	120	157/102	73	1	0	0	1	...	3.833038	217881	23.993866	
8760	XKA5925	47	Male	250	161/75	105	0	1	1	1	...	2.375214	36998	35.406146	
8761	EPE6801	36	Male	178	119/67	60	1	0	1	0	...	0.029104	209943	27.294020	
8762	ZWN9666	25	Female	356	138/67	75	1	1	0	0	...	9.005234	247338	32.914151	

5 rows × 26 columns

SHAPE OF THE DATAFRAME:

- The shape attribute of pandas.
- DataFrame stores the number of rows and columns as a tuple (number of rows, number of columns).
- The shape function in Python is used to find the dimensions of data structures, such as NumPy arrays and Pandas DataFrames.

In [5]:
```
df.shape
```

Out[5]:
```
(8763, 26)
```

describe() function:

- Pandas describe() is used to view some basic statistical details like percentile, mean, std etc. of a data frame or a series of numeric values.
- When this method is applied to a series of string, it returns a different output which is shown below.

include = 'all' function:

- It provides as an option, the result will include a union of attributes of each type.
- The includes and excludes parameters, can be used to limit which columns in a DataFrame are analyzed for the output.
- The parameters are ignored when analyzing a Series & describes a numeric Series.

In [6]:
```
df.describe(include = "all")
```

Out[6]:

	Patient ID	Age	Sex	Cholesterol	Blood Pressure	Heart Rate	Diabetes	Family History	Smoking	Obesity	...	S H
count	8763	8763.000000	8763	8763.000000	8763	8763.000000	8763.000000	8763.000000	8763.000000	8763.000000	...	876
unique	8763	NaN	2	NaN	3915	NaN	NaN	NaN	NaN	NaN	...	
top	BMW7812	NaN	Male	NaN	146/94	NaN	NaN	NaN	NaN	NaN	...	
freq	1	NaN	6111	NaN	8	NaN	NaN	NaN	NaN	NaN	...	
mean	NaN	53.707977	NaN	259.877211	NaN	75.021682	0.652288	0.492982	0.896839	0.501426	...	
std	NaN	21.249509	NaN	80.863276	NaN	20.550948	0.476271	0.499979	0.304186	0.500026	...	
min	NaN	18.000000	NaN	120.000000	NaN	40.000000	0.000000	0.000000	0.000000	0.000000	...	
25%	NaN	35.000000	NaN	192.000000	NaN	57.000000	0.000000	0.000000	1.000000	0.000000	...	
50%	NaN	54.000000	NaN	259.000000	NaN	75.000000	1.000000	0.000000	1.000000	1.000000	...	
75%	NaN	72.000000	NaN	330.000000	NaN	93.000000	1.000000	1.000000	1.000000	1.000000	...	
max	NaN	90.000000	NaN	400.000000	NaN	110.000000	1.000000	1.000000	1.000000	1.000000	...	1

11 rows × 26 columns

df.columns function:

- The column labels/names of the DataFrame.
- Pandas DataFrame is a two-dimensional size-mutable, potentially heterogeneous tabular data structure with labeled axes (rows and columns).
- Arithmetic operations align on both row and column labels.
- It can be thought of as a dict-like container for Series objects.

- This is the primary data structure of the Pandas.

In [7]:
```
df.columns
```

Out[7]:
```
Index(['Patient ID', 'Age', 'Sex', 'Cholesterol', 'Blood Pressure',
       'Heart Rate', 'Diabetes', 'Family History', 'Smoking', 'Obesity',
       'Alcohol Consumption', 'Exercise Hours Per Week', 'Diet',
       'Previous Heart Problems', 'Medication Use', 'Stress Level',
       'Sedentary Hours Per Day', 'Income', 'BMI', 'Triglycerides',
       'Physical Activity Days Per Week', 'Sleep Hours Per Day', 'Country',
       'Continent', 'Hemisphere', 'Heart Attack Risk'],
      dtype='object')
```

info() function:

- The info() function is used to print a concise summary of a DataFrame.
- This method prints information about a DataFrame including the index dtype and column dtypes, non-null values and memory usage.

In [8]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8763 entries, 0 to 8762
Data columns (total 26 columns):
 #   Column                           Non-Null Count  Dtype  
---  ------                           --------------  -----  
 0   Patient ID                       8763 non-null   object 
 1   Age                              8763 non-null   int64  
 2   Sex                              8763 non-null   object 
 3   Cholesterol                      8763 non-null   int64  
 4   Blood Pressure                   8763 non-null   object 
 5   Heart Rate                       8763 non-null   int64  
 6   Diabetes                         8763 non-null   int64  
 7   Family History                   8763 non-null   int64  
 8   Smoking                          8763 non-null   int64  
 9   Obesity                          8763 non-null   int64  
 10  Alcohol Consumption              8763 non-null   int64  
 11  Exercise Hours Per Week          8763 non-null   float64
 12  Diet                             8763 non-null   object 
 13  Previous Heart Problems          8763 non-null   int64  
 14  Medication Use                   8763 non-null   int64  
 15  Stress Level                     8763 non-null   int64  
 16  Sedentary Hours Per Day          8763 non-null   float64
 17  Income                           8763 non-null   int64  
 18  BMI                              8763 non-null   float64
 19  Triglycerides                    8763 non-null   int64  
 20  Physical Activity Days Per Week  8763 non-null   int64  
 21  Sleep Hours Per Day              8763 non-null   int64  
 22  Country                          8763 non-null   object 
 23  Continent                        8763 non-null   object 
 24  Hemisphere                       8763 non-null   object 
 25  Heart Attack Risk                8763 non-null   int64  
dtypes: float64(3), int64(16), object(7)
memory usage: 1.7+ MB
```

drop() function:

- The drop() function is used to drop specified labels from rows or columns.
- Remove rows or columns by specifying label names and corresponding axis, or by specifying directly index or column names.
- When using a multi-index, labels on different levels can be removed by specifying the level.

inplace = True function:

- When **inplace = True,** the data is modified in place, which means it will return nothing and the dataframe is now updated.
- When **inplace = False: which is the default,** then the operation is performed & it returns a copy of the object. You then need to save it to something.

Drop the Duplicate values - if any

In [9]:
```
df.drop_duplicates(inplace=True)
df.shape
```

Out[9]:
```
(8763, 26)
```

In [10]:
```
columns_to_drop = ['Hemisphere', 'Patient ID', 'Income', 'Country', 'Continent']
df.drop(columns_to_drop, axis=1, inplace=True)
df
```

Out[10]:

	Age	Sex	Cholesterol	Blood Pressure	Heart Rate	Diabetes	Family History	Smoking	Obesity	Alcohol Consumption	...	Diet	Previous Heart Problems	Medicati U
0	67	Male	208	158/88	72	0	0	1	0	0	...	Average	0	
1	21	Male	389	165/93	98	1	1	1	1	1	...	Unhealthy	1	
2	21	Female	324	174/99	72	1	0	0	0	0	...	Healthy	1	
3	84	Male	383	163/100	73	1	1	1	0	1	...	Average	1	
4	66	Male	318	91/88	93	1	1	1	1	0	...	Unhealthy	1	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
8758	60	Male	121	94/76	61	1	1	1	0	1	...	Healthy	1	
8759	28	Female	120	157/102	73	1	0	0	1	0	...	Healthy	0	
8760	47	Male	250	161/75	105	0	1	1	1	1	...	Average	1	
8761	36	Male	178	119/67	60	1	0	1	0	0	...	Unhealthy	1	
8762	25	Female	356	138/67	75	1	1	0	0	1	...	Healthy	0	

8763 rows × 21 columns

Unique function:

- The unique function in pandas is used to find the unique values from a series.
- A series is a single column of a data frame.
- We can use the unique function on any possible set of elements in Python.
- It can be used on a series of strings, integers, tuples, or mixed elements.
- Also finds the unique elements of an array and returns these unique elements as a sorted array.

isnull() function:

- It detect missing values in the given series object.

- It return a boolean same-sized object indicating if the values are NA.
- Missing values gets mapped to True and non-missing value gets mapped to False.

What is the use of isnull () method in pandas?

- The isnull() function is used to detect missing values for an array-like object.
- This function takes a scalar or array-like object and indicates whether values are missing (NaN in numeric arrays, None or NaN in object arrays, NaT in datetimelike).
- Object to check for null or missing values.

df.isnull().sum() function:

- returns the number of missing values in the data set.
- A simple way to deal with data containing missing values is to skip rows with missing values in the dataset.

isna() function:

- The isna() method returns a DataFrame object where all the values are replaced with a Boolean value True for NA (not-a -number) values & otherwise False.
- isna() method checks whether the objects of a Dataframe or a Series contain missing or null values (NA, NaN) & returns a new object with the same shape as the original but with boolean values True or False as the elements.
- True indicates the presence of null or missing values and False indicates otherwise.

1. Lets start organizing the age column

In [11]:
```
df['Age'].unique()
```

Out[11]:
```
array([67, 21, 84, 66, 54, 90, 20, 43, 73, 71, 77, 60, 88, 69, 38, 50, 45,
       36, 48, 40, 79, 63, 27, 25, 86, 42, 52, 29, 30, 47, 44, 33, 51, 70,
       85, 31, 56, 24, 74, 72, 55, 26, 53, 46, 57, 22, 35, 39, 80, 65, 83,
       82, 28, 19, 75, 18, 34, 37, 89, 32, 49, 23, 59, 62, 64, 61, 76, 41,
       87, 81, 58, 78, 68], dtype=int64)
```

In [12]:
```
df['Age'].isnull().sum()
```

Out[12]:
```
0
```

Above we can see that all data of 'Age' column is clear, no null, and all of them are in the type of integer

In [13]:
```
df.head(10)
```

Out[13]:

	Age	Sex	Cholesterol	Blood Pressure	Heart Rate	Diabetes	Family History	Smoking	Obesity	Alcohol Consumption	...	Diet	Previous Heart Problems	Medication Use
0	67	Male	208	158/88	72	0	0	1	0	0	...	Average	0	0
1	21	Male	389	165/93	98	1	1	1	1	1	...	Unhealthy	1	0
2	21	Female	324	174/99	72	1	0	0	0	0	...	Healthy	1	1
3	84	Male	383	163/100	73	1	1	1	0	1	...	Average	1	0
4	66	Male	318	91/88	93	1	1	1	1	0	...	Unhealthy	1	0
5	54	Female	297	172/86	48	1	1	1	0	1	...	Unhealthy	1	1
6	90	Male	358	102/73	84	0	0	1	0	1	...	Healthy	0	0
7	84	Male	220	131/68	107	0	0	1	1	1	...	Average	0	1
8	20	Male	145	144/105	68	1	0	1	1	0	...	Average	0	0
9	43	Female	248	160/70	55	0	1	1	1	1	...	Unhealthy	0	0

10 rows × 21 columns

2. Organizing the 'Sex' column

In [14]:
```
df['Sex'].unique()
```

Out[14]:
```
array(['Male', 'Female'], dtype=object)
```

With the help of the One-hot-encoding, 'Sex' column is converted to is_male column

- 'Sex' --> 'is_male' (if male: 1, otherwise: 0)

In [15]:
```
df = pd.get_dummies(df, columns=['Sex'], drop_first=True)
df.rename(columns={'Sex_Male': 'is_male'},inplace=True)
```

Drop first drops the original 'Sex' column. Now it is True if patient is Male otherwise it is False.

In [16]:
```
df['is_male'] = df['is_male'].astype(int)
```

changing type to integer True -> 1 , False -> 0 so it won't be any problem when we use this column in model.

In [17]:
```
df.head(10)
```

Out[17]:

	Age	Cholesterol	Blood Pressure	Heart Rate	Diabetes	Family History	Smoking	Obesity	Alcohol Consumption	Exercise Hours Per Week	...	Previous Heart Problems	Medication Use	Stress Level
0	67	208	158/88	72	0	0	1	0	0	4.168189	...	0	0	9
1	21	389	165/93	98	1	1	1	1	1	1.813242	...	1	0	1
2	21	324	174/99	72	1	0	0	0	0	2.078353	...	1	1	9
3	84	383	163/100	73	1	1	1	0	1	9.828130	...	1	0	9
4	66	318	91/88	93	1	1	1	1	0	5.804299	...	1	0	6
5	54	297	172/86	48	1	1	1	0	1	0.625008	...	1	1	2
6	90	358	102/73	84	0	0	1	0	1	4.098177	...	0	0	7
7	84	220	131/68	107	0	0	1	1	1	3.427929	...	0	1	4
8	20	145	144/105	68	1	0	1	1	0	16.868302	...	0	0	5
9	43	248	160/70	55	0	1	1	1	1	0.194515	...	0	0	4

10 rows × 21 columns

3.Orginizing the Cholesterol Column

```
In [18]: df['Cholesterol'].unique()
```

```
Out[18]: array([208, 389, 324, 383, 318, 297, 358, 220, 145, 248, 373, 374, 228,
                259, 122, 379, 166, 303, 340, 294, 359, 202, 133, 159, 271, 273,
                328, 154, 135, 197, 321, 375, 360, 263, 201, 347, 129, 229, 251,
                121, 190, 185, 279, 336, 192, 180, 203, 368, 222, 243, 218, 120,
                285, 377, 369, 311, 139, 266, 153, 339, 329, 333, 398, 124, 183,
                163, 362, 390, 200, 396, 255, 209, 247, 250, 227, 246, 223, 330,
                195, 194, 178, 155, 240, 237, 216, 276, 224, 326, 198, 301, 314,
                304, 334, 213, 254, 230, 316, 277, 388, 206, 384, 205, 261, 308,
                338, 382, 291, 168, 171, 378, 253, 245, 226, 281, 123, 173, 231,
                234, 268, 306, 186, 293, 161, 380, 239, 149, 320, 219, 335, 265,
                126, 307, 270, 225, 193, 148, 296, 136, 364, 353, 252, 232, 387,
                299, 357, 214, 370, 345, 351, 344, 152, 150, 131, 272, 302, 337,
                170, 356, 274, 188, 125, 138, 376, 181, 184, 275, 394, 128, 217,
                399, 283, 289, 284, 327, 262, 212, 350, 385, 162, 141, 361, 244,
                295, 287, 144, 354, 363, 352, 140, 196, 172, 319, 325, 331, 392,
                147, 187, 346, 286, 151, 300, 165, 343, 366, 317, 386, 158, 157,
                242, 241, 365, 257, 348, 175, 298, 269, 267, 397, 310, 341, 204,
                127, 290, 280, 132, 322, 179, 199, 143, 312, 288, 395, 189, 156,
                238, 381, 391, 355, 210, 400, 260, 235, 167, 256, 249, 207, 130,
                134, 137, 305, 236, 315, 292, 323, 146, 258, 332, 372, 142, 309,
                177, 367, 371, 211, 282, 342, 264, 176, 160, 233, 313, 164, 349,
                221, 191, 174, 393, 278, 215, 169, 182], dtype=int64)
```

Looks like 'Cholesterol' column is already organized, and we don't need to change anything

4. Blood Pressure

```
In [19]: df['Blood Pressure'].unique()
```

```
Out[19]: array(['158/88', '165/93', '174/99', ..., '137/94', '94/76', '119/67'],
               dtype=object)
```

```
In [20]: df['Blood Pressure'].isnull().sum()
```

```
Out[20]: 0
```

Observation:

- We can see that there are not any NULL values for the Blood Pressure.
- However, it is in number/number format which we cannot use in our modelling.
- We can separate the columns. Originally: (systolic/diastolic), now I will create separately two columns systolic_pressure & diastolic_pressure

In [21]:
```python
def handle_blood_pressure_systolic(value):
    value = str(value)
    value = value.split('/')
    return int(value[0])

def handle_blood_pressure_diastolic(value):
    value = str(value)
    value = value.split('/')
    return int(value[1])

df['systolic_pressure'] = df['Blood Pressure'].apply(handle_blood_pressure_systolic)
df['diastolic_pressure'] = df['Blood Pressure'].apply(handle_blood_pressure_diastolic)

df.drop(columns='Blood Pressure', axis=1, inplace=True)
df.head(10)
```

Out[21]:

	Age	Cholesterol	Heart Rate	Diabetes	Family History	Smoking	Obesity	Alcohol Consumption	Exercise Hours Per Week	Diet	...	Stress Level	Sedentary Hours Per Day	BMI
0	67	208	72	0	0	1	0	0	4.168189	Average	...	9	6.615001	31.251233
1	21	389	98	1	1	1	1	1	1.813242	Unhealthy	...	1	4.963459	27.194973
2	21	324	72	1	0	0	0	0	2.078353	Healthy	...	9	9.463426	28.176571
3	84	383	73	1	1	1	0	1	9.828130	Average	...	9	7.648981	36.464704
4	66	318	93	1	1	1	1	0	5.804299	Unhealthy	...	6	1.514821	21.809144
5	54	297	48	1	1	1	0	1	0.625008	Unhealthy	...	2	7.798752	20.146840
6	90	358	84	0	0	1	0	1	4.098177	Healthy	...	7	0.627356	28.885811
7	84	220	107	0	0	1	1	1	3.427929	Average	...	4	10.543780	22.221862
8	20	145	68	1	0	1	1	0	16.868302	Average	...	5	11.348787	35.809901
9	43	248	55	0	1	1	1	1	0.194515	Unhealthy	...	4	4.055115	22.558917

10 rows × 22 columns

5. Organizing the Heart Rate Column

```
In [22]: df['Heart Rate'].unique()
```

```
Out[22]: array([ 72,  98,  73,  93,  48,  84, 107,  68,  55,  97,  70,  85, 102,
                 40,  56, 104,  71,  69,  66,  81,  52, 105,  96,  74,  49,  45,
                 50,  46,  44, 106,  83,  86,  65, 101,  51,  43,  79,  90,  94,
                 78,  92,  54, 109,  61,  64,  82, 110,  42,  63,  41, 100,  76,
                 75,  58,  53,  60,  77,  47,  59,  57,  87,  67,  88,  99,  80,
                 95, 108,  89,  62, 103,  91], dtype=int64)
```

```
In [23]: df['Heart Rate'].isnull().sum()
```

```
Out[23]: 0
```

```
In [24]: df['Heart Rate'].isna().sum()
```

```
Out[24]: 0
```

Looks like Heart Rate is already organized and cleaned. All values are in integer format, and the column does not contain any NaN value.

In [25]:
```
df.head(10)
```

Out[25]:

	Age	Cholesterol	Heart Rate	Diabetes	Family History	Smoking	Obesity	Alcohol Consumption	Exercise Hours Per Week	Diet	...	Stress Level	Sedentary Hours Per Day	BMI
0	67	208	72	0	0	1	0	0	4.168189	Average	...	9	6.615001	31.251233
1	21	389	98	1	1	1	1	1	1.813242	Unhealthy	...	1	4.963459	27.194973
2	21	324	72	1	0	0	0	0	2.078353	Healthy	...	9	9.463426	28.176571
3	84	383	73	1	1	1	0	1	9.828130	Average	...	9	7.648981	36.464704
4	66	318	93	1	1	1	1	0	5.804299	Unhealthy	...	6	1.514821	21.809144
5	54	297	48	1	1	1	0	1	0.625008	Unhealthy	...	2	7.798752	20.146840
6	90	358	84	0	0	1	0	1	4.098177	Healthy	...	7	0.627356	28.885811
7	84	220	107	0	0	1	1	1	3.427929	Average	...	4	10.543780	22.221862
8	20	145	68	1	0	1	1	0	16.868302	Average	...	5	11.348787	35.809901
9	43	248	55	0	1	1	1	1	0.194515	Unhealthy	...	4	4.055115	22.558917

10 rows × 22 columns

6. Organizing the 'Diabetes' Feature

In [26]:
```
df['Diabetes'].unique()
```

Out[26]:
```
array([0, 1], dtype=int64)
```

In [27]:
```
df['Diabetes'].isnull().sum()
```

Out[27]:
```
0
```

In [28]:
```
df['Diabetes'].isna().sum()
```

Out[28]:
```
0
```

In [29]:
```
df.head(10)
```

Out[29]:

	Age	Cholesterol	Heart Rate	Diabetes	Family History	Smoking	Obesity	Alcohol Consumption	Exercise Hours Per Week	Diet	...	Stress Level	Sedentary Hours Per Day	BMI
0	67	208	72	0	0	1	0	0	4.168189	Average	...	9	6.615001	31.251233
1	21	389	98	1	1	1	1	1	1.813242	Unhealthy	...	1	4.963459	27.194973
2	21	324	72	1	0	0	0	0	2.078353	Healthy	...	9	9.463426	28.176571
3	84	383	73	1	1	1	0	1	9.828130	Average	...	9	7.648981	36.464704
4	66	318	93	1	1	1	1	0	5.804299	Unhealthy	...	6	1.514821	21.809144
5	54	297	48	1	1	1	0	1	0.625008	Unhealthy	...	2	7.798752	20.146840
6	90	358	84	0	0	1	0	1	4.098177	Healthy	...	7	0.627356	28.885811
7	84	220	107	0	0	1	1	1	3.427929	Average	...	4	10.543780	22.221862
8	20	145	68	1	0	1	1	0	16.868302	Average	...	5	11.348787	35.809901
9	43	248	55	0	1	1	1	1	0.194515	Unhealthy	...	4	4.055115	22.558917

10 rows × 22 columns

Diabetes column is already organized.

7. Family History

In [30]:
```
df['Family History'].unique()
```

Out[30]:
```
array([0, 1], dtype=int64)
```

In [31]:
```
df['Family History'].isnull().sum()
```

Out[31]:
```
0
```

```
In [32]: df['Family History'].isna().sum()
Out[32]: 0
```

Family history is already a clean feature

8. Smoking

```
In [33]: df['Smoking'].unique()
Out[33]: array([1, 0], dtype=int64)

In [34]: df['Smoking'].isnull().sum()
Out[34]: 0

In [35]: df['Smoking'].isna().sum()
Out[35]: 0
```

Smoking is already a clean feature

9. Obesity

```
In [36]: df['Obesity'].unique()
Out[36]: array([0, 1], dtype=int64)

In [37]: df['Obesity'].isnull().sum()
Out[37]: 0

In [38]: df['Obesity'].isna().sum()
Out[38]: 0
```

Obesity is already a clean feature

10. Alcohol Consumption

In [39]:
```
df['Alcohol Consumption'].unique()
```
Out[39]:
```
array([0, 1], dtype=int64)
```

In [40]:
```
df['Alcohol Consumption'].isnull().sum()
```
Out[40]:
```
0
```

In [41]:
```
df['Alcohol Consumption'].isna().sum()
```
Out[41]:
```
0
```

In [42]:
```
df.head(10)
```
Out[42]:

	Age	Cholesterol	Heart Rate	Diabetes	Family History	Smoking	Obesity	Alcohol Consumption	Exercise Hours Per Week	Diet	...	Stress Level	Sedentary Hours Per Day	BMI
0	67	208	72	0	0	1	0	0	4.168189	Average	...	9	6.615001	31.251233
1	21	389	98	1	1	1	1	1	1.813242	Unhealthy	...	1	4.963459	27.194973
2	21	324	72	1	0	0	0	0	2.078353	Healthy	...	9	9.463426	28.176571
3	84	383	73	1	1	1	0	1	9.828130	Average	...	9	7.648981	36.464704
4	66	318	93	1	1	1	1	0	5.804299	Unhealthy	...	6	1.514821	21.809144
5	54	297	48	1	1	1	0	1	0.625008	Unhealthy	...	2	7.798752	20.146840
6	90	358	84	0	0	1	0	1	4.098177	Healthy	...	7	0.627356	28.885811
7	84	220	107	0	0	1	1	1	3.427929	Average	...	4	10.543780	22.221862
8	20	145	68	1	0	1	1	0	16.868302	Average	...	5	11.348787	35.809901
9	43	248	55	0	1	1	1	1	0.194515	Unhealthy	...	4	4.055115	22.558917

10 rows × 22 columns

Alcohol consumption is already a clean feature, no null values are found and in integer type.

11. Exercise Hours Per Week

```
In [43]: df['Exercise Hours Per Week'].unique()
```

```
Out[43]: array([ 4.16818884,  1.81324162,  2.07835299, ...,  3.14843791,
                 3.78994983, 18.08174797])
```

```
In [44]: df['Exercise Hours Per Week'].isnull().sum()
```

```
Out[44]: 0
```

```
In [45]: df['Exercise Hours Per Week'].isna().sum()
```

```
Out[45]: 0
```

Exercise Hours Per Week column is also clean

12. Diet Column

```
In [46]: df['Diet'].unique()
```

```
Out[46]: array(['Average', 'Unhealthy', 'Healthy'], dtype=object)
```

With the changes below, now Unhealthy is represented as 0, Average is represented as 1 and Healthy is represented as 2

In [47]:
```
def handle_diet(value):
    value = str(value)

    if value == 'Unhealthy':
        return 0
    elif value == 'Average':
        return 1
    elif value == 'Healthy':
        return 2
    else:
        return np.nan

df['Diet'] = df['Diet'].apply(handle_diet)
df['Diet']
```

Out[47]:
```
0       1
1       0
2       2
3       1
4       0
       ..
8758    2
8759    2
8760    1
8761    0
8762    2
Name: Diet, Length: 8763, dtype: int64
```

The type of the Diet is now integer instead of Object.

In [48]:
```
df['Diet'].unique()
```

Out[48]:
```
array([1, 0, 2], dtype=int64)
```

In [49]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 8763 entries, 0 to 8762
Data columns (total 22 columns):
 #   Column                           Non-Null Count  Dtype  
---  ------                           --------------  -----  
 0   Age                              8763 non-null   int64  
 1   Cholesterol                      8763 non-null   int64  
 2   Heart Rate                       8763 non-null   int64  
 3   Diabetes                         8763 non-null   int64  
 4   Family History                   8763 non-null   int64  
 5   Smoking                          8763 non-null   int64  
 6   Obesity                          8763 non-null   int64  
 7   Alcohol Consumption              8763 non-null   int64  
 8   Exercise Hours Per Week          8763 non-null   float64
 9   Diet                             8763 non-null   int64  
 10  Previous Heart Problems          8763 non-null   int64  
 11  Medication Use                   8763 non-null   int64  
 12  Stress Level                     8763 non-null   int64  
 13  Sedentary Hours Per Day          8763 non-null   float64
 14  BMI                              8763 non-null   float64
 15  Triglycerides                    8763 non-null   int64  
 16  Physical Activity Days Per Week  8763 non-null   int64  
 17  Sleep Hours Per Day              8763 non-null   int64  
 18  Heart Attack Risk                8763 non-null   int64  
 19  is_male                          8763 non-null   int32  
 20  systolic_pressure                8763 non-null   int64  
 21  diastolic_pressure               8763 non-null   int64  
dtypes: float64(3), int32(1), int64(18)
memory usage: 1.5 MB
```

13. Previous Heart Problems

In [50]:
```
df['Previous Heart Problems'].unique()
```

Out[50]:
```
array([0, 1], dtype=int64)
```

```
In [51]: df['Previous Heart Problems'].isnull().sum()

Out[51]: 0

In [52]: df['Previous Heart Problems'].isna().sum()

Out[52]: 0
```

Previous heart problems column is already cleaned, no null values and all values are in integer format

14. Medication Use

```
In [53]: df['Medication Use'].unique()

Out[53]: array([0, 1], dtype=int64)

In [54]: df['Medication Use'].isnull().sum()

Out[54]: 0

In [55]: df['Medication Use'].isna().sum()

Out[55]: 0
```

Medication Use column is already cleaned, no null values and all values are in integer format

15. Stress Level

```
In [56]: df['Stress Level'].unique()

Out[56]: array([ 9,  1,  6,  2,  7,  4,  5,  8, 10,  3], dtype=int64)

In [57]: df['Stress Level'].isnull().sum()

Out[57]: 0
```

Stress Level Column is already cleaned, no null values and all values are in integer format.

16. Sedentary Hours Per Day (Hours that are spent without moving)

```
In [58]: df['Sedentary Hours Per Day'].unique()

Out[58]: array([6.61500145, 4.96345884, 9.46342584, ..., 2.37521373, 0.02910426,
                9.00523438])

In [59]: df['Sedentary Hours Per Day'].isnull().sum()

Out[59]: 0

In [60]: df['Sedentary Hours Per Day'].isna().sum()

Out[60]: 0
```

Sedentary Hours Per Day column is already cleaned, no null values and all values are in float format

17. Cleaning BMI Feature

```
In [61]: df['BMI'].unique()

Out[61]: array([31.25123273, 27.19497335, 28.17657068, ..., 35.40614616,
                27.29402009, 32.91415086])

In [62]: df['BMI'].isnull().sum()

Out[62]: 0

In [63]: df['BMI'].isna().sum()

Out[63]: 0
```

BMI feature is already cleaned

18. Triglycerides

```
In [64]: df['Triglycerides'].unique()
```

```
Out[64]: array([286, 235, 587, 378, 231, 795, 284, 370, 790, 232, 469, 523, 590,
               506, 635, 773,  68, 402, 517, 247, 747, 360, 358, 526, 605, 667,
               316, 551, 482, 718, 297, 661, 558, 209, 586, 743, 411, 785, 697,
               519, 595, 452, 158, 679, 675, 792, 584, 366, 741, 474,  92, 410,
               398, 493, 614, 682, 106, 216, 408, 628, 481,  67,  82, 305, 164,
               211, 511, 766, 547, 327, 367, 681, 131,  42, 692, 664, 543, 689,
               569, 458, 683, 779, 136, 643, 653,  55, 275, 314, 760, 404, 576,
               690, 648, 385, 255, 468, 784, 509, 205, 109, 530, 654, 331, 485,
               250, 113, 377, 180, 229, 602, 285, 471, 554, 344, 416, 445, 709,
               426, 528, 388, 441, 306, 749, 347, 341, 451, 356, 336, 455, 223,
               262, 239, 555, 363, 489, 788, 121, 553, 617, 174, 167, 563, 665,
                65, 657, 237, 141, 767, 292, 214, 221, 447, 634, 460, 711,  97,
               267, 695, 717, 383, 332, 449, 701, 524, 549,  31, 276, 744, 128,
                52, 394,  54, 739, 407, 751, 436, 473, 218, 129, 579, 492, 696,
               202, 197, 521, 325,  35, 123, 694, 434, 248, 348, 750, 431, 714,
               649, 668, 401, 610, 244, 691,  88, 532, 777, 420, 350, 652, 413,
               754, 753, 457, 122, 312, 778, 676, 775, 183, 601, 317, 592, 191,
                83,  32, 453, 423, 234, 650, 565, 798, 769, 412,  63, 198,  93,
               764, 737,  94, 298, 288, 735, 190, 281, 146, 574, 359, 155, 719,
               466, 273, 515, 187, 544, 103, 132, 118, 115,  85,  38, 117, 362,
               133, 498, 645, 339, 787, 733, 663, 291, 502,  78,  81, 257, 624,
                91, 374, 270, 797, 446, 464, 450, 722, 556, 184, 428, 796, 656,
               134, 196, 623, 522, 376, 730, 463,  99, 593,  47, 148, 302,  57,
               280, 389, 629, 294, 186, 700, 774, 181, 375, 467, 603, 616, 380,
               495, 698, 318, 207, 780,  51,  84, 425, 310, 126,  56, 472, 669,
               688, 655,  39, 333, 501, 479, 540, 433, 179, 490, 204, 644, 525,
               546, 486, 320, 319,  58, 591, 165, 732, 195, 478, 461, 631, 301,
                50, 315, 194, 199, 160, 149, 527, 406, 161, 125, 200, 277, 308,
                69, 427, 236,  77, 500, 269,  79, 168, 575, 606, 355, 636,  64,
               251, 245, 228, 287, 800, 483, 791, 260, 604, 536, 559, 124, 254,
               159,  73, 542, 390, 755,  60,  61, 491,  40, 437, 215, 440, 379,
               789, 266, 505, 243, 783, 403, 637, 156, 729, 438, 507, 725, 562,
               324,  87, 253, 626, 541, 364, 456,  30, 182, 621, 494, 776, 442,
               429, 684, 219,  70,  98, 166,  95, 135, 646, 337, 226, 710, 608,
               208, 724, 704, 512, 206, 224, 622, 598, 465, 119, 293, 630, 386,
               513,  45, 578, 261, 217, 715, 282, 391, 580, 192, 399, 249, 396,
               278, 448, 782, 419, 503, 220,  49, 304, 157, 150, 545, 627, 582,
               178, 263,  33, 299, 303,  66, 763, 256, 139, 651, 756, 372, 345,
                48,  46, 421,  43, 771, 210, 781,  41, 508, 353, 566, 726, 736,
               326, 759, 477, 369, 188, 104, 329, 309, 384, 599, 415, 770, 571,
               552, 145, 632, 373,  71, 550, 583, 322, 475, 357, 673, 454, 757,
               201, 100, 274, 258, 613, 233, 330, 731, 761, 296, 573, 335, 716,
               642, 142, 674, 572, 638, 222, 752, 740, 397, 594, 705, 381, 615,
               539, 242, 499, 435, 680, 535, 238, 283,  89, 589, 666, 678,  76,
               176, 620,  75, 721, 143, 723, 570,  44, 203, 259, 677, 734, 662,
               707, 745, 487, 577, 443, 120, 111, 365, 116, 538, 162, 742, 212,
               581, 313,  36, 400, 619, 609, 252, 706, 264, 290, 138, 300, 346,
               712,  34, 387, 140, 154, 758, 462, 672, 713,  86, 414, 699, 529,
               382, 432, 368, 193,  72, 537, 560, 189, 342, 531, 311, 241, 685,
               497, 640, 321, 480, 144, 585, 171, 727, 660, 799, 600, 597, 213,
               708, 151, 265, 618, 658, 746, 307,  53, 514, 611, 153, 352, 225,
               567, 702, 520, 417, 102, 607, 548, 647, 476, 762, 147, 424, 459,
               409,  74, 510,  37, 323, 240, 175, 786,  80, 439, 504, 772, 670,
                59, 334, 703, 392,  90, 496, 422, 279, 343, 671, 794, 163, 328,
               625, 272, 227, 152, 105, 693,  96, 484, 568, 633, 659, 230, 112,
               793, 101, 172, 110, 612, 185, 289, 418, 533, 686, 641, 169, 349,
               173, 516,  62, 557, 596, 728, 371, 738, 444, 561, 114, 765, 338,
               588, 246, 295, 564, 488, 177, 687, 395, 518, 127, 639, 137, 354,
               271, 107, 340, 534, 768, 130, 720, 405, 430, 268, 108, 748, 351,
               393, 361, 170, 470], dtype=int64)
```

In [65]:
```
Triglycerides = df['Triglycerides'].value_counts(ascending=False)
Triglycerides
```

Out[65]:
```
799    25
507    22
121    22
593    22
469    22
       ..
120     3
213     3
185     3
295     3
130     2
Name: Triglycerides, Length: 771, dtype: int64
```

In [66]:
```
df['Triglycerides'].isnull().sum()
```

Out[66]:
```
0
```

In [67]:
```
df['Triglycerides'].isna().sum()
```

Out[67]:
```
0
```

19. Physical Activity Days Per Week feature

```
In [68]: df['Physical Activity Days Per Week'].unique()
```
```
Out[68]: array([0, 1, 4, 3, 5, 6, 7, 2], dtype=int64)
```

```
In [69]: df['Physical Activity Days Per Week'].isnull().sum()
```
```
Out[69]: 0
```

```
In [70]: df['Physical Activity Days Per Week'].isna().sum()
```
```
Out[70]: 0
```

20. Sleep Hours Per Day feature

```
In [71]: df['Sleep Hours Per Day'].unique()
```
```
Out[71]: array([ 6,  7,  4,  5, 10,  8,  9], dtype=int64)
```

```
In [72]: df['Sleep Hours Per Day'].isnull().sum()
```
```
Out[72]: 0
```

```
In [73]: df['Sleep Hours Per Day'].isna().sum()
```
```
Out[73]: 0
```

All features are cleaned, now let's head to the visualization part.

Data Visualization:

- Data visualization is the discipline of trying to understand data by placing it in a visual context so that patterns, trends, and correlations that might not otherwise be detected can be exposed.
- Python offers multiple great graphing libraries packed with lots of different features.
- The representation of data through use of common graphics, such as charts, plots, infographics & even animations.

- These visual displays of information communicate complex data relationships and data-driven insights in a way that is easy to understand.
- The main types of data visualization include charts, graphs and maps in the form of line charts, bar graphs, tree charts, dual-axis charts, mind maps, funnel charts and heatmaps
- Data visualization charts are graphical representations of data that tell a story using symbols in order to improve the understanding of large amounts of data.

Why Python is best for data visualization?

- Python has easy syntaxes and takes very less time to code things, also python provides different packages or libraries for data visualization using the features that exist.
- Some python libraries used for data visualization are matplotlib & seaborn and many other packages which are used for data visualization.

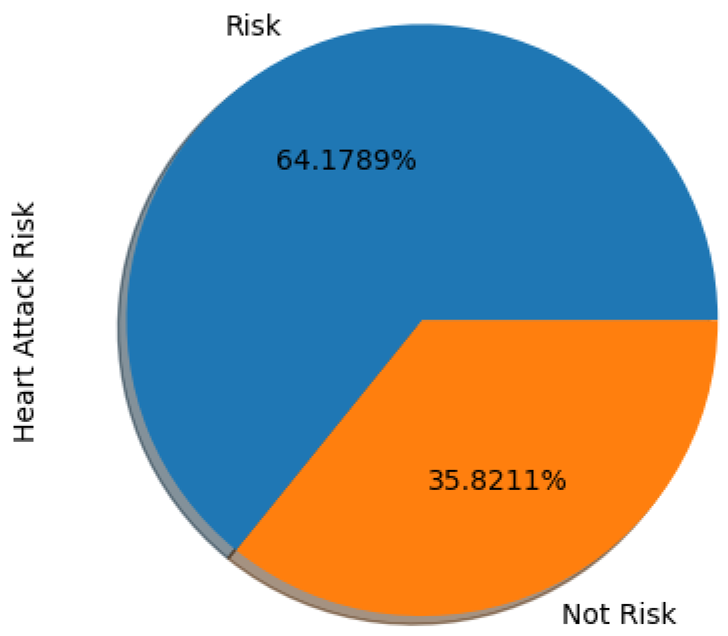
Why data visualization?

- Data visualization allows business users to gain insight into their vast amounts of data.
- It benefits them to recognize new patterns and errors in the data.
- Making sense of these patterns helps the users pay attention to areas that indicate red flags or progress.
- This process, in turn, drives the business ahead.

- Below you can see that the actual values for patients that can have a Heart Attack (Risky) and patients who are not likely to

In [74]:
```
df['Heart Attack Risk'].value_counts().plot.pie(autopct="%1.4f%%",labels=["Risk","Not Risk"], shadow=True,
                                                textprops={'color': 'black'})
```

Out[74]:
```
<Axes: ylabel='Heart Attack Risk'>
```



value_count() function:

- The value_counts() function is used to get a Series containing counts of unique values.
- The resulting object will be in descending order so that the first element is the most frequently-occurring element.
- Excludes NA values by default.

In [75]:
```
df['Heart Attack Risk'].value_counts()
```

Out[75]:
```
0    5624
1    3139
Name: Heart Attack Risk, dtype: int64
```

In [76]:
```
df.describe()
```

Out[76]:

	Age	Cholesterol	Heart Rate	Diabetes	Family History	Smoking	Obesity	Alcohol Consumption	Exercise Hours Per Week	
count	8763.000000	8763.000000	8763.000000	8763.000000	8763.000000	8763.000000	8763.000000	8763.000000	8763.000000	8763.00
mean	53.707977	259.877211	75.021682	0.652288	0.492982	0.896839	0.501426	0.598083	10.014284	1.007
std	21.249509	80.863276	20.550948	0.476271	0.499979	0.304186	0.500026	0.490313	5.783745	0.817
min	18.000000	120.000000	40.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.002442	0.00
25%	35.000000	192.000000	57.000000	0.000000	0.000000	1.000000	0.000000	0.000000	4.981579	0.00
50%	54.000000	259.000000	75.000000	1.000000	0.000000	1.000000	1.000000	1.000000	10.069559	1.00
75%	72.000000	330.000000	93.000000	1.000000	1.000000	1.000000	1.000000	1.000000	15.050018	2.00
max	90.000000	400.000000	110.000000	1.000000	1.000000	1.000000	1.000000	1.000000	19.998709	2.00

8 rows × 22 columns

- Above we can see that our Data is not balanced, it is 64% to 35%. Therefore, we should make the data balanced.
- In the mean section for Heart Attack Risk, the value is 0.358211. So It should be closer to 0.5 so that we can achieve the balanced data.

sort()

- The sort() method is a built-in Python method that, by default, sorts the list in ascending order: to put in a certain place or rank according to kind, class, or nature.
- However, you can modify the order from ascending to descending by specifying the sorting criteria.

How does the sort () method work?

- When sort() compares two values, it sends the values to the compare function, and sorts the values according to the returned (negative, zero, positive) value.
- **Example: The sort function will sort 40 as a value lower than 100. When comparing 40 and 100, sort() calls the function(40,100).**

sort_index()

- The sort_index() method sorts the DataFrame by the index.

In [77]:
```
heart_attack_risk_sorted = df['Heart Attack Risk'].value_counts().sort_index()
print(heart_attack_risk_sorted)
```

```
0    5624
1    3139
Name: Heart Attack Risk, dtype: int64
```

- Now let's check the values for the columns with CatPlot

CatPlot:

- catplot() method is used to plot categorical plots.
- With the use of one of many visual representations, this function gives users access to a number of axes-level functions that illustrate the connection between numerical data and one or more category variables.

What is Catplot used for?

- Figure-level interface for drawing categorical plots onto a FacetGrid.
- This function provides access to several axes-level functions that show the relationship between a numerical and one or more categorical variables using one of several visual representations.

What is the function of Catplot in seaborn?

- Catplot can handle 8 different plots currently available in Seaborn.
- catplot function can do all these types of plots and one can specify the type of plot one needs with the kind parameter.
- The default kind in catplot() is “strip”, corresponding to stripplot().

What are the different types of Catplots?

- catplot() function is that you can access several different kinds of plots–from boxplots and violin plots to swarm plots and bar plots–with just one function.
- Since sns.catplot() returns a FacetGrid object, which you can then tune easily to create beautiful & functional visualizations.

What is the background color of the seaborn Catplot?

- By default, plots made with seaborn have a white background.
- However, it is possible to change its background color passing a dictionary to the rc argument of the set_style function.

iloc[ ] function:

- It returns a Pandas Series when one row is selected, and a Pandas DataFrame when multiple rows are selected, or if any column in full is selected.
- To counter this, pass a single-valued list if you require DataFrame output.

In [78]:
```
plots = []

plots.append(sns.catplot(data=df.iloc[:, 0:5]))
plots.append(sns.catplot(data=df.iloc[:, 5:10]))
plots.append(sns.catplot(data=df.iloc[:, 10:15]))
plots.append(sns.catplot(data=df.iloc[:, 15:20]))
plots.append(sns.catplot(data=df.iloc[:, 20:22]))

for i in range(5):
    plots[i].set_xticklabels(rotation=90)
```

Inference from the CatPlot

- Above you can see the distribution of the values for each feature.
- For example, Previous Heart Attack Problems(1/0), Medication Use(1/0) are discrete values meaning that they represent a binary value (Yes/No).
- However, systolic_pressure and diastolic_pressure hold values such as 127, 128,235 etc.
- In addition to the values, we can observe that our data doesn't contain any outlier values. So we don't have to worry about them.

Heatmap:

- A two-dimensional graphical representation of data, each value in a matrix is represented by a different colour.
- With the help of the Matplotlib tools provided by the Seaborn package, annotated heatmaps can be created & customised to the creator's specifications.
- A heatmap (aka heat map) depicts values for a main variable of interest across two axis variables as a grid of colored squares.
- The axis variables are divided into ranges like a bar chart or histogram, and each cell's color indicates the value of the main variable in the corresponding cell range.

CORRELATION OF A HEATMAP:

- A correlation heatmap uses colored cells, typically in a monochromatic scale, to show a 2D correlation matrix (table) between two discrete dimensions or event types.
- The values of the first dimensions appear as rows of the table, while the values of the second dimension are represented by the columns of the table.
- A correlation heatmap is a graphical tool that displays the correlation between multiple variables as a color-coded matrix. It's like a color chart 🌈 that shows us how closely related different variables are.

What is heatmap used for?

- A heatmap is a graphical representation of data that uses a system of color coding to represent different values.
- Heatmaps are used in various forms of analytics but are most commonly used to show user behavior on specific web pages or webpage templates.

Summary of a heatmap:

- Heatmaps are a method of representing data graphically where values are depicted by color, making it easy to visualize complex data and understand it at a glance.
- Heatmaps can be created by hand, though modern heatmaps are generally created using specialized heatmapping software.

What is heatmap correlation Python?

- The strength and direction of the correlation between two pairs of variables in a dataset are displayed graphically in a correlation heatmap, which depicts the correlation matrix.
- It is an effective technique for finding patterns and connections in massive datasets.

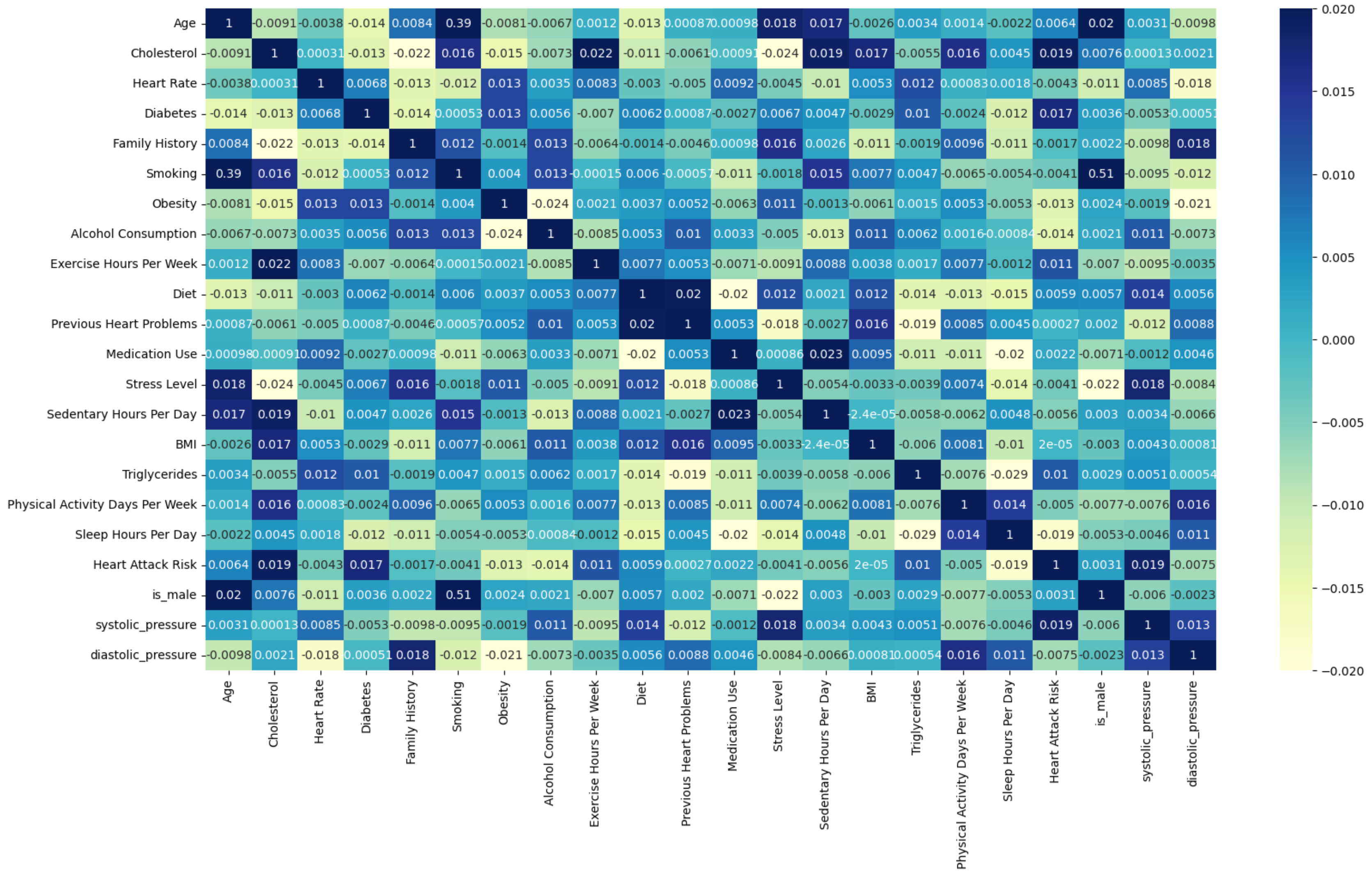
What are the advantages of heat map?

- Heat maps are excellent for getting to know a business area better.
- By measuring various demographics, you will better understand who your customers are and how to attract them.
- We can also use heat maps to analyze potential new markets for expansion.

- Heatmap: We have scaled the Heatmap between -0.020 and 0.020 so that we could observe the changes.

```
In [79]: plt.figure(figsize = (19,10))
         corelation=df.corr()
         sns.heatmap(corelation, cmap="YlGnBu", annot=True, vmin=-0.0200, vmax=0.0200)
```

```
Out[79]: <Axes: >
```



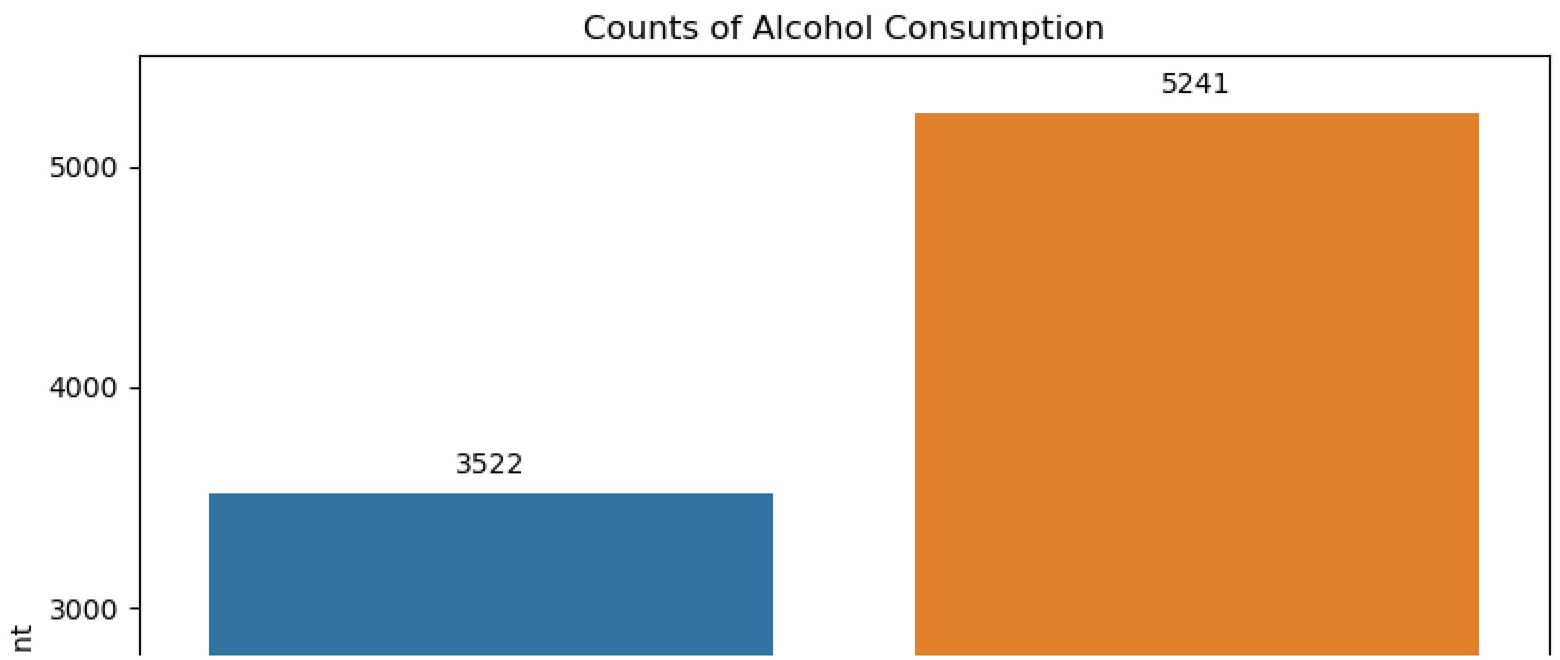
From this Heatmap, we can understand that, Cholesterol, Diabetes, Exercise Hours Per Week, Triglycerides, and systolic_pressure (blood pressure) is more related with Heart Attack Risk than other features.

BMI and Previous Heart Problems have low correlation with Heart Attack Risk. Therefore, we can drop them.

In [80]:
```
columns_to_drop = ['BMI','Previous Heart Problems']
df.drop(columns_to_drop, axis=1, inplace=True)
```

Counts of features:

In [81]:
```
plt.figure(figsize=(8, 7))

plots_to_show = df[['Alcohol Consumption', 'Stress Level', 'is_male','Diabetes','Family History','Smoking','Me

for i in plots_to_show.columns:
    ctg_num = plots_to_show[i].value_counts()
    chart = sns.barplot(x=ctg_num.index, y=ctg_num)

    for p in chart.patches:
        chart.annotate(format(p.get_height(), '.0f'), (p.get_x() + p.get_width() / 2., p.get_height()), ha='ce
                       va='center', xytext=(0, 10), textcoords='offset points', fontsize=10)

    plt.title(f'Counts of {i}')  # Set title for each plot
    plt.xlabel(i)  # Set x-axis label based on the column name
    plt.ylabel('Count')
    plt.tight_layout()  # Adjust layout to prevent overlapping

    plt.show()
```



Balancing The Imbalance Data

- Above it can be seen that Heart Attack Risk has a median of 0.358211 meaning that data is imbalanced & distributed as 64% to 36%.
- To fix this issue, we can use several methods. For our case we will be using Random Under sampling.

Importing the required libraries for predictions

In [82]:
```
conda install -c conda-forge imbalanced-learn
```

```
Retrieving notices: ...working... done
Collecting package metadata (current_repodata.json): ...working... done
Solving environment: ...working... done

# All requested packages already installed.


Note: you may need to restart the kernel to use updated packages.
```

```
DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): repo.anaconda.com:443
DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): conda.anaconda.org:443
DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): repo.anaconda.com:443
DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): repo.anaconda.com:443
DEBUG:urllib3.connectionpool:https://repo.anaconda.com:443 "GET /pkgs/main/notices.json HTTP/1.1" 404 None
DEBUG:urllib3.connectionpool:https://repo.anaconda.com:443 "GET /pkgs/msys2/notices.json HTTP/1.1" 404 None
DEBUG:urllib3.connectionpool:https://repo.anaconda.com:443 "GET /pkgs/r/notices.json HTTP/1.1" 404 None
DEBUG:urllib3.connectionpool:https://conda.anaconda.org:443 "GET /conda-forge/notices.json HTTP/1.1" 404 Non
e
DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): conda.anaconda.org:443
DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): conda.anaconda.org:443
DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): repo.anaconda.com:443
DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): repo.anaconda.com:443
DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): repo.anaconda.com:443
DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): repo.anaconda.com:443
DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): repo.anaconda.com:443
DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): repo.anaconda.com:443
DEBUG:urllib3.connectionpool:https://repo.anaconda.com:443 "GET /pkgs/main/noarch/current_repodata.json HTT
P/1.1" 200 None
DEBUG:urllib3.connectionpool:https://repo.anaconda.com:443 "GET /pkgs/msys2/win-64/current_repodata.json HTT
P/1.1" 304 0
DEBUG:urllib3.connectionpool:https://repo.anaconda.com:443 "GET /pkgs/r/noarch/current_repodata.json HTTP/1.
1" 304 0
DEBUG:urllib3.connectionpool:https://repo.anaconda.com:443 "GET /pkgs/r/win-64/current_repodata.json HTTP/1.
1" 304 0
DEBUG:urllib3.connectionpool:https://repo.anaconda.com:443 "GET /pkgs/main/win-64/current_repodata.json HTT
P/1.1" 200 None
DEBUG:urllib3.connectionpool:https://repo.anaconda.com:443 "GET /pkgs/msys2/noarch/current_repodata.json HTT
P/1.1" 304 0
DEBUG:urllib3.connectionpool:https://conda.anaconda.org:443 "GET /conda-forge/noarch/current_repodata.json H
TTP/1.1" 200 None
DEBUG:urllib3.connectionpool:https://conda.anaconda.org:443 "GET /conda-forge/win-64/current_repodata.json H
TTP/1.1" 200 None


==> WARNING: A newer version of conda exists. <==
  current version: 23.7.4
  latest version: 23.11.0

Please update conda by running

    $ conda update -n base -c defaults conda
```

```
Or to minimize the number of packages updated during conda update use

     conda install conda=23.11.0
```

TensorFlow:

- TensorFlow is a free and open-source software library for machine learning and artificial intelligence.
- It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks.
- TensorFlow allows you to create dataflow graphs that describe how data moves through a graph.
- The graph consists of nodes that represent a mathematical operation & A connection/edge between nodes is a multidimensional data array.
- A framework composed of two core building blocks: a library for defining computational graphs and a runtime for executing such graphs on a variety of different hardware.

What is TensorFlow used for?

- The TensorFlow platform helps you implement best practices for data automation, model tracking, performance monitoring & model retraining.
- Using production-level tools to automate & track model training over the lifetime of a product, service or business process is critical to success.

Advantages of TensorFlow:

- TensorFlow allows developers to create dataflow graphs—structures that describe how data moves through a graph, or a series of processing nodes.
- Each node in the graph represents a mathematical operation, and each connection or edge between nodes is a multidimensional data array, or tensor.

Where is TensorFlow mostly used?

- Google uses TensorFlow to power ML implementations in products like Search, Gmail, and Translate, to aid researchers in new discoveries & even to forge advances in humanitarian and environmental challenges.

Why to use TensorFlow in AI?

- TensorFlow is used to create a large-scale neural network with many layers.

- It is mainly used for deep learning or machine learning problems such as Classification, Perception, Understanding, Discovering Prediction & Creation.

Scikit-learn:

- An open source data analysis library and the gold standard for Machine Learning (ML) in the Python ecosystem.
- Scikit-learn is a Python module for machine learning built on top of SciPy (Scientific Python).
- Key concepts and features include: Algorithmic decision-making methods, including: Classification: identifying & categorizing data based on patterns.

What is scikit-learn used for?

- Scikit-Learn, also known as sklearn is a python library to implement machine learning models and statistical modelling.
- Through scikit-learn, we implement various ML models for regression, classification, clustering & statistical tools for analyzing these models.

Imblearn:

- Imbalanced-Learn is a Python module that helps in balancing the datasets which are highly skewed or biased towards some classes.
- Thus, it helps in resampling the classes which are otherwise oversampled or undesampled.
- If there is a greater imbalance ratio, the output is biased to the class which has a higher number of examples.

What is Imblearn used for?

- Imblearn techniques are the methods by which we can generate a data set that has an equal ratio of classes.
- The predictive model built on this type of data set would be able to generalize well.

In [83]:
```
!pip install Tensorflow
!pip install scikit-learn==1.2.2
!pip install imblearn
```

```
Requirement already satisfied: Tensorflow in c:\users\admin\anaconda3\lib\site-packages (2.15.0)
Requirement already satisfied: tensorflow-intel==2.15.0 in c:\users\admin\anaconda3\lib\site-packages (from
Tensorflow) (2.15.0)
Requirement already satisfied: absl-py>=1.0.0 in c:\users\admin\anaconda3\lib\site-packages (from tensorflow
-intel==2.15.0->Tensorflow) (2.0.0)
Requirement already satisfied: astunparse>=1.6.0 in c:\users\admin\anaconda3\lib\site-packages (from tensorf
low-intel==2.15.0->Tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=23.5.26 in c:\users\admin\anaconda3\lib\site-packages (from tens
orflow-intel==2.15.0->Tensorflow) (23.5.26)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in c:\users\admin\anaconda3\lib\site-pack
ages (from tensorflow-intel==2.15.0->Tensorflow) (0.5.4)
Requirement already satisfied: google-pasta>=0.1.1 in c:\users\admin\anaconda3\lib\site-packages (from tenso
rflow-intel==2.15.0->Tensorflow) (0.2.0)
Requirement already satisfied: h5py>=2.9.0 in c:\users\admin\anaconda3\lib\site-packages (from tensorflow-in
tel==2.15.0->Tensorflow) (3.7.0)
Requirement already satisfied: libclang>=13.0.0 in c:\users\admin\anaconda3\lib\site-packages (from tensorfl
ow-intel==2.15.0->Tensorflow) (16.0.6)
Requirement already satisfied: ml-dtypes~=0.2.0 in c:\users\admin\anaconda3\lib\site-packages (from tensorfl
ow-intel==2.15.0->Tensorflow) (0.2.0)
Requirement already satisfied: numpy<2.0.0,>=1.23.5 in c:\users\admin\anaconda3\lib\site-packages (from tens
orflow-intel==2.15.0->Tensorflow) (1.24.3)
Requirement already satisfied: opt-einsum>=2.3.2 in c:\users\admin\anaconda3\lib\site-packages (from tensorf
low-intel==2.15.0->Tensorflow) (3.3.0)
Requirement already satisfied: packaging in c:\users\admin\anaconda3\lib\site-packages (from tensorflow-inte
l==2.15.0->Tensorflow) (23.0)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.2
0.3 in c:\users\admin\anaconda3\lib\site-packages (from tensorflow-intel==2.15.0->Tensorflow) (4.23.4)
Requirement already satisfied: setuptools in c:\users\admin\anaconda3\lib\site-packages (from tensorflow-int
el==2.15.0->Tensorflow) (68.0.0)
Requirement already satisfied: six>=1.12.0 in c:\users\admin\anaconda3\lib\site-packages (from tensorflow-in
tel==2.15.0->Tensorflow) (1.16.0)
Requirement already satisfied: termcolor>=1.1.0 in c:\users\admin\anaconda3\lib\site-packages (from tensorfl
ow-intel==2.15.0->Tensorflow) (2.4.0)
Requirement already satisfied: typing-extensions>=3.6.6 in c:\users\admin\anaconda3\lib\site-packages (from
tensorflow-intel==2.15.0->Tensorflow) (4.7.1)
Requirement already satisfied: wrapt<1.15,>=1.11.0 in c:\users\admin\anaconda3\lib\site-packages (from tenso
rflow-intel==2.15.0->Tensorflow) (1.14.1)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in c:\users\admin\anaconda3\lib\site-pac
kages (from tensorflow-intel==2.15.0->Tensorflow) (0.31.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in c:\users\admin\anaconda3\lib\site-packages (from tenso
rflow-intel==2.15.0->Tensorflow) (1.60.0)
Requirement already satisfied: tensorboard<2.16,>=2.15 in c:\users\admin\anaconda3\lib\site-packages (from t
ensorflow-intel==2.15.0->Tensorflow) (2.15.1)
```

```
Requirement already satisfied: tensorflow-estimator<2.16,>=2.15.0 in c:\users\admin\anaconda3\lib\site-packa
ges (from tensorflow-intel==2.15.0->Tensorflow) (2.15.0)
Requirement already satisfied: keras<2.16,>=2.15.0 in c:\users\admin\anaconda3\lib\site-packages (from tenso
rflow-intel==2.15.0->Tensorflow) (2.15.0)
Requirement already satisfied: wheel<1.0,>=0.23.0 in c:\users\admin\anaconda3\lib\site-packages (from astunp
arse>=1.6.0->tensorflow-intel==2.15.0->Tensorflow) (0.38.4)
Requirement already satisfied: google-auth<3,>=1.6.3 in c:\users\admin\anaconda3\lib\site-packages (from ten
sorboard<2.16,>=2.15->tensorflow-intel==2.15.0->Tensorflow) (2.25.2)
Requirement already satisfied: google-auth-oauthlib<2,>=0.5 in c:\users\admin\anaconda3\lib\site-packages (f
rom tensorboard<2.16,>=2.15->tensorflow-intel==2.15.0->Tensorflow) (1.2.0)
Requirement already satisfied: markdown>=2.6.8 in c:\users\admin\anaconda3\lib\site-packages (from tensorboa
rd<2.16,>=2.15->tensorflow-intel==2.15.0->Tensorflow) (3.4.1)
Requirement already satisfied: requests<3,>=2.21.0 in c:\users\admin\anaconda3\lib\site-packages (from tenso
rboard<2.16,>=2.15->tensorflow-intel==2.15.0->Tensorflow) (2.31.0)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in c:\users\admin\anaconda3\lib\site-pa
ckages (from tensorboard<2.16,>=2.15->tensorflow-intel==2.15.0->Tensorflow) (0.7.2)
Requirement already satisfied: werkzeug>=1.0.1 in c:\users\admin\anaconda3\lib\site-packages (from tensorboa
rd<2.16,>=2.15->tensorflow-intel==2.15.0->Tensorflow) (2.2.3)
Requirement already satisfied: cachetools<6.0,>=2.0.0 in c:\users\admin\anaconda3\lib\site-packages (from go
ogle-auth<3,>=1.6.3->tensorboard<2.16,>=2.15->tensorflow-intel==2.15.0->Tensorflow) (5.3.2)
Requirement already satisfied: pyasn1-modules>=0.2.1 in c:\users\admin\anaconda3\lib\site-packages (from goo
gle-auth<3,>=1.6.3->tensorboard<2.16,>=2.15->tensorflow-intel==2.15.0->Tensorflow) (0.2.8)
Requirement already satisfied: rsa<5,>=3.1.4 in c:\users\admin\anaconda3\lib\site-packages (from google-auth
<3,>=1.6.3->tensorboard<2.16,>=2.15->tensorflow-intel==2.15.0->Tensorflow) (4.9)
Requirement already satisfied: requests-oauthlib>=0.7.0 in c:\users\admin\anaconda3\lib\site-packages (from
google-auth-oauthlib<2,>=0.5->tensorboard<2.16,>=2.15->tensorflow-intel==2.15.0->Tensorflow) (1.3.1)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\admin\anaconda3\lib\site-packages (from
requests<3,>=2.21.0->tensorboard<2.16,>=2.15->tensorflow-intel==2.15.0->Tensorflow) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in c:\users\admin\anaconda3\lib\site-packages (from requests<3,>
=2.21.0->tensorboard<2.16,>=2.15->tensorflow-intel==2.15.0->Tensorflow) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\admin\anaconda3\lib\site-packages (from reques
ts<3,>=2.21.0->tensorboard<2.16,>=2.15->tensorflow-intel==2.15.0->Tensorflow) (1.26.16)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\admin\anaconda3\lib\site-packages (from reques
ts<3,>=2.21.0->tensorboard<2.16,>=2.15->tensorflow-intel==2.15.0->Tensorflow) (2023.11.17)
Requirement already satisfied: MarkupSafe>=2.1.1 in c:\users\admin\anaconda3\lib\site-packages (from werkzeu
g>=1.0.1->tensorboard<2.16,>=2.15->tensorflow-intel==2.15.0->Tensorflow) (2.1.1)
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in c:\users\admin\anaconda3\lib\site-packages (from pyas
n1-modules>=0.2.1->google-auth<3,>=1.6.3->tensorboard<2.16,>=2.15->tensorflow-intel==2.15.0->Tensorflow) (0.
4.8)
Requirement already satisfied: oauthlib>=3.0.0 in c:\users\admin\anaconda3\lib\site-packages (from requests-
oauthlib>=0.7.0->google-auth-oauthlib<2,>=0.5->tensorboard<2.16,>=2.15->tensorflow-intel==2.15.0->Tensorflo
w) (3.2.2)
Requirement already satisfied: scikit-learn==1.2.2 in c:\users\admin\anaconda3\lib\site-packages (1.2.2)
Requirement already satisfied: numpy>=1.17.3 in c:\users\admin\anaconda3\lib\site-packages (from scikit-lear
n==1.2.2) (1.24.3)
Requirement already satisfied: scipy>=1.3.2 in c:\users\admin\anaconda3\lib\site-packages (from scikit-learn
==1.2.2) (1.10.1)
Requirement already satisfied: joblib>=1.1.1 in c:\users\admin\anaconda3\lib\site-packages (from scikit-lear
n==1.2.2) (1.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\admin\anaconda3\lib\site-packages (from scik
it-learn==1.2.2) (2.2.0)
Requirement already satisfied: imblearn in c:\users\admin\anaconda3\lib\site-packages (0.0)
Requirement already satisfied: imbalanced-learn in c:\users\admin\anaconda3\lib\site-packages (from imblear
n) (0.11.0)
Requirement already satisfied: numpy>=1.17.3 in c:\users\admin\anaconda3\lib\site-packages (from imbalanced-
learn->imblearn) (1.24.3)
Requirement already satisfied: scipy>=1.5.0 in c:\users\admin\anaconda3\lib\site-packages (from imbalanced-l
earn->imblearn) (1.10.1)
Requirement already satisfied: scikit-learn>=1.0.2 in c:\users\admin\anaconda3\lib\site-packages (from imbal
anced-learn->imblearn) (1.2.2)
Requirement already satisfied: joblib>=1.1.1 in c:\users\admin\anaconda3\lib\site-packages (from imbalanced-
learn->imblearn) (1.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\admin\anaconda3\lib\site-packages (from imba
lanced-learn->imblearn) (2.2.0)
```

SMOTETomek

- SMOTETomek is somewhere upsampling and downsampling.
- SMOTETomek is a hybrid method which is a mixture of the above two methods, it uses an under-sampling method (Tomek) with an oversampling method (SMOTE).

What do you mean by smote in detail?

- Synthetic Minority Oversampling Technique (SMOTE) is a statistical technique for increasing the number of cases in your dataset in a balanced way.
- The component works by generating new instances from existing minority cases that you supply as input.

What is the smote sample strategy?

- At a high level, the SMOTE algorithm can be described as follows:

1. Take difference between a sample and its nearest neighbour.
2. Multiply the difference by a random number between 0 and 1.

- Add this difference to the sample to generate a new synthetic example in feature space.

What is the function of smote?

- SMOTE (synthetic minority oversampling technique) is one of the most commonly used oversampling methods to solve the imbalance problem.
- It aims to balance class distribution by randomly increasing minority class examples by replicating them.
- SMOTE synthesises new minority instances between existing minority instances.

Benefits of SMOTE:

- Reduced risk of overfitting: By generating new synthetic samples, instead of simply duplicating existing samples, SMOTE can help to reduce the risk of overfitting which commonly accompanies random oversampling.
- Hence, SMOTE is meant to be an improvement over random oversampling.

What are the undersampling techniques?

- Undersampling is a technique that can reduce the size of the majority class in a dataset.
- It involves removing samples from the majority class until it matches the size of the minority class or until specific criteria are met.

Advantages of undersampling:

- No need to introduce redundant information into your dataset.
- The main advantage that undersampling has is that you do not have to add any artificial observations to your dataset that introduce repeated or redundant information into your dataset.

Which undersampling technique is best?

- The simplest, most explainable, and oftentime most effective techniques for performing sampling are random under- and over-sampling.
- These two sampling methodologies have broadly simpler characteristics.

How do you overcome undersampling?

- Under-sampling balances the dataset by reducing the size of the abundant class.
- This method is used when quantity of data is sufficient.
- By keeping all samples in the rare class and randomly selecting an equal number of samples in the abundant class, a balanced new dataset can be retrieved for further modelling

What is oversampling in machine learning?

- Random Oversampling involves supplementing the training data with multiple copies of some of the minority classes.
- Oversampling can be done more than once (2x, 3x, 5x, 10x, etc.).
- This is one of the earliest proposed methods, that is also proven to be robust.

What is the purpose of oversampling?

- Oversampling is capable of improving resolution and signal-to-noise ratio, and can be helpful in avoiding aliasing and phase distortion by relaxing anti-aliasing filter performance requirements.
- A signal is said to be oversampled by a factor of N if it is sampled at N times the Nyquist rate.

Is undersampling better than oversampling?

- In extreme cases where the number of observations in the rare class(es) is really small, oversampling is better, as you will not lose important information on the distribution of the other classes in the dataset.

NearMiss:

- NearMiss is an under-sampling technique.
- It aims to balance class distribution by randomly eliminating majority class examples.
- When instances of two different classes are very close to each other, we remove the instances of the majority class to increase the spaces between the two classes.

How does NearMiss work?

- NearMiss – Version 1 : It selects samples of the majority class for which average distances to the k closest instances of the minority class is smallest.
- NearMiss – Version 2 : It selects samples of the majority class for which average distances to the k farthest instances of the minority class is smallest.

In [84]:
```python
from imblearn.combine import SMOTETomek
from imblearn.under_sampling import NearMiss
```

MinMaxScaler:

- Transform features by scaling each feature to a given range.
- Scales each feature by shrinking the range to a defined minimum and maximum, typically between 0 and 1.
- The transformation is based on the minimum and maximum values of each feature.

What is the function of min-max scaling?

- A function for min-max scaling of pandas DataFrames or NumPy arrays.
- An alternative approach to Z-score normalization (or standardization) is the so-called Min-Max scaling (often also simply called "normalization" - a common cause for ambiguities).
- In this approach, the data is scaled to a fixed range - usually 0 to 1.

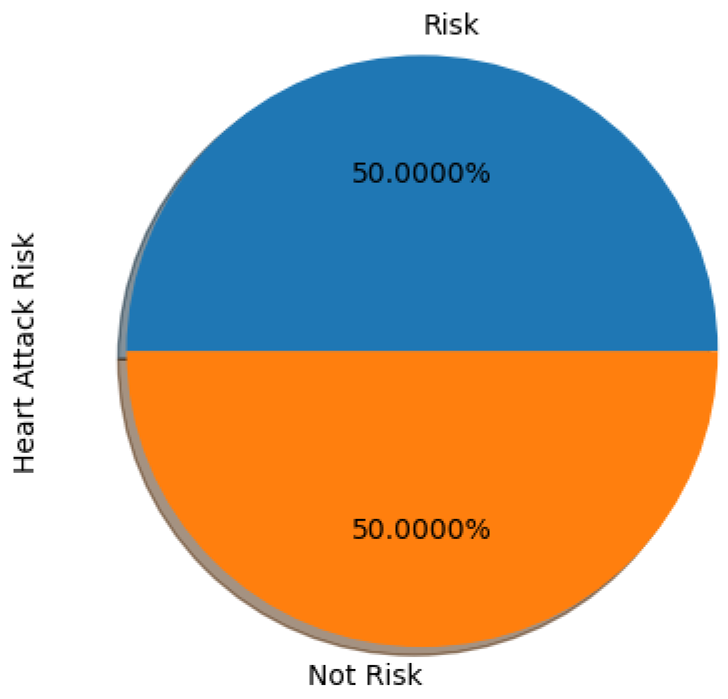
In [85]:
```
from imblearn.under_sampling import RandomUnderSampler
from imblearn.over_sampling import RandomOverSampler
from sklearn.preprocessing import MinMaxScaler

random_over_sampler = RandomOverSampler(sampling_strategy=1)

y = df['Heart Attack Risk']
X = df.drop(['Heart Attack Risk'], axis=1)

X, y = random_over_sampler.fit_resample(X,y)

scaler = MinMaxScaler()
X_min_max = pd.DataFrame(scaler.fit_transform(X))

print(pd.DataFrame(y).describe())
```

```
       Heart Attack Risk
count       11248.000000
mean            0.500000
std             0.500022
min             0.000000
25%             0.000000
50%             0.500000
75%             1.000000
max             1.000000
```

In [86]:
```
balanced_df = pd.concat([pd.DataFrame(X), pd.DataFrame(y)], axis=1)
```

In [87]:
```
balanced_df['Heart Attack Risk'].value_counts().plot.pie(autopct="%1.4f%%",labels=["Risk","Not Risk"], shadow=
                                                          textprops={'color': 'black'})
```

Out[87]:
```
<Axes: ylabel='Heart Attack Risk'>
```



In [88]:
```
balanced_df['Heart Attack Risk'].value_counts()
```

Out[88]:
```
0    5624
1    5624
Name: Heart Attack Risk, dtype: int64
```

Observation:

- Above you can see that now our data is balanced, and we used Over Sampling for that.
- Since we've used over sampling, Risky was 3139, now it is the same with not Risky which is 5624.

- The reason why we've used over sampling is that our data was not big enough for under sampling and it gave low accuracy when using it.

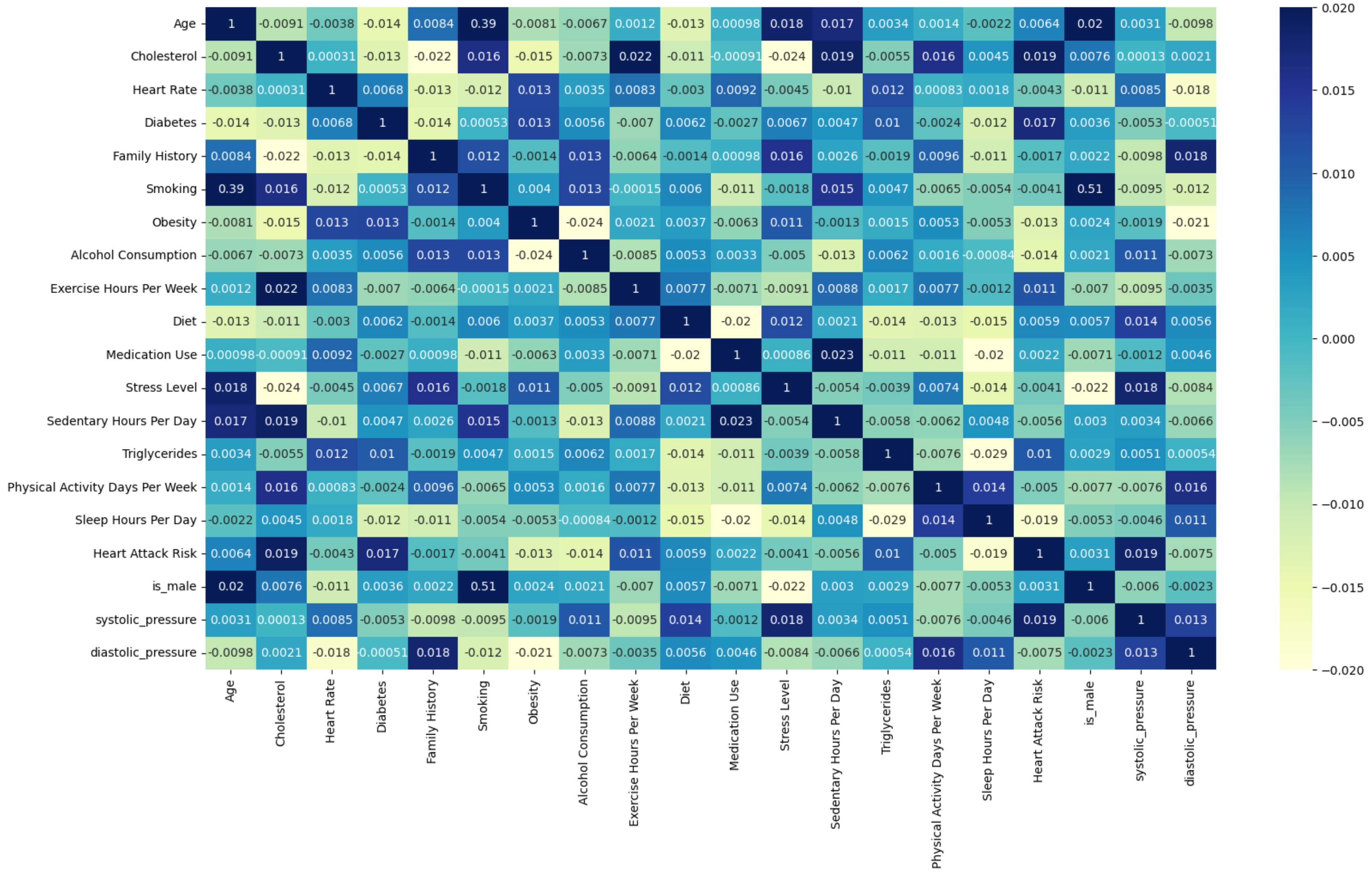
In [89]:
```
balanced_df.describe()
```

Out[89]:

	Age	Cholesterol	Heart Rate	Diabetes	Family History	Smoking	Obesity	Alcohol Consumption	Exercise Hours Per Week
count	11248.000000	11248.000000	11248.000000	11248.000000	11248.000000	11248.000000	11248.000000	11248.000000	11248.000000
mean	53.684477	260.255690	74.942568	0.653183	0.494043	0.895359	0.497600	0.597173	10.038608
std	21.289991	80.848685	20.507665	0.475978	0.499987	0.306104	0.500016	0.490488	5.811297
min	18.000000	120.000000	40.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.002442
25%	35.000000	192.000000	57.000000	0.000000	0.000000	1.000000	0.000000	0.000000	4.977367
50%	54.000000	259.000000	75.000000	1.000000	0.000000	1.000000	0.000000	1.000000	10.115655
75%	72.000000	331.000000	93.000000	1.000000	1.000000	1.000000	1.000000	1.000000	15.097196
max	90.000000	400.000000	110.000000	1.000000	1.000000	1.000000	1.000000	1.000000	19.998709

In [90]:
```
#HeatMap

plt.figure(figsize = (19,10))
corelation=df.corr()
sns.heatmap(corelation, cmap="YlGnBu",annot=True,vmin=-0.0200, vmax=0.0200)
```

Out[90]:
```
<Axes: >
```



	Age	Cholesterol	Heart Rate	Diabetes	Family History	Smoking	Obesity	Alcohol Consumption	Exercise Hours Per Week	Diet	Medication Use	Stress Level	Sedentary Hours Per Day	Triglycerides	Physical Activity Days Per Week	Sleep Hours Per Day	Heart Attack Risk	is_male	systolic_pressure	diastolic_pressure
Age	1	-0.0091	-0.0038	-0.014	0.0084	0.39	-0.0081	-0.0067	0.0012	-0.013	0.00098	0.018	0.017	0.0034	0.0014	-0.0022	0.0064	0.02	0.0031	-0.0098
Cholesterol	-0.0091	1	0.00031	-0.013	-0.022	0.016	-0.015	-0.0073	0.022	-0.011	-0.00091	-0.024	0.019	-0.0055	0.016	0.0045	0.019	0.0076	0.00013	0.0021
Heart Rate	-0.0038	0.00031	1	0.0068	-0.013	-0.012	0.013	0.0035	0.0083	-0.003	0.0092	-0.0045	-0.01	0.012	0.00083	0.0018	-0.0043	-0.011	0.0085	-0.018
Diabetes	-0.014	-0.013	0.0068	1	-0.014	0.00053	0.013	0.0056	-0.007	0.0062	-0.0027	0.0067	0.0047	0.01	-0.0024	-0.012	0.017	0.0036	-0.0053	-0.00051
Family History	0.0084	-0.022	-0.013	-0.014	1	0.012	-0.0014	0.013	-0.0064	-0.0014	0.00098	0.016	0.0026	-0.0019	0.0096	-0.011	-0.0017	0.0022	-0.0098	0.018
Smoking	0.39	0.016	-0.012	0.00053	0.012	1	0.004	0.013	-0.00015	0.006	-0.011	-0.0018	0.015	0.0047	-0.0065	-0.0054	-0.0041	0.51	-0.0095	-0.012
Obesity	-0.0081	-0.015	0.013	0.013	-0.0014	0.004	1	-0.024	0.0021	0.0037	-0.0063	0.011	-0.0013	0.0015	0.0053	-0.0053	-0.013	0.0024	-0.0019	-0.021
Alcohol Consumption	-0.0067	-0.0073	0.0035	0.0056	0.013	0.013	-0.024	1	-0.0085	0.0053	0.0033	-0.005	-0.013	0.0062	0.0016	-0.00084	-0.014	0.0021	0.011	-0.0073
Exercise Hours Per Week	0.0012	0.022	0.0083	-0.007	-0.0064	-0.00015	0.0021	-0.0085	1	0.0077	-0.0071	-0.0091	0.0088	0.0017	0.0077	-0.0012	0.011	-0.007	-0.0095	-0.0035
Diet	-0.013	-0.011	-0.003	0.0062	-0.0014	0.006	0.0037	0.0053	0.0077	1	-0.02	0.012	0.0021	-0.014	-0.013	-0.015	0.0059	0.0057	0.014	0.0056
Medication Use	0.00098	-0.00091	0.0092	-0.0027	0.00098	-0.011	-0.0063	0.0033	-0.0071	-0.02	1	0.00086	0.023	-0.011	-0.011	-0.02	0.0022	-0.0071	-0.0012	0.0046
Stress Level	0.018	-0.024	-0.0045	0.0067	0.016	-0.0018	0.011	-0.005	-0.0091	0.012	0.00086	1	-0.0054	-0.0039	0.0074	-0.014	-0.0041	-0.022	0.018	-0.0084
Sedentary Hours Per Day	0.017	0.019	-0.01	0.0047	0.0026	0.015	-0.0013	-0.013	0.0088	0.0021	0.023	-0.0054	1	-0.0058	-0.0062	0.0048	-0.0056	0.003	0.0034	-0.0066
Triglycerides	0.0034	-0.0055	0.012	0.01	-0.0019	0.0047	0.0015	0.0062	0.0017	-0.014	-0.011	-0.0039	-0.0058	1	-0.0076	-0.029	0.01	0.0029	0.0051	0.00054
Physical Activity Days Per Week	0.0014	0.016	0.00083	-0.0024	0.0096	-0.0065	0.0053	0.0016	0.0077	-0.013	-0.011	0.0074	-0.0062	-0.0076	1	0.014	-0.005	-0.0077	-0.0076	0.016
Sleep Hours Per Day	-0.0022	0.0045	0.0018	-0.012	-0.011	-0.0054	-0.0053	-0.00084	-0.0012	-0.015	-0.02	-0.014	0.0048	-0.029	0.014	1	-0.019	-0.0053	-0.0046	0.011
Heart Attack Risk	0.0064	0.019	-0.0043	0.017	-0.0017	-0.0041	-0.013	-0.014	0.011	0.0059	0.0022	-0.0041	-0.0056	0.01	-0.005	-0.019	1	0.0031	0.019	-0.0075
is_male	0.02	0.0076	-0.011	0.0036	0.0022	0.51	0.0024	0.0021	-0.007	0.0057	-0.0071	-0.022	0.003	0.0029	-0.0077	-0.0053	0.0031	1	-0.006	-0.0023
systolic_pressure	0.0031	0.00013	0.0085	-0.0053	-0.0098	-0.0095	-0.0019	0.011	-0.0095	0.014	-0.0012	0.018	0.0034	0.0051	-0.0076	-0.0046	0.019	-0.006	1	0.013
diastolic_pressure	-0.0098	0.0021	-0.018	-0.00051	0.018	-0.012	-0.021	-0.0073	-0.0035	0.0056	0.0046	-0.0084	-0.0066	0.00054	0.016	0.011	-0.0075	-0.0023	0.013	1

Feature Selection & Score Calculation:

- For the feature selection there are few methods. So, Below we can find the list of methods for feature selection.

1. Filter Methods
2. Chi-Squared Test
3. SelectKBest
4. Mutual Information Classification
5. Wrapper Methods
6. RFE (Recursive Feature Elimination)
7. Forward Selection
8. Backward Elimination
9. Embedded Methods
10. LASSO Regression
11. Tree Based Methods
12. RandomForestClassifier
13. DecisionTreeClassifier
14. Gradient Boost
15. Principal Component Analysis (PCA)
16. 1- Chi-Squared Test
17. Mutual Info Classification

- Since our data is balanced we will compare the Accuracy score of the models & for the feature numbers, we will iterate through all number of features to determine which feature fits the best.

sklearn Feature_selection:

- The classes in the sklearn.feature_selection module can be used for feature selection/dimensionality reduction on sample sets, either to improve estimators' accuracy scores or to boost their performance on very high-dimensional datasets.

Feature Selection:

- Feature Selection is the method of reducing the input variable to your model by using only relevant data and getting rid of noise in data.
- It is the process of automatically choosing relevant features for your machine learning model based on the type of problem you are trying to solve.

Why is feature selection important?

- In the machine learning process, feature selection is used to make the process more accurate.
- It also increases the prediction power of the algorithms by selecting the most critical variables and eliminating the redundant and irrelevant ones.

What are the three types of feature selection methods?

- Let's build a simple voting selector that ensembles three different features selection methods:
- 1. A filter method based on Pearson correlation.
- 2. An unsupervised method based on multicollinearity.
- 3. A wrapper, Recursive Feature Elimination.

sklearn Model_selection:

- Sklearn's model selection module provides various functions to cross-validate our model, tune the estimator's hyperparameters, or produce validation and learning curves.
- Model_selection is a method for setting a blueprint to analyze data and then using it to measure new data.
- Selecting a proper model allows you to generate accurate results when making a prediction.
- To do that, you need to train your model by using a specific dataset.
- Then, you test the model against another dataset.

What does Sklearn Model_selection do in Python?

- model_selection package in Python splits arrays or matrices into random subsets for train and test data, respectively.

Filter Methods:

- Filter methods are generally used as a preprocessing step.
- The selection of features is independent of any machine learning algorithms.
- Instead, features are selected on the basis of their scores in various statistical tests for their correlation with the outcome variable.
- The filter method filters out the irrelevant feature and redundant columns from the model by using different metrics through ranking.
- The advantage of using filter methods is that it needs low computational time & does not overfit the data.
- The features are filtered based on general characteristics (some metric such as correlation) of the dataset such correlation with the dependent variable.

Wrapper Methods:

- Also referred to as greedy algorithms train the algorithm by using a subset of features in an iterative manner.

- Based on the conclusions made from training in prior to the model, addition and removal of features takes place.
- These methods are usually computationally very expensive.

Purpose of a wrapper method:

- To make writing computer programs easier by abstracting away the details of a subroutine's underlying implementation.

What is a wrapper model?

- A model wrapper is passed in as part of the worker options dictionary when you register a task and defines the following:
- 1. Model training logic (TrainAsync()) for a specific model.
- 2. Model evaluation logic (EvaluateAsync()) for a specific model.
- 3. Projection logic (Projection) that defines the structure.

Example of Wrapper Methods:

- On the basis of the output of the model, features are added or subtracted, and with this feature set, the model has trained again.
- Some techniques of wrapper methods are: Forward selection - Forward selection is an iterative process, which begins with an empty set of features.

Embedded Methods:

- These are fast processing methods similar to the filter method but more accurate than the filter method.
- These methods are also iterative, which evaluates each iteration and optimally finds the most important features that contribute the most to training in a particular iteration.

Forward Selection:

- In forward selection, the first variable selected for an entry into the constructed model is the one with the largest correlation with the dependent variable.
- Once the variable has been selected, it is evaluated on the basis of certain criteria.
- Forward feature selection allows us to tune this hyperparameter for optimal performance.
- Results are in perfect alignment with our observation.
- This technique is model agnostic and can be used for evaluating feature importance for any classification/regression model.

Backward Elimination:

- The backward elimination technique is used in ML to find the best subset of features from a given set of features.

- It works by iteratively removing features that are not predictive of the target variable or have the least predictive power.
- Start with model containing all possible explanatory variables.
- For each variable in turn, investigate effect of removing that variable from the current model.
- Remove the least informative variable, unless this variable is nonetheless supplying significant information about the response.
- Backward elimination has a further advantage, in that several factors together may have better predictive power than any subset of these factors.
- As a result, the backward elimination process is more likely to include these factors as a group in the final model than is the forward selection process.

METHOD - 1. Chi-square:

- A statistical test used to examine the differences between categorical variables from a random sample in order to judge the goodness of fit between expected and observed results or a statistical test used to compare observed results with expected results.
- The chi-square test is a hypothesis test used for categorical variables with nominal or ordinal measurement scale.

What are the three properties of chi-square distribution?

- The following are the important properties of the chi-square test:
- 1. Two times the number of degrees of freedom is equal to the variance.
- 2. The number of degree of freedom is equal to the mean distribution.
- 3. The chi-square distribution curve approaches the normal distribution when the degree of freedom increases.

What is the application of chi-square distribution?

- Using the chi-square distribution, you can test the hypothesis that a population variance is equal to a certain value using the test of a single variance or calculate confidence intervals for a population's variance.

SelectKBest

- Uses statistical tests like chi-squared test, ANOVA F-test or mutual information score to score and rank the features based on their relationship with the output variable.
- Then, it selects the K features with the highest scores to be included in the final feature subset.

Mutual_info_classif:

- This method basically utilizes mutual information.

- It calculates the mutual information value for each of the independent variables with respect to the dependent variable and selects the ones which have the most information gain.
- In other words, it basically measures the dependency of features with the target value.
- A higher score means more dependent variables.

What is the function of Mutual_info_classif?

- Estimate mutual information for a discrete target variable.
- Mutual information (MI) between two random variables is a non-negative value, which measures the dependency between the variables.

train_test_split:

- The train_test_split function shuffles the dataset and then splits it.
- The test_size parameter determines the proportion of the original dataset to include in the test split.
- In this case, we've set it to 0.2, meaning 20% of the data will be used for the test set & the remaining 80% for the training set.

Ensemble Learning:

- Ensemble learning is an approach in which two or more models are fitted to the same data, and the predictions of each model are combined.
- Ensemble learning aims to achieve better performance with the ensemble of models than with any individual model.

What is ensemble used for?

- Ensemble methods are techniques that create multiple models and then combine them to produce improved results.
- Ensemble methods in machine learning usually produce more accurate solutions than a single model would.

Where is ensemble learning used?

- In statistics & Machine Learning, ensemble methods use multiple learning algorithms to obtain better predictive performance than could be obtained from any of the constituent learning algorithms alone.

sklearn ensemble:

- Ensembles: Gradient boosting, random forests, bagging, voting, stacking.
- Ensemble methods combine the predictions of several base estimators built with a given learning algorithm in order to improve generalizability/robustness over a single estimator.

- The sklearn.ensemble module includes two averaging algorithms based on randomized decision trees: the RandomForest algorithm and the Extra-Trees method.

Advantages of Ensemble Algorithm:

- Ensemble is a proven method for improving the accuracy of the model and works in most of the cases.
- Ensemble makes the model more robust and stable thus ensuring decent performance on the test cases in most scenarios.
- We can use ensemble to capture linear and simple as well nonlinear complex relationships in the data.
- This can be done by using two different models and forming an ensemble of two.

Disadvantages of Ensemble Algorithm:

- Ensemble reduces the model interpret-ability and makes it very difficult to draw any crucial business insights at the end.
- It is time-consuming and thus might not be the best idea for real-time applications.
- The selection of models for creating an ensemble is an art which is really hard to master.

Random forest :

- It's a supervised learning algorithm which is used for both classification as well as regression.
- Random forest algorithm creates decision trees on data samples and then gets the prediction from each of them and finally selects the best solution by means of voting.

Is Random Forest an ensemble model?

- Random forest is an ensemble machine learning algorithm.
- It is perhaps the most popular and widely used machine learning algorithm given its good or excellent performance across a wide range of classification and regression predictive modeling problems.

How does the Random forest algorithm work?

- It works in four steps:
- 1. Select the prediction result with the most votes as the final prediction & Select random samples from a given dataset.
- 2. Construct a decision tree for each sample and get a prediction result from each decision tree.
- 3. Perform a vote for each predicted result.
- 4. Select the prediction result with the most votes as the final prediction.

Advantages of Random Forest:

- Random forests is considered as a highly accurate and robust method because of the number of decision trees participating in the process.
- It does not suffer from the overfitting problem. The main reason is that it takes the average of all the predictions, which cancels out the biases.
- The algorithm can be used in both classification and regression problems.
- Random forests can also handle missing values. There are two ways to handle these: using median values to replace continuous variables, and computing the proximity-weighted average of missing values.
- We can get the relative feature importance, which helps in selecting the most contributing features for the classifier.

Disadvantages of Random Forest:

- Random forests is slow in generating predictions because it has multiple decision trees. Whenever it makes a prediction, all the trees in the forest have to make a prediction for the same given input and then perform voting on it. This whole process is time-consuming.
- The model is difficult to interpret compared to a decision tree, where you can easily make a decision by following the path in the tree.

sklearn.metrics:

- The sklearn.metrics module implements several loss, score, and utility functions to measure classification performance.
- Some metrics might require probability estimates of the positive class, confidence values or binary decisions values.

precision_score:

- The precision is the ratio tp/(tp + fp) where tp is the number of true positives and fp the number of false positives.
- The precision is intuitively the ability of the classifier not to label as positive a sample that is negative.

recall_score:

- The recall is the ratio tp/(tp + fn) where tp is the number of true positives and fn the number of false negatives.
- The recall is intuitively the ability of the classifier to find all the positive samples.

f1_score:

- The F1 score can be interpreted as a harmonic mean of the precision and recall, where an F1 score reaches its best value at 1 and worst score at 0.
- The relative contribution of precision and recall to the F1 score are equal.
- The formula for the F1 score is: F1 = 2 * (precision * recall)/(precision + recall).

roc_auc_score:

- roc_auc_score is defined as the area under the ROC curve, which is the curve having False Positive Rate(FPR) on the x-axis & True Positive Rate(TPR) on the y-axis at all classification thresholds.
- But it's impossible to calculate FPR and TPR for regression methods, so we cannot take this road.

What is the difference between ROC and AUC score?

- ROC(Receiver Operating Characteristic Curve) is a probability curve and AUC(Area Under the Curve) represents the degree or measure of separability.
- It tells how much the model is capable of distinguishing between classes.
- **Higher the AUC, the better the model is at predicting 0 classes as 0 and 1 classes as 1.**

accuracy_score:

- The accuracy_score function of the sklearn.metrics package package assigns subset accuracy in multi-label classification & calculates the accuracy score for a set of predicted labels against the true labels.
- Accuracy describes the model's behaviour across all classes.
- Informally, It's the fraction of predictions our model got right.
- **Formally, accuracy has the following definition: Accuracy = Number of correct predictions/Total number of predictions.**

random_state:

- The random_state parameter determines whether multiple calls to fit (for estimators) or to split (for CV splitters) will produce the same results, according to these rules: If an integer is passed, calling fit or split multiple times always yields the same results.
- if a fixed value is assigned like random_state = 0 or 1 or 42 or any other integer then no matter how many times you execute your code the result would be the same - i.e, same values in train & test datasets.
- Random state ensures that the splits that you generate are reproducible.

In [91]:
```
from sklearn.feature_selection import chi2
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import mutual_info_classif
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import precision_score, recall_score, f1_score, roc_auc_score, accuracy_score

feature_selection_methods = [
    ('chi2', chi2),  # chi2
    ('mutual_info_classif', mutual_info_classif),  # mutual_info_classif
]

X_train, X_test, y_train, y_test = train_test_split(X_min_max, y, test_size=0.2, random_state=42)

best_selection_methods = {
    'chi2': {"Accuracy": 0, "Precision": 0, "Recall":0,"f1":0,"ROC_AUC": 0,"Feature_number":0,"Features":[]},
    'mutual_info_classif': {"Accuracy": 0, "Precision": 0, "Recall":0,"f1":0,"ROC_AUC": 0,"Feature_number":0,'
    'lasso': {"Accuracy": 0, "Precision": 0, "Recall":0,"f1":0,"ROC_AUC": 0,"Feature_number":0,"Features":[],
    'PCA': {"Accuracy": 0, "Precision": 0, "Recall":0,"f1":0,"ROC_AUC": 0,"Feature_number":0,"Features":[], "C
    'RFE': {"Accuracy": 0, "Precision": 0, "Recall":0,"f1":0,"ROC_AUC": 0,"Feature_number":0,"Features":[], "C
    'Stacking': {"Accuracy": 0, "Precision": 0, "Recall":0,"f1":0,"ROC_AUC": 0,"Feature_number":0,"Features":[
    'Max Voting': {"Accuracy": 0, "Precision": 0, "Recall":0,"f1":0,"ROC_AUC": 0,"Feature_number":0,"Features'
}

def is_better_method(name, accuracy, precision, recall, f1, roc_auc, feature_num, selected_features, classifie
    if accuracy > best_selection_methods[name]['Accuracy']:
        best_selection_methods[name]['Accuracy'] = accuracy
        best_selection_methods[name]['Precision'] = precision
        best_selection_methods[name]['Recall'] = recall
        best_selection_methods[name]['f1'] = f1
        best_selection_methods[name]['ROC_AUC'] = roc_auc
        best_selection_methods[name]['Feature_number'] = feature_num
        best_selection_methods[name]['Features'] = selected_features

        if name == "PCA" or name == "lasso" or name == "RFE" or name == "Stacking" or name == "Max Voting":
            best_selection_methods[name]['Classifier'] = classifier_name

def calculate_scores(y_test,y_pred, X_test_selected , method):
    precision = precision_score(y_test, y_pred)
    recall = recall_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred)
    roc_auc = roc_auc_score(y_test, method.predict_proba(X_test_selected)[:, 1])
    accuracy = accuracy_score(y_test, y_pred)
    return precision, recall, f1, roc_auc, accuracy

for i in range(1,20):
    for name, selection_model in feature_selection_methods:
        method = SelectKBest(selection_model, k=i)
        X_selected = method.fit_transform(X_train, y_train)
        selected_features = X.columns[method.get_support()]
        print(f"Selected Features using {name}: {selected_features}")
        print(f"Number of Features: {len(selected_features)}")

        RFC = RandomForestClassifier(random_state=42)
        RFC.fit(X_selected, y_train)

        X_test_selected = method.transform(X_test) # update selected features for the test data
        y_pred = RFC.predict(X_test_selected)

        precision, recall, f1, roc_auc, accuracy = calculate_scores(y_test, y_pred, X_test_selected, RFC)

        is_better_method(name, accuracy, precision, recall, f1, roc_auc, len(selected_features), selected_feat

        print(f"Precision: {precision:.4f}, Recall: {recall:.4f}, F1-score: {f1:.4f}")
        print(f"ROC AUC: {roc_auc:.4f}, Accuracy: {accuracy:.4f}")
        print("-" * 50)
```

```
Selected Features using chi2: Index(['Obesity'], dtype='object')
Number of Features: 1
Precision: 0.4982, Recall: 0.4956, F1-score: 0.4969
ROC AUC: 0.4960, Accuracy: 0.4960
--------------------------------------------------
Selected Features using mutual_info_classif: Index(['Exercise Hours Per Week'], dtype='object')
Number of Features: 1
Precision: 0.6679, Recall: 0.7726, F1-score: 0.7165
ROC AUC: 0.6872, Accuracy: 0.6929
--------------------------------------------------
Selected Features using chi2: Index(['Diabetes', 'Obesity'], dtype='object')
Number of Features: 2
Precision: 0.4910, Recall: 0.3133, F1-score: 0.3825
ROC AUC: 0.4940, Accuracy: 0.4920
--------------------------------------------------
Selected Features using mutual_info_classif: Index(['Exercise Hours Per Week', 'Sedentary Hours Per Day'],
dtype='object')
Number of Features: 2
Precision: 0.6940, Recall: 0.7708, F1-score: 0.7304
```

METHOD - 2: Lasso (Least Absolute Shrinkage and Selection Operator)

- In statistics and ML, lasso (least absolute shrinkage and selection operator; also Lasso or LASSO) is a regression analysis method that performs both variable selection and regularization in order to enhance the prediction accuracy and interpretability of the resulting statistical model. (or)

1. LASSO regression, also known as L1 regularization: A popular technique used in statistical modeling & ML to estimate the relationships between variables and make predictions.
2. LASSO stands for Least Absolute Shrinkage and Selection Operator.

Lasso regression classification?

- Lasso regression is a classification algorithm that uses shrinkage in simple and sparse models(i.e model with fewer parameters).
- In Shrinkage, data values are shrunk towards a central point like the mean.

When can you use Lasso regression?

- We use lasso regression when we have large number of predictor variables.

- The equation of LASSO is similar to ridge regression & looks like as given below. Here the objective is as follows: **If λ = 0: We get same coefficients as linear regression & If λ = vary large: All coefficients are shriked towards zero.**

KNN:

- KNN/K Nearest Neighbor is a Machine Learning algorithm that uses the similarity between our data to make classifications (supervised machine learning) or clustering (unsupervised machine learning).
- With KNN we can have a certain set of data and from it draw patterns that can classify or group our data.
- **The concept of neighborhood depends on the idea that those close to us tend to be more like us.**

Classifier:

- A classifier in machine learning is an algorithm that automatically orders or categorizes data into one or more of a set of "**classes**".
- Also sometimes called a measure word or counter word.
- One of the most common examples is an email classifier that scans emails to filter them by class label: Spam or Not Spam.
- Also classifier is an algorithm that can classify data into labeled classes.
- The simplest classifier: K-NN algorithm.

What is the principle of classifier?

- A general guiding principle for classification is the convergence of evidence.
- This means that in one object, or group of objects, various properties con- verge or coincide, in contrast to other objects. - This points to the individuality of that object and reflects its system character.

Why are classifiers important?

- These classifiers can determine the probability of an input fitting into one or more categories.
- In multiple category scenarios, the algorithm reviews the probability that a data point fits into each classification.

KNeighborsClassifier:

- KNeighborsClassifier is a supervised learning algorithm that makes classifications based on data neighbors.
- Changing the number K of neighbors can change the classification, ie it is a slightly sensitive parameter and should be chosen with caution!
- When we have less scattered data and few outliers , KNeighborsClassifier shines.
- kNN classifier identifies the class of a data point using the majority voting principle.
- If k is set to 5, the classes of 5 nearest points are examined & Prediction is done according to the predominant class.
- Similarly, kNN regression takes the mean value of 5 nearest locations.

Advantages of KNN:

- Quick calculation time.
- Simple algorithm – to interpret.
- Versatile – useful for regression and classification.
- High accuracy – you do not need to compare with better-supervised learning models.
- No assumptions about data – no need to make additional assumptions, tune several parameters, or build a model. This makes it crucial in nonlinear data case.

Disadvantages of KNN:

- Accuracy depends on the quality of the data.
- With large data, the prediction stage might be slow.
- Sensitive to the scale of the data and irrelevant features.
- Require high memory – need to store all of the training data.
- Given that it stores all of the training, it can be computationally expensive.

Summary of KNN Algorithm:

- K is a positive integer.
- With a new sample, you have to specify K.
- K is selected from database closest to the new sample.
- KNN doesn’t learn any model.
- KNN makes predictions using the similarity between an input sample and each training instance.

KNN is a great place to start when first learning to build models based on different data sets - Data set with a lot of different points & accurate information is your best place, to begin with KNN.

We should Keep these 3 points in mind:

- 1.A data set with lots of different points and labelled data is the ideal to use.
- 2.The best languages to use with KNN are R & python.
- 3.To find the most accurate results from your data set, you need to learn the correct practices for using this algorithm.

Now we're Using KNeighborsClassifier

In [92]:
```
from sklearn.linear_model import Lasso
from sklearn.neighbors import KNeighborsClassifier

lasso = Lasso(alpha=0.001)
lasso.fit(X_min_max, y) # y is the target (Heart Attack Risk)

non_zero_indices = lasso.coef_ != 0

selected_features_lasso = X_min_max.columns[non_zero_indices]
X_lasso = X_min_max[selected_features_lasso]

X_train_lasso, X_test_lasso, y_train_lasso, y_test_lasso = train_test_split(X_lasso, y, test_size=0.2, random_

def calculate_lasso(classifier, classifier_name):
    classifier.fit(X_train_lasso, y_train_lasso)

    y_pred_lasso = classifier.predict(X_test_lasso)

    y_pred_proba = classifier.predict_proba(X_test_lasso)[:, 1]

    precision = precision_score(y_test_lasso, y_pred_lasso)
    recall = recall_score(y_test_lasso, y_pred_lasso)
    f1 = f1_score(y_test_lasso, y_pred_lasso)
    roc_auc = roc_auc_score(y_test_lasso, y_pred_proba)
    accuracy = accuracy_score(y_test_lasso, y_pred_lasso)

    selected_features_names = X_min_max.columns[non_zero_indices].tolist()

    print(f"Classifier: {classifier_name}")
    print(f"Precision: {precision:.4f}, Recall: {recall:.4f}, F1-score: {f1:.4f}")
    print(f"ROC AUC: {roc_auc:.4f}, Accuracy: {accuracy:.4f}")
    print("-" * 50)

    is_better_method("lasso", accuracy, precision, recall, f1, roc_auc, len(selected_features_lasso), selectec


calculate_lasso(classifier=KNeighborsClassifier(),classifier_name="KNeighborsClassifier")
```

```
Classifier: KNeighborsClassifier
Precision: 0.5733, Recall: 0.6159, F1-score: 0.5939
ROC AUC: 0.6065, Accuracy: 0.5769
--------------------------------------------------
```

Using RandomForestClassifier

In [93]:
```
calculate_lasso(classifier=RandomForestClassifier(),classifier_name="RandomForestClassifier")
```

```
Classifier: RandomForestClassifier
Precision: 0.8371, Recall: 0.6867, F1-score: 0.7545
ROC AUC: 0.8190, Accuracy: 0.7756
--------------------------------------------------
```

DECISION TREE:

- Decision Trees are a type of Supervised Machine Learning Algorithm, where its a diagram or chart that people use to determine a course of action or show a statistical probability.
- Each branch of the decision tree represents a possible decision, outcome, or reaction.

Why do we use Decision Tree?

- The goal of using a Decision Tree is to create a training model that can use to predict the class or value of the target variable by learning simple decision rules inferred from prior data(training data).
- In Decision Trees, for predicting a class label for a record we start from the root of the tree.

Where is decision tree used?

- Decision trees are commonly used in operations research, specifically in decision analysis, to help identify a strategy most likely to reach a goal, but are also a popular tool in machine learning.

How decision tree is used for classification?

- Decision tree builds classification or regression models in the form of a tree structure.
- It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed.
- Decision trees can handle both categorical and numerical data.

What is the purpose of a decision tree?

- Decision trees help you to evaluate our options.
- Decision Trees are excellent tools for helping you to choose between several courses of action.
- They provide a highly effective structure within which you can lay out options and investigate the possible outcomes of choosing those options.

Why are decision trees bad?

- Among the major decision tree disadvantages are its complexity.
- Computing probabilities of different possible branches, determining the best split of each node & selecting optimal combining weights to prune algorithms contained in the decision tree are complicated tasks that require much expertise and experience.

Maximum Depth:

- Max Depth. Controls the maximum depth of the tree that will be created.
- It can also be described as the length of the longest path from the tree root to a leaf.
- The root node is considered to have a depth of 0.

What happens if the maximum depth decreases?

- The maximum depth that you allow the tree to grow to.
- But if you set it too low, that is not good as well; then you might be giving the decision tree too little flexibility to capture the patterns and interactions in the training data.

Advantages of decision trees:

- Compared to other algorithms decision trees requires less effort for data preparation during pre-processing.
- A decision tree does not require normalization of data.
- A decision tree does not require scaling of data as well.
- Missing values in the data also do NOT affect the process of building a decision tree to any considerable extent.
- A Decision tree model is very intuitive and easy to explain to technical teams as well as stakeholders.

Disadvantages of decision trees:

- A small change in the data can cause a large change in the structure of the decision tree causing instability.
- For a Decision tree sometimes calculation can go far more complex compared to other algorithms.
- Decision tree often involves higher time to train the model.

- Decision tree training is relatively expensive as the complexity and time has taken are more.
- The Decision Tree algorithm is inadequate for applying regression and predicting continuous values.
- They are often relatively inaccurate.
- Many other predictors perform better with similar data.

Decision Tree Classifier:

- A Supervised Machine learning algorithm that is used for supervised + classification problems.
- Under the hood in the decision tree, each node asks a True or False question about one of the features and moves left or right with respect to the decision.
- Also a class capable of performing multi-class classification on a dataset.
- In case that there are multiple classes with the same and highest probability, the classifier will predict the class with the lowest index amongst those classes.

Why is decision tree classifier used?

- Decision trees are used to solve classification problems and categorize objects depending on their learning features.
- They can also be used for regression problems or as a method to predict continuous outcomes from unforeseen data.

Where is decision tree classifier used?

- The algorithm can be thought of as a graphical tree-like structure that uses various tuned parameters to predict the results. - The decision trees apply a top-down approach to the dataset that is fed during training.

What are the 4 types of decision tree?

- There are 4 popular types of decision tree algorithms: ID3, CART (Classification and Regression Trees), Chi-Square & Reduction in Variance.

Now we're Using DecisionTreeClassifier

In [94]:
```python
from sklearn.tree import DecisionTreeClassifier
calculate_lasso(classifier=DecisionTreeClassifier(),classifier_name="DecisionTreeClassifier")
```

```
Classifier: DecisionTreeClassifier
Precision: 0.6538, Recall: 0.7788, F1-score: 0.7108
ROC AUC: 0.6813, Accuracy: 0.6818
--------------------------------------------------
```

Gradient Boosting:

- A machine learning technique based on boosting in a functional space, where the target is pseudo-residuals rather than the typical residuals used in traditional boosting.
- Gradient Boosting is a powerful technique that can be used for both classification and regression problems in ML.
- It is more accurate than many other algorithms and with cutting-edge methods like bagging, random forest & decision tree, accurate results are available.
- Also, this is one of the best algorithms to handle larger datasets and compute with the weak learners at least loss.
- In classification problems, the goal is to predict which category a data point belongs to.
- For example, we might want to predict whether an email is spam or not spam.

Why choose gradient boosting?

- Generally more accurate compare to other modes, train faster especially on larger datasets, most of them provide support handling categorical features, some of them handle missing values natively.

What is the idea behind of gradient boosting?

- Gradient boosting is a greedy algorithm and can overfit a training dataset quickly.
- It can benefit from regularization methods that penalize various parts of the algorithm & generally improve the performance of the algorithm by reducing overfitting.

What are the limitations of gradient boosting?

- Training the model may require more time and resources compared to simpler algorithms.
- In the case of complex models or high learning rates, gradient descent can be prone to the overfitting problem.
- Gradient descent is often considered a less interpretable or black box model.

Why is gradient boosting so slow?

- In gradient-boosting, the algorithm is a sequential algorithm.
- It requires the N-1 trees to have been fit to be able to fit the tree at stage N.
- Therefore, the algorithm is quite computationally expensive.

Gradient Boosting Classifier:

- Gradient Boosting is a functional gradient algorithm that repeatedly selects a function that leads in the direction of a weak hypothesis or negative gradient so that it can minimize a loss function.
- **Gradient boosting classifier combines several weak learning models to produce a powerful predicting model.**

Why is gradient boosting classifier good?

- Often provides predictive accuracy that cannot be trumped.
- Lots of flexibility - can optimize on different loss functions and provides several hyper parameter tuning options that make the function fit very flexible.

In [95]:
```
from sklearn.ensemble import GradientBoostingClassifier
calculate_lasso(classifier=GradientBoostingClassifier(),classifier_name="GradientBoostingClassifier")
```

```
Classifier: GradientBoostingClassifier
Precision: 0.5728, Recall: 0.5708, F1-score: 0.5718
ROC AUC: 0.5872, Accuracy: 0.5707
--------------------------------------------------
```

Naive Bayes:

- **It calculates probabilities independently for each class based on conditions & without conditions & then predicts outcomes.**
- A machine learning algorithm we use to solve classification problems.
- It is based on the Bayes Theorem.
- It is one of the simplest yet powerful ML algorithms in use and finds applications in many industries.
- It is simple and easy to implement.
- It doesn't require as much training data.
- It handles both continuous and discrete data.
- It is highly scalable with the number of predictors and data points.

Naive Bayes classifier:

- A supervised machine learning algorithm, which is used for classification tasks, like text classification.
- It's also part of a family of generative learning algorithms, meaning that it seeks to model the distribution of inputs of a given class or category.
- The Naive Bayes classifier assumes that all features in the input data are independent of each other, which is often not true in real-world scenarios.

- However, despite this simplifying assumption, the naive Bayes classifier is widely used because of its efficiency & good performance in many real-world applications.
- It offers simplicity, efficiency and effectiveness, making it suitable for various applications.
- Some best examples are sentimental analysis, recommendation systems, classifying new articles & spam filtration.
- Naive Bayes classifier is the fast, accurate and reliable algorithm.
- Also have high accuracy and speed on large datasets.
- Naive Bayes classifier assumes that the effect of a particular feature in a class is independent of other features.

Now we're Using Naive Bayes Classifier

In [96]:
```
from sklearn.naive_bayes import GaussianNB
calculate_lasso(classifier=GaussianNB(),classifier_name="GaussianNB")
```

```
Classifier: GaussianNB
Precision: 0.5005, Recall: 0.4788, F1-score: 0.4894
ROC AUC: 0.5004, Accuracy: 0.4982
--------------------------------------------------
```

METHOD - 3. PCA (Principal Component Analysis):

- PCA, is a dimensionality reduction method that is often used to reduce the dimensionality of large data sets, by transforming a large set of variables into a smaller one that still contains most of the information in the large set.
- Principal components analysis (PCA) is a standard tool in multivariate data analysis to reduce the number of dimensions, while retaining as much as possible of the data's variation.

What are the benefits of PCA analysis?

- Improve Algorithm Runtime.
- Improve Classification Accuracy.
- Visualization.
- Reduce Noise in Data.
- Feature Selection.

Applications of PCA:

- PCA is used to visualize multidimensional data.
- It is used to reduce the number of dimensions in healthcare data.

- PCA can help resize an image.
- It can be used in finance to analyze stock data and forecast returns.

Why is PCA preferable?

- The principal component method of factor analysis will help us.
- If you want to categorize the dependent & independent variables in your data, this algorithm will be our choice of consideration.
- Also, if you want to eliminate the noise components in your dimension analysis, PCA is the best computation method.

Logistic Regression:

- A data analysis technique that uses mathematics to find the relationships between two data factors.
- Logistic regression is not a stable learning algorithm.
- It then uses this relationship to predict the value of one of those factors based on the other.
- The prediction usually has a finite number of outcomes, like yes or no.
- It is named 'Logistic Regression' because its underlying technique is quite the same as Linear Regression.
- The term “Logistic” is taken from the Logit function that is used in this method of classification.
- Example of logistic regression can be to find if a person will default their credit card payment or not.

Types of logistic regression:

- There are three main types of logistic regression: binary, multinomial and ordinal.
- They differ in execution and theory.
- Binary regression deals with two possible values, essentially: yes or no.
- Multinomial logistic regression deals with three or more values.
- Ordinal Logistic Regression used for ordered categorical outcomes

Key advantages of logistic regression:

- Easier to implement machine learning methods: A machine learning model can be effectively set up with the help of training and testing.
- Suitable for linearly separable datasets: A linearly separable dataset refers to a graph where a straight line separates the two data classes.

What is the concept of regression?

- Regression is a statistical method used in finance, investing, and other disciplines that attempts to determine the strength & character of the relationship between one dependent variable (usually denoted by Y) and a series of other variables (known as independent variables).

Why is logistic regression very popular?

- Logistic regression is famous because it can convert the values of logits (logodds), which can range from -infinity to +infinity to a range between 0 and 1.
- As logistic functions output the probability of occurrence of an event, it can be applied to many real-life scenario.

In [97]:
```
from sklearn.decomposition import PCA
from sklearn.linear_model import LogisticRegression

pca = PCA(n_components=12) # reduction of the features to 12.

pca.fit(X_min_max)

data_pca = pca.transform(X_min_max)

X_train_pca, X_test_pca, y_train_pca, y_test_pca = train_test_split(data_pca, y, test_size=0.2, random_state=4


def calculate_pca(classifier, classifier_name):
    classifier.fit(X_train_pca, y_train_pca)
    y_pred_pca = classifier.predict(X_test_pca)

    accuracy = accuracy_score(y_test_pca, y_pred_pca)
    precision = precision_score(y_test_pca, y_pred_pca)
    recall = recall_score(y_test_pca, y_pred_pca)
    f1 = f1_score(y_test_pca, y_pred_pca)
    roc_auc = roc_auc_score(y_test_pca, y_pred_pca)


    print(f"Classifier: {classifier_name}")
    print(f"Precision: {precision:.4f}, Recall: {recall:.4f}, F1-score: {f1:.4f}")
    print(f"ROC AUC: {roc_auc:.4f}, Accuracy: {accuracy:.4f}")
    print("-" * 50)

    is_better_method("PCA", accuracy, precision, recall, f1, roc_auc, 0 , [], classifier_name)


calculate_pca(classifier=LogisticRegression() , classifier_name="LogisticRegression")
```

```
Classifier: LogisticRegression
Precision: 0.4960, Recall: 0.4982, F1-score: 0.4971
ROC AUC: 0.4938, Accuracy: 0.4938
--------------------------------------------------
```

Using RandomForestClassifier

```
In [98]: calculate_pca(classifier=RandomForestClassifier(n_estimators=250) , classifier_name="RandomForestClassifier")
```

```
Classifier: RandomForestClassifier
Precision: 0.7710, Recall: 0.7239, F1-score: 0.7467
ROC AUC: 0.7535, Accuracy: 0.7533
--------------------------------------------------
```

Support Vector Machine (SVM)

- A type of supervised learning algorithm used in machine learning to solve classification, outliers detection & regression tasks.
- SVMs are particularly good at solving binary classification problems, which require classifying the elements of a data set into two groups.
- But generally, they work best in classification problems.
- A SVMe is a nonparametric model that takes more time for training, but the algorithm's learning is not limited.
- In the case of nonlinear data, the kernel function can be used in SVM to solve the data patterns.

Why is it called support vector machine?

- SVM only uses the objects (samples) on the edges of the margin (called support vectors) to separate objects (samples) rather than using the differences in class means.
- Since the separating hyperplane is supported (defined) by the vectors (data points) nearest the margin, so the algorithm is called SVM.

What is the main objective of SVM?

- The objective of the SVM algorithm is to find a hyperplane that, to the best degree possible, separates data points of one class from those of another class.
- “Best” is defined as the hyperplane with the largest margin between the two classes, represented by plus versus minus in the figure.

Real-Life Applications:

- Face detection – SVMc classify parts of the image as a face and non-face and create a square boundary around the face.
- Text and hypertext categorization – SVMs allow Text and hypertext categorization for both inductive and transductive models.

Advantages:

- SVM's are very good when we have no idea on the data.
- Works well with even unstructured and semi structured data like text, Images and trees.
- The kernel trick is real strength of SVM: With an appropriate kernel function, we can solve any complex problem.
- SVM Classifiers offer good accuracy and perform faster prediction compared to Naive Bayes algorithm.
- They also use less memory because they use a subset of training points in the decision phase.
- SVM works well with a clear margin of separation and with high dimensional space.

Why is SVM classifier used?

- It can handle both classification & regression on linear and non-linear data.
- Another reason we use SVMs is because they can find complex relationships between your data without you needing to do a lot of transformations on your own.

In [99]:
```python
from sklearn.svm import SVC
calculate_pca(classifier=SVC(kernel='sigmoid',cache_size=1000), classifier_name="SVM")
```

```
Classifier: SVM
Precision: 0.4871, Recall: 0.4681, F1-score: 0.4774
ROC AUC: 0.4854, Accuracy: 0.4853
--------------------------------------------------
```

Using DecisionTreeClassifier

In [100]:
```python
calculate_pca(classifier=DecisionTreeClassifier(),classifier_name="DecisionTreeClassifier")
```

```
Classifier: DecisionTreeClassifier
Precision: 0.6672, Recall: 0.7788, F1-score: 0.7187
ROC AUC: 0.6934, Accuracy: 0.6938
--------------------------------------------------
```

Using GradientBoostingClassifier

In [101]:
```
calculate_pca(classifier=GradientBoostingClassifier(n_estimators=150),classifier_name="GradientBoostingClassif
```

```
Classifier: GradientBoostingClassifier
Precision: 0.5723, Recall: 0.5708, F1-score: 0.5716
ROC AUC: 0.5702, Accuracy: 0.5702
--------------------------------------------------
```

Using Naive Bayes Classifier

In [102]:
```
calculate_pca(classifier=GaussianNB(),classifier_name="GaussianNB")
```

```
Classifier: GaussianNB
Precision: 0.5143, Recall: 0.5257, F1-score: 0.5199
ROC AUC: 0.5124, Accuracy: 0.5124
--------------------------------------------------
```

METHOD - 4. RFE (Recursive Feature Elimination):

- Recursive feature elimination (RFE) is a feature selection method that fits a model & removes the weakest feature (or features) until the specified number of features is reached.
- Recursive Feature Elimination offers a compelling solution and RFE iteratively removes less important features, creating a subset that maximizes predictive accuracy.
- By leveraging a machine learning algorithm and an importance-ranking metric, RFE evaluates each feature's impact on model performance.
- An iterative method that trains a model and progressively removes the least crucial features based on a chosen metric, typically model accuracy.

RFE method used for?

- Recursive Feature Elimination, is a popular feature selection algorithm.
- RFE is popular because it is easy to configure and use and because it is effective at selecting those features (columns) in a training dataset that are more or most relevant in predicting the target variable.
- RFE is used for feature selection to improve model performance, reduce dimensionality, and enhance interpretability in various machine learning algorithms.

Best Practices for RFE

1. **Choose the Appropriate Number of Features:** It helps to balance model power and complexity by choosing an appropriate number of features. Try different numbers of features and evaluate the model’s performance.
2. **Sets the Number of Cross-Validation Folds:** Cross-validation helps reduce overfitting and improve model generalisation. You should set the number of cross-validation folds based on the size of your dataset and the number of features.
3. **High Dimensional Processing:** Recursive Feature Elimination can handle high-dimensional datasets but can be computationally expensive. Dimensionality reduction techniques such as PCA and LDA can be used before applying RFE.
4. **Dealing with Multicollinearity:** RFE can handle multicollinearity but may not be the best approach. Other techniques, such as PCA and regularisation, can also deal with multicollinearity.
5. **Avoid Overfitting or Underfitting:** RFE can reduce the risk of overfitting by choosing the most important features. However, removing important features can also lead to underfitting. Evaluate the overall performance of the models inside the holdout set to ensure that the models are well-rested and well-fitted.

Advantages of RFE:

- Can handle high-dimensional datasets and identify the most important features.
- Can handle interactions between features and is suitable for complex datasets.
- Can be used with any supervised learning algorithm.

Limitations of RFE:

- Can be computationally expensive for large datasets.
- May not be the best approach for datasets with many correlated features.
- May not work well with noisy or irrelevant features.

What is recursive feature addition in Sklearn?

- Recursive feature addition (RFA) is a forward feature selection process.
- This technique begins by building a model on the entire set of variables & computing an importance score for each variable. - Features are ranked by the model's coef_ or feature_importances_ attributes.

Real-World Applications of Recursive Feature Elimination:

- **1. Bioinformatics:** RFE selects genes for cancer diagnosis and prognosis. By choosing the most meaningful genes, RFE can help improve the accuracy of cancer diagnosis and provide patients with personalised treatment plans.
- **2. Image Processing:** RFE has been used to select image classification and recognition features. By choosing the most informative features, RFE can help improve the accuracy of image classification and recognition systems in various applications, such as autonomous driving and security systems.

- **3. Finance:** RFE has been used in finance to select credit scoring and fraud detection features. By selecting the most relevant features, RFE can help improve the accuracy of credit scoring models and detect fraudulent activities in financial transactions.
- **4. Marketing:** RFE has been used to select customer segmentation and recommendation system features. By selecting the most relevant features, RFE can help identify customer segments and provide personalised recommendations, improving customer satisfaction and increasing sales.

What is recursive feature elimination in logistic regression?

- Recursive Feature Elimination in logistic regression selects the most significant features for the logistic regression model, improving interpretability & predictive accuracy.

In [103]:
```
from sklearn.feature_selection import RFE

X_train, X_test, y_train, y_test = train_test_split(X_min_max, y, test_size=0.2, random_state=42)

def calculate_rfe(classifier,classifier_name):

    rfe = RFE(estimator=classifier,n_features_to_select=15)

    rfe.fit(X_train, y_train)

    selected_features = rfe.support_

    selected_columns = X_train.columns[selected_features]

    # Use the selected columns to filter X_train and X_test
    X_train_rfe = X_train[selected_columns]
    X_test_rfe = X_test[selected_columns]

    classifier.fit(X_train_rfe, y_train)

    y_pred_rfe = classifier.predict(X_test_rfe)

    accuracy = accuracy_score(y_test, y_pred_rfe)
    precision = precision_score(y_test, y_pred_rfe)
    recall = recall_score(y_test, y_pred_rfe)
    f1 = f1_score(y_test, y_pred_rfe)
    roc_auc = roc_auc_score(y_test, y_pred_rfe)

    print(f"Classifier: {classifier_name}")
    print(f"Precision: {precision:.4f}, Recall: {recall:.4f}, F1-score: {f1:.4f}")
    print(f"ROC AUC: {roc_auc:.4f}, Accuracy: {accuracy:.4f}")
    print("-" * 50)

    feature_num = 0

    for feature in selected_features:
        if feature == True:
            feature_num +=1

    is_better_method("RFE", accuracy, precision, recall, f1, roc_auc, feature_num , selected_features, classif
calculate_rfe(classifier=LogisticRegression(),classifier_name="LogisticRegression")
```

```
Classifier: LogisticRegression
Precision: 0.4965, Recall: 0.4991, F1-score: 0.4978
ROC AUC: 0.4942, Accuracy: 0.4942
--------------------------------------------------
```

Using DecisionTreeClassifier

In [104]:
```
calculate_rfe(classifier=DecisionTreeClassifier(),classifier_name="DecisionTreeClassifier")
```

```
Classifier: DecisionTreeClassifier
Precision: 0.6679, Recall: 0.7761, F1-score: 0.7180
ROC AUC: 0.6934, Accuracy: 0.6938
--------------------------------------------------
```

Using RandomForestClassifier

In [105]:
```
calculate_rfe(classifier=RandomForestClassifier(),classifier_name="RandomForest")
```

```
Classifier: RandomForest
Precision: 0.8420, Recall: 0.6982, F1-score: 0.7634
ROC AUC: 0.7830, Accuracy: 0.7827
--------------------------------------------------
```

Using GradientBoostingClassifier

In [106]:
```
calculate_rfe(classifier=GradientBoostingClassifier(),classifier_name="GradientBoostingClassifier")
```

```
Classifier: GradientBoostingClassifier
Precision: 0.5517, Recall: 0.5478, F1-score: 0.5497
ROC AUC: 0.5493, Accuracy: 0.5493
--------------------------------------------------
```

METHOD - 5. Stacking

- Stacking is a strong ensemble learning strategy in machine learning that combines the predictions of numerous base models to get a final prediction with better performance.
- It is also known as stacked ensembles or stacked generalization.
- In stacking, the final estimator, also known as a meta-learner or meta-model, is the model that combines the predictions of the base models to make the final predictions.
- After the base models make their individual predictions, the final forecaster learns to effectively combine these predictions to provide an overall prediction.
- The basic idea is to train machine learning algorithms with training dataset & then generate a new dataset with these models.

What is the stacking process?

- After reducing, calibrating, and performing photometry on data, we are finally able to stack.
- Stacking, in short, consists of aligning and literally stacking images of like fields on top of one another to increase flux and depth.

Use of stacking?

- Stacking is an ensemble learning technique that uses predictions for multiple nodes(for example kNN, decision trees, or SVM) to build a new model.
- This final model is used for making predictions on the test dataset.
- Also to predict multiple nodes to build a new model and improve model performance.

Types of stacking?

- 1. Which a unit load is stacked on the top of another (load-on-load).
- 2. The other in which a unit load is stacked on the top of two adjacent unit loads (load-between-loads).

Does stacking reduce Overfitting?

- The stacking algorithm is known to improve the accuracy of the final prediction compared to using a single machine learning algorithm.
- The reason behind this is that the stacking algorithm combines the strengths of several base models, reducing the chances of overfitting.

In [107]:
```
from sklearn.ensemble import StackingClassifier

X_train, X_test, y_train, y_test = train_test_split(X_min_max, y, test_size=0.2, random_state=42)

base_models = [
    ('LogisticRegression', LogisticRegression()),
    ('lasso', Lasso()),
    ('DecisionTreeClassifier', DecisionTreeClassifier()),
    ('RandomForestClassifier0', RandomForestClassifier(n_estimators=50)),
    ('RandomForestClassifier1', RandomForestClassifier(n_estimators=75)),
    ('RandomForestClassifier2', RandomForestClassifier(n_estimators=100)),
    ('RandomForestClassifier3', RandomForestClassifier(n_estimators=125)),
    ('RandomForestClassifier4', RandomForestClassifier(n_estimators=150)),
    ('GradientBoostingClassifier', GradientBoostingClassifier()),
    ('KNeighborsClassifier', KNeighborsClassifier())
]

stacked_model = StackingClassifier(estimators = base_models, final_estimator=RandomForestClassifier())

stacked_model.fit(X_train.values, y_train.values) # training the stacked_model

stacked_predictions = stacked_model.predict(X_test.values) # making predictions with stacked_model

accuracy = accuracy_score(y_test, stacked_predictions)
precision = precision_score(y_test, stacked_predictions)
recall = recall_score(y_test, stacked_predictions)
f1 = f1_score(y_test, stacked_predictions)
roc_auc = roc_auc_score(y_test, stacked_predictions)

is_better_method("Stacking",accuracy,precision,recall,f1,roc_auc,0,[], classifier_name="RandomForestClassifier
```

In [108]:
```
def print_best_selection_methods():
    for method in best_selection_methods:

        if method == "PCA" or method == "lasso" or method == "RFE" or method =="Stacking" or method == "Max Vc
            print(f"Classifier: {best_selection_methods[method]['Classifier']}")

        print("Name:{}, Accuracy:{},  Precision:{}, Recall:{}, f1:{}, ROC_AUC:{}, feature_number:{}, Features:
            best_selection_methods[method]['Accuracy'], best_selection_methods[method]['Precision'], best_sele
            best_selection_methods[method]['f1'], best_selection_methods[method]['ROC_AUC'], best_selection_me
            best_selection_methods[method]['Features']
            ))
        print("-"*50)
```

METHOD - 6. Max Voting:

- Max-voting, which is generally used for classification problems, is one of the simplest ways of combining predictions from multiple machine learning algorithms.
- In max-voting, each base model makes a prediction and votes for each sample.
- Only the sample class with the highest votes is included in the final predictive class.

What is max voting formula?

- The Max Voting Algorithm is a straightforward yet powerful technique in machine learning.
- By aggregating predictions from multiple models, it harnesses the collective wisdom to arrive at consensus-based decisions.
- Its simplicity, robustness, and improved accuracy make it a valuable tool in various domains.

Voting Classifier:

- A machine learning model that trains on an ensemble of numerous models and predicts an output (class) based on their highest probability of chosen class as the output.
- Combines different machine learning classifiers and uses a voting rule ('soft' or 'hard') to predict the class labels.
- It balances out the individual weaknesses of models when their performance is almost the same; here we will combine individual classifiers used before to build an ensemble.

Standard Scaler:

- Standardize features by removing the mean and scaling to unit variance.

- Removes the mean and scales the data to the unit variance.
- However, outliers have an influence when calculating the empirical mean and standard deviation, which narrows the range of characteristic values.
- These differences in the initial features can cause problems for many machine learning models.

In [109]:
```
from sklearn.ensemble import VotingClassifier
from sklearn.preprocessing import StandardScaler

standard_scaler = StandardScaler()
X_standard = pd.DataFrame(scaler.fit_transform(X))

x_train, x_test, y_train, y_test = train_test_split(X_standard, y, test_size=0.2, random_state=42)

def calculate_max_voting(estimators):

    max_voting_model = VotingClassifier(estimators=estimators)
    max_voting_model.fit(x_train,y_train)

    max_voting_pred = max_voting_model.predict(x_test)

    accuracy = accuracy_score(y_test, max_voting_pred)
    precision = precision_score(y_test, max_voting_pred)
    recall = recall_score(y_test, max_voting_pred)
    f1 = f1_score(y_test, max_voting_pred)
    roc_auc = roc_auc_score(y_test, max_voting_pred)

    is_better_method("Max Voting", accuracy, precision, recall, f1, roc_auc, 0, [], estimators)

estimators = []

estimators.append(('dtc',DecisionTreeClassifier()))
estimators.append(('gbc',GradientBoostingClassifier(random_state=42)))
estimators.append(('rfc',RandomForestClassifier(n_estimators=125)))
estimators.append(('nb',GaussianNB())) # gaussian naive bayes
estimators.append(('lr',LogisticRegression()))

calculate_max_voting(estimators=estimators)
```

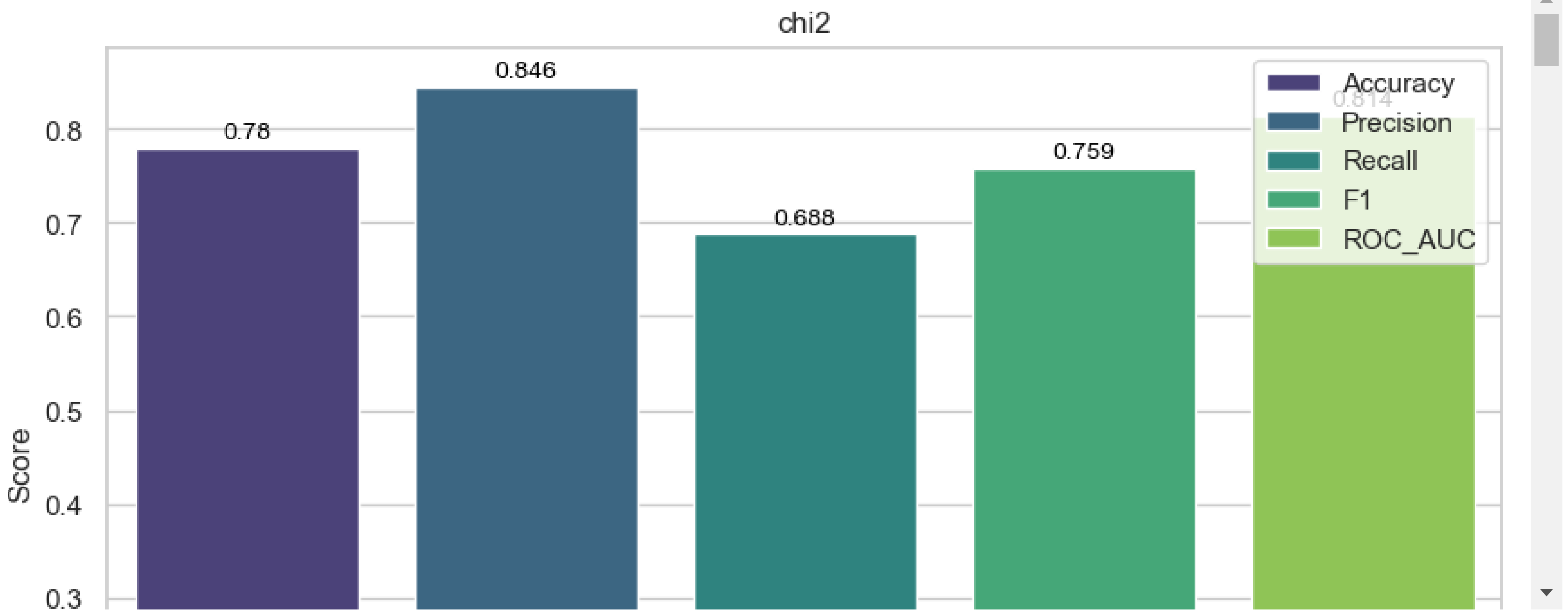
In [110]:
```
print_best_selection_methods()
```

```
Name:chi2, Accuracy:0.7804444444444445,  Precision:0.8456521739130435, Recall:0.6884955752212389, f1:0.75902
43902439023, ROC_AUC:0.8144188527180783, feature_number:18, Features:Index(['Cholesterol', 'Heart Rate', 'Di
abetes', 'Family History', 'Smoking',
       'Obesity', 'Alcohol Consumption', 'Exercise Hours Per Week', 'Diet',
       'Medication Use', 'Stress Level', 'Sedentary Hours Per Day',
       'Triglycerides', 'Physical Activity Days Per Week',
       'Sleep Hours Per Day', 'is_male', 'systolic_pressure',
       'diastolic_pressure'],
      dtype='object')
--------------------------------------------------
Name:mutual_info_classif, Accuracy:0.7808888888888889,  Precision:0.8450704225352113, Recall:0.6902654867256
637, f1:0.7598636142230881, ROC_AUC:0.8188191371681416, feature_number:17, Features:Index(['Age', 'Cholester
ol', 'Heart Rate', 'Family History', 'Smoking',
       'Obesity', 'Alcohol Consumption', 'Exercise Hours Per Week', 'Diet',
       'Medication Use', 'Stress Level', 'Sedentary Hours Per Day',
       'Triglycerides', 'Physical Activity Days Per Week', 'is_male',
       'systolic_pressure', 'diastolic_pressure'],
      dtype='object')
--------------------------------------------------
Classifier: RandomForestClassifier
Name:lasso, Accuracy:0.7755555555555556,  Precision:0.837108953613808, Recall:0.6867256637168142, f1:0.75449
68400583374, ROC_AUC:0.8190332648546144, feature_number:15, Features:[1, 2, 3, 5, 6, 7, 8, 9, 12, 13, 14, 1
5, 16, 17, 18]
--------------------------------------------------
Classifier: RandomForestClassifier
Name:PCA, Accuracy:0.7533333333333333,  Precision:0.7709707822808671, Recall:0.7238938053097345, f1:0.746691
0086718394, ROC_AUC:0.7534647597977243, feature_number:0, Features:[]
--------------------------------------------------
Classifier: RandomForest
Name:RFE, Accuracy:0.7826666666666666,  Precision:0.8420490928495198, Recall:0.6982300884955752, f1:0.763425
2539912918, ROC_AUC:0.783043615676359, feature_number:15, Features:[ True  True  True False  True False Fals
e  True  True  True  True  True
  True  True  True  True False  True  True]
--------------------------------------------------
Classifier: RandomForestClassifier
Name:Stacking, Accuracy:0.816,  Precision:0.9850948509485095, Recall:0.643362831858407, f1:0.778372591006424
1, ROC_AUC:0.8167707016434892, feature_number:0, Features:[]
--------------------------------------------------
Classifier: [('dtc', DecisionTreeClassifier()), ('gbc', GradientBoostingClassifier(random_state=42)), ('rf
c', RandomForestClassifier(n_estimators=125)), ('nb', GaussianNB()), ('lr', LogisticRegression())]
Name:Max Voting, Accuracy:0.6142222222222222,  Precision:0.6175942549371634, Recall:0.6088495575221239, f1:
```

```
0.6131907308377897, ROC_AUC:0.6142462073324905, feature_number:0, Features:[]
--------------------------------------------------
```

Visualization of the Results

In [111]:
```
metrics = {'Accuracy': 0, 'Precision': 0, 'Recall': 0, 'F1': 0, 'ROC_AUC': 0}

def show_plots():
    for method in best_selection_methods:
        metrics['Accuracy'] = best_selection_methods[method]['Accuracy']
        metrics['Precision'] = best_selection_methods[method]['Precision']
        metrics['Recall'] = best_selection_methods[method]['Recall']
        metrics['F1'] = best_selection_methods[method]['f1']
        metrics['ROC_AUC'] = best_selection_methods[method]['ROC_AUC']

        plt.figure(figsize=(10, 6))
        sns.set(style="whitegrid")
        metrics_plot = sns.barplot(x=list(metrics.keys()), y=list(metrics.values()), hue=list(metrics.keys()),
                                   dodge=False, palette="viridis")
        metrics_plot.set_title(method)
        metrics_plot.set_ylabel('Score')

        for index, value in enumerate(metrics.values()):
            metrics_plot.text(index, value + 0.01, str(round(value, 3)), ha='center', color='black', fontsize=

        plt.show()

show_plots()
```


chi2
Score
0.8
0.7
0.6
0.5
0.4
0.3
0.78
0.846
0.688
0.759
0.814
Accuracy
Precision
Recall
F1
ROC_AUC

Comparing the scores of the classifiers

In [112]:
```
results = pd.DataFrame.from_dict(best_selection_methods, orient='index')
print(results)
```

```
                     Accuracy  Precision    Recall        f1   ROC_AUC  \
chi2                 0.780444   0.845652  0.688496  0.759024  0.814419   
mutual_info_classif  0.780889   0.845070  0.690265  0.759864  0.818819   
lasso                0.775556   0.837109  0.686726  0.754497  0.819033   
PCA                  0.753333   0.770971  0.723894  0.746691  0.753465   
RFE                  0.782667   0.842049  0.698230  0.763425  0.783044   
Stacking             0.816000   0.985095  0.643363  0.778373  0.816771   
Max Voting           0.614222   0.617594  0.608850  0.613191  0.614246   

                     Feature_number  \
chi2                             18   
mutual_info_classif              17   
lasso                            15   
PCA                               0   
RFE                              15   
Stacking                          0   
Max Voting                        0   

                                                              Features  \
chi2                 Index(['Cholesterol', 'Heart Rate', 'Diabetes'...   
mutual_info_classif  Index(['Age', 'Cholesterol', 'Heart Rate', 'Fa...   
lasso                [1, 2, 3, 5, 6, 7, 8, 9, 12, 13, 14, 15, 16, 1...   
PCA                                                                 []   
RFE                  [True, True, True, False, True, False, False, ...   
Stacking                                                            []   
Max Voting                                                          []   

                                                            Classifier  
chi2                                                               NaN  
mutual_info_classif                                                NaN  
lasso                                           RandomForestClassifier  
PCA                                             RandomForestClassifier  
RFE                                                       RandomForest  
Stacking                                        RandomForestClassifier  
Max Voting           [(dtc, DecisionTreeClassifier()), (gbc, Gradie...  
```