OBJECTIVES

- Import the dataset into Jupiter Notebook using all required libraries
- Eliminate outliers to clean the data and improve model performance.
- Plot Boxplot before outlier and after outliers
- Perform descriptive statistical analysis to understand the distribution and relationship of features.
- Plot of different graph and chart
- Normalization of features before PCA
- Apply Principal Component Analysis for reduce dimensionality of features
- Models training and testing using 3 algorithms logistic Regression, Support vector machine and Naiive Bayes before using PCA and using PC
- Apply and compare results before and after using Principal Component Analysis (PCA), an unsupervised learning technique
- Visualize the results using Power BI for better understanding and presentation
- Suggestion insights based on dataset and model in health organization for decision making

PREGNANCY RISK HYPERTENSION PREDICTION DATASET

INTRODUCTION

The dataset, named "pregnancy_hypertension_data.csv," contains clinical and demographic information related to pregnant women, with the aim of analysing factors associated with hypertension during pregnancy. It includes 1000 records, each representing an individual patient, and focuses on identifying potential risk factors for hypertension, a critical condition that can impact maternal and fetal health. Dataset Overview The dataset comprises the following variables:

• Age: Age of the patient (in years, ranging from 15 to 45).
• SystolicBP: Systolic blood pressure (in mmHg, ranging from 81.7 to 199.5).
• DiastolicBP: Diastolic blood pressure (in mmHg, ranging from 33.8 to 121.8).
• BMI: Body Mass Index (in kg/m², ranging from 10.6 to 46.7).
• Body Temperature: Body temperature (in Celsius, ranging from 35.2 to 40.0).
• Glucose: Blood glucose level (in mg/dL, ranging from 54.3 to 208.7).
• Outcome_Hypertension: Binary outcome indicating the presence of hypertension ("hypertension" or "no hypertension").

DATASET WITH TOTAL TUPLE 1000

COLUMN NAME	TYPE OF DATA
Age	Discrete data
SystolicBP	Continuous data
DiastolicBP	Continuous data
BMI	Continuous data
Body Temperature	Continuous data
Glucose	Continuous data
Outcome_Hypertension	Categorical data

Aim for Analysis

The primary objective of analysing this dataset is to investigate the relationship between the recorded variables (age, blood pressure, BMI, body temperature, and glucose levels) and the occurrence of hypertension during pregnancy. By exploring these factors, the analysis aims to:

• Identify key risk factors contributing to hypertension.

• Support predictive modelling to assess the likelihood of hypertension based on clinical measurements.

• Provide insights that can inform clinical decision-making and improve maternal health.

Outcomes

This dataset serves as a valuable resource for researchers and healthcare professionals seeking to understand and mitigate the risks of pregnancy-related hypertension.

All tasks were performed

- Using Jupiter Notebook
- Python programming language.
- In machine learning model we use logistic regression, naive Bayes and SVM
- Visualized by using Power Bi

FIRST TO IMPORT ALL NECESSARY LIBRARY

In [53]:
```
import pandas as pd
import numpy  as np
import matplotlib.pyplot as plt
import seaborn as sns
```

TO LOAD AND READ THE FILE OF DATASET

In [54]:
```
df=pd.read_csv('pregnancy_hypertension_data .csv')
```

TO READ DATASET BY FIRST 100 ROW AND LAST 100 ROW

In [55]:
```
df.head(5)
```

Out[55]:

	Age	SystolicBP	DiastolicBP	BMI	BodyTemperature	Glucose	Outcome_Hypertension
0	21	116.4	81.3	26.8	36.9	83.6	no hypertension
1	34	118.6	85.3	23.0	36.8	98.2	no hypertension
2	43	126.8	83.4	18.8	36.8	118.0	no hypertension
3	29	119.8	74.0	25.1	37.2	83.3	no hypertension
4	25	90.5	52.7	24.9	36.6	112.6	no hypertension

In [56]:
```
df.tail(5)
```

Out[56]:

	Age	SystolicBP	DiastolicBP	BMI	BodyTemperature	Glucose	Outcome_Hypertension
995	29	117.9	82.0	23.0	35.6	89.5	no hypertension
996	31	120.4	72.9	30.3	37.4	82.4	hypertension
997	25	110.5	64.2	18.2	37.1	105.3	no hypertension
998	35	112.0	66.0	40.5	37.6	88.5	hypertension
999	40	135.5	90.4	27.6	36.2	82.4	hypertension

TO CHECK COLUMNS LIST

In [57]:
```
df.columns
```

Out[57]:
```
Index(['Age', 'SystolicBP', 'DiastolicBP', 'BMI', 'BodyTemperature', 'Glucose',
       'Outcome_Hypertension'],
      dtype='object')
```

TO CHECK INFORMATION OF DATASET BY DATA TYPES, NULL

In [58]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 7 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   Age                   1000 non-null   int64  
 1   SystolicBP            1000 non-null   float64
 2   DiastolicBP           1000 non-null   float64
 3   BMI                   1000 non-null   float64
 4   BodyTemperature       1000 non-null   float64
 5   Glucose               1000 non-null   float64
 6   Outcome_Hypertension  1000 non-null   object 
dtypes: float64(5), int64(1), object(1)
memory usage: 54.8+ KB
```

TO CHECK SHAPE OF DATASET HERE WE LOOK ON COLUMN AND ROWS COUNT

In [59]:
```
df.shape
```

Out[59]:
```
(1000, 7)
```

TO CHECK FOR MISSING VALUE

In [60]:
```
df.isnull().sum()
```

Out[60]:
```
Age                   0
SystolicBP            0
DiastolicBP           0
BMI                   0
BodyTemperature       0
Glucose               0
Outcome_Hypertension  0
dtype: int64
```

TO DROP THE COLUMN WITH IRREREVANT IN OUR DATASET

In [61]:
```
df.drop(['BodyTemperature','Glucose'],axis=1,errors='ignore',inplace=True)
```

TO CONFIRM REMOVED COLUMNS LIST FROM 7 COLUMNS TO 5 COLUMNS

In [62]:
```
df.columns
```

Out[62]:
```
Index(['Age', 'SystolicBP', 'DiastolicBP', 'BMI', 'Outcome_Hypertension'], dtype
='object')
```

In [63]:
```
df.shape
```

Out[63]:
```
(1000, 5)
```

DESCRIPTIVE ANALYSIS BEFORE CLEANING OF DATA FIRST

In [64]:
```
df.describe()
```

Out[64]:

	Age	SystolicBP	DiastolicBP	BMI
count	1000.000000	1000.000000	1000.000000	1000.000000
mean	30.448000	118.654400	78.129000	24.143800
std	9.243957	12.872041	12.301747	4.547261
min	15.000000	81.700000	33.800000	10.600000
25%	22.000000	110.475000	70.000000	21.000000
50%	31.000000	118.100000	78.100000	23.900000
75%	39.000000	125.400000	86.200000	27.000000
max	45.000000	199.500000	121.800000	46.700000

SUMMARY INTERPRETATION

The dataset includes 1,000 individuals and provides descriptive statistics for four variables: Age, Systolic Blood Pressure (SystolicBP), Diastolic Blood Pressure (DiastolicBP), and Body Mass Index (BMI). The mean, or average, represents the central value of the data. For instance, the mean age is 30.45 years, SystolicBP is 118.65 mmHg, DiastolicBP is 78.13 mmHg, and BMI is 24.14. These averages provide a general understanding of the typical values in the dataset. The median, which is the middle value when all values are ordered, is also very close to the mean for all variables—31 for Age, 118 for SystolicBP, 78.1 for DiastolicBP, and 23.9 for BMI.

This closeness between the mean and median suggests that the data is fairly symmetrically distributed without significant outliers. The standard deviation (std) measures how spread out the data is around the mean. For example, Age has a standard deviation of 9.24, indicating that most individuals' ages fall within about ±9 years of the mean. Similarly, SystolicBP and DiastolicBP have standard deviations of 12.87 and 12.30 respectively, showing moderate variability in blood pressure readings, while BMI has a standard deviation of 4.55.

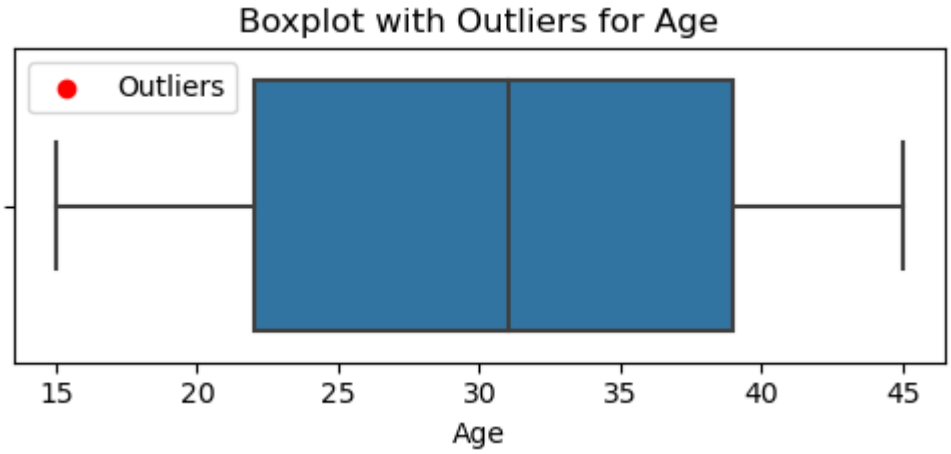
Overall, the dataset shows consistent values with a balanced distribution across all variables.

CLEANING DATASET BY DETECT OUTLIER AND REMOVE THE OUTLIER

In [65]:
```
# Define the function to detect outliers using IQR
def find_outliers(df, column):
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
```

```
    return df[(df[column] < lower_bound) | (df[column] > upper_bound)]

# List of columns to check
columns = ['Age', 'SystolicBP', 'DiastolicBP', 'BMI']

# Loop through each column
for c in columns:
    outliers = find_outliers(df, c)
    print(f"Outliers in {c}:")
    print(outliers)
    print(f"Count: {len(outliers)}\n")

    # Plot boxplot
    plt.figure(figsize=(5, 2.5))
    sns.boxplot(x=df[c])

    # Overlay outliers as red dots
    y_vals = [0] * len(outliers)
    plt.scatter(outliers[c], y_vals, color='red', zorder=10, label='Outliers')

    plt.title(f"Boxplot with Outliers for {c}")
    plt.xlabel(c)
    plt.legend()
    plt.tight_layout()
    plt.show()
```

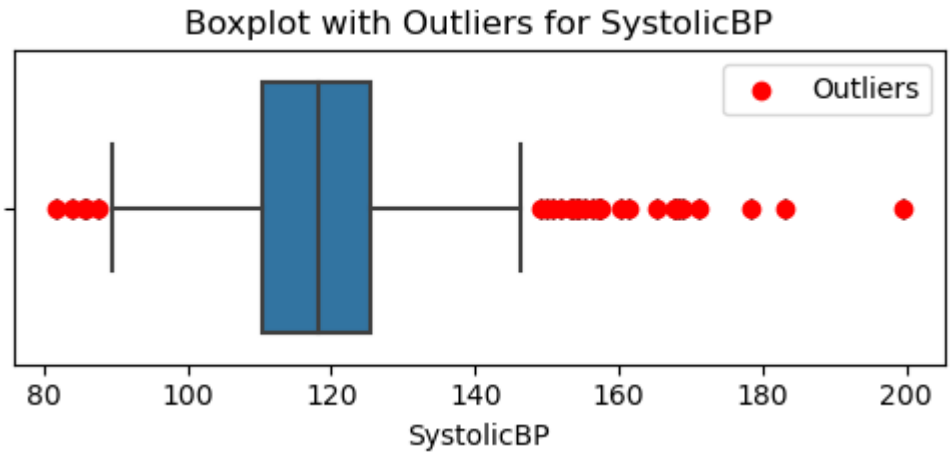
```
Outliers in Age:
Empty DataFrame
Columns: [Age, SystolicBP, DiastolicBP, BMI, Outcome_Hypertension]
Index: []
Count: 0
```



Outliers in SystolicBP:

	Age	SystolicBP	DiastolicBP	BMI	Outcome_Hypertension
21	35	199.5	115.5	26.2	hypertension
89	16	85.9	56.4	24.0	no hypertension
145	31	183.0	110.0	23.0	hypertension
157	21	161.3	121.8	25.4	hypertension
171	22	167.8	93.9	30.4	hypertension
179	28	153.2	67.7	24.1	hypertension
232	25	157.6	90.5	27.5	hypertension
238	44	149.9	119.0	27.4	hypertension
275	16	83.9	42.8	20.9	no hypertension
293	44	171.2	90.3	19.8	hypertension
331	40	169.0	117.3	23.9	hypertension
380	34	157.1	105.2	26.1	hypertension
430	37	157.3	95.8	26.4	hypertension
467	23	153.5	88.0	25.7	hypertension
472	28	81.7	33.8	26.7	no hypertension
530	22	154.5	61.7	23.8	hypertension
590	45	149.0	113.1	19.0	hypertension
626	37	150.7	99.7	21.0	hypertension
654	15	160.2	92.1	22.5	hypertension
751	31	152.0	80.9	12.9	hypertension
767	31	155.3	101.3	31.0	hypertension
787	45	168.4	108.6	14.1	hypertension
813	45	178.4	104.8	21.9	hypertension
909	30	165.2	91.6	21.0	hypertension
916	17	85.7	40.5	25.0	no hypertension
924	28	87.6	51.2	27.3	no hypertension
928	35	156.4	107.5	26.7	hypertension
942	38	154.2	79.3	28.1	hypertension
983	38	167.7	117.0	21.9	hypertension

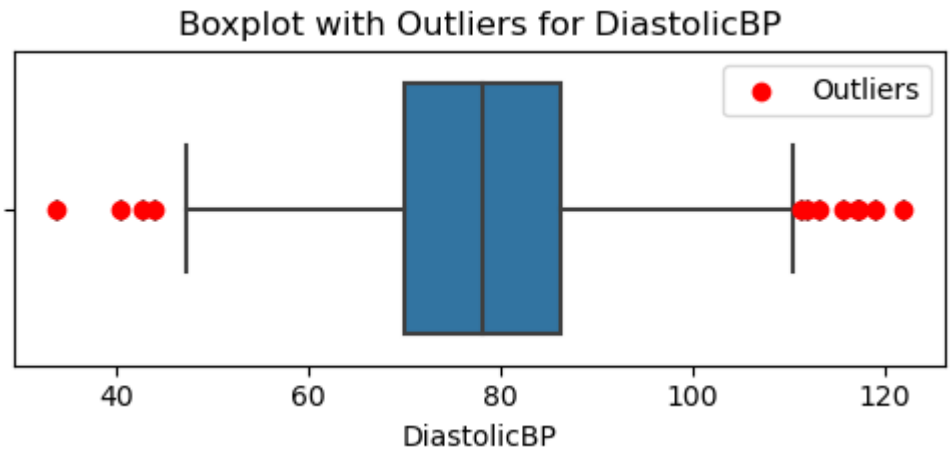
Count: 29



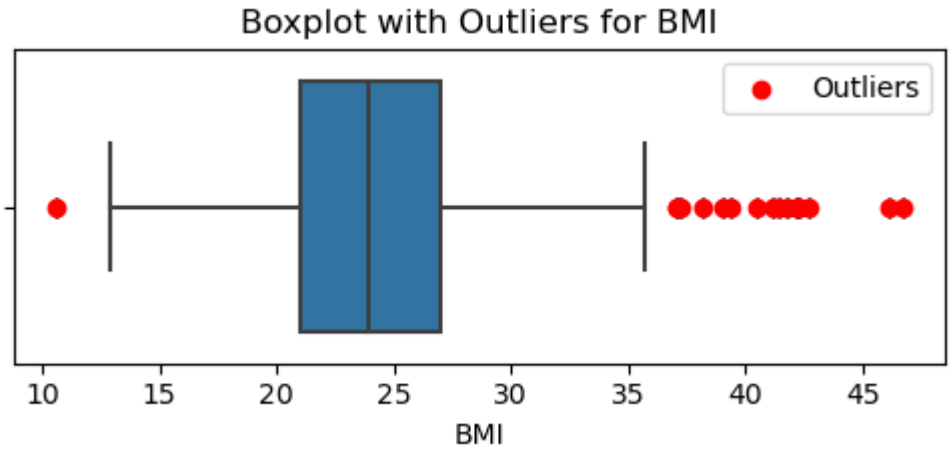
Outliers in DiastolicBP:

	Age	SystolicBP	DiastolicBP	BMI	Outcome_Hypertension
21	35	199.5	115.5	26.2	hypertension
157	21	161.3	121.8	25.4	hypertension
238	44	149.9	119.0	27.4	hypertension
275	16	83.9	42.8	20.9	no hypertension
331	40	169.0	117.3	23.9	hypertension
472	28	81.7	33.8	26.7	no hypertension
527	27	95.6	44.0	22.7	no hypertension
590	45	149.0	113.1	19.0	hypertension
703	26	144.9	111.9	28.0	hypertension
853	45	146.2	111.3	16.0	hypertension
916	17	85.7	40.5	25.0	no hypertension
983	38	167.7	117.0	21.9	hypertension

Count: 12




```
Outliers in BMI:
     Age  SystolicBP  DiastolicBP   BMI Outcome_Hypertension
82    17       119.5         78.1  42.3          hypertension
182   19        96.2         51.8  41.2          hypertension
219   24       122.9         85.4  42.2          hypertension
359   30       102.3         65.6  37.2          hypertension
367   44       121.2         77.9  10.6       no hypertension
440   41       117.8         71.7  46.1          hypertension
455   17       120.4         76.7  42.2          hypertension
490   40       116.0         66.6  39.0          hypertension
494   38       114.6         74.6  42.7          hypertension
669   33       111.9         66.6  39.4          hypertension
699   15       114.5         78.3  41.4          hypertension
725   23       130.3         94.0  46.7          hypertension
798   34       124.2         77.3  37.1          hypertension
815   19       103.7         66.2  37.1          hypertension
868   24       127.6         90.5  41.8          hypertension
870   33       113.7         70.2  38.2          hypertension
998   35       112.0         66.0  40.5          hypertension
Count: 17
```

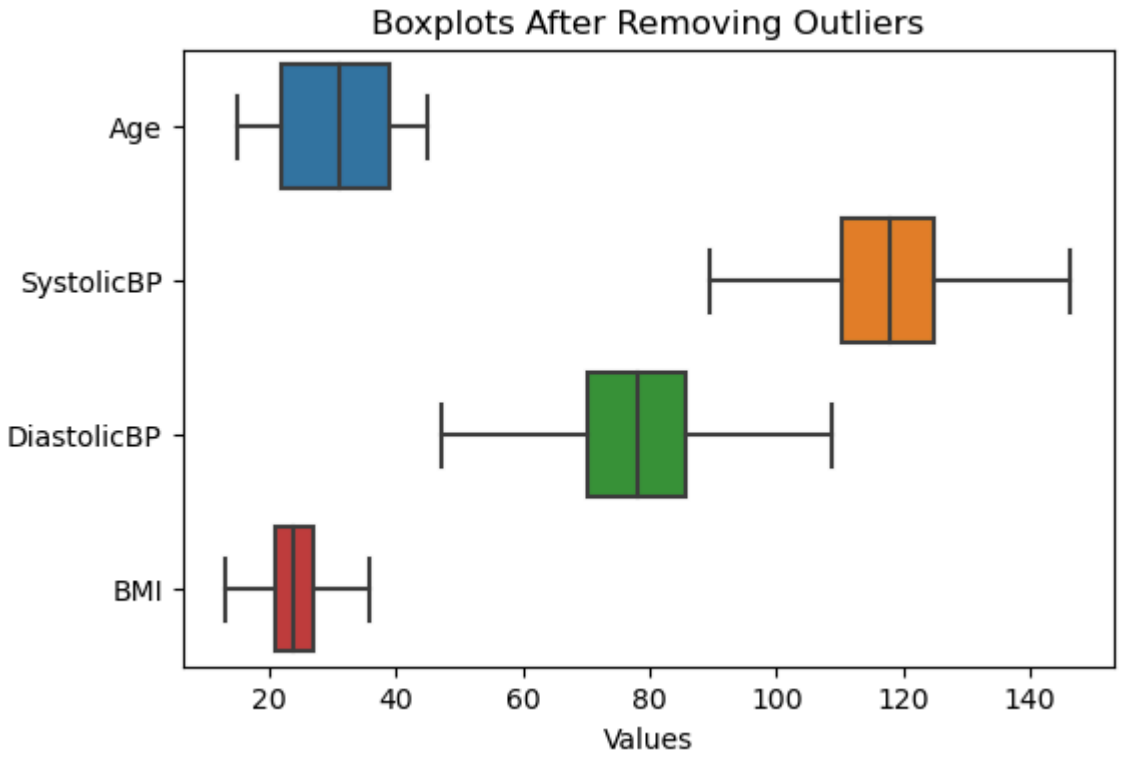


REMOVE THE OUTLIERS AND THE SAVED CLEANED DATASET

In [66]:
```
def remove_outliers(df, col):
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
    lower = Q1 - 1.5 * IQR
    upper = Q3 + 1.5 * IQR
```

```
    return df[(df[col] >= lower) & (df[col] <= upper)]

# Make a copy of the original data
clean_data = df.copy()

# Columns to clean
columns = ['Age', 'SystolicBP', 'DiastolicBP', 'BMI']

# Remove outliers from each column
for c in columns:
    clean_data = remove_outliers(clean_data, c)

# Check new shape after removing outliers
print("Data shape after removing outliers:", clean_data.shape)

# boxplots of cleaned data
plt.figure(figsize=(6, 4))
sns.boxplot(data=clean_data[columns], orient='h')
plt.title("Boxplots After Removing Outliers")
plt.xlabel("Values")
plt.show()
```

```
Data shape after removing outliers: (949, 5)
```



SAVED CLEANED DATASET

In [67]:
```
# Save the cleaned data to a CSV file
clean_data.to_csv('pregnancy_cleaned_data.csv', index=False)

print("Cleaned dataset saved as 'pregnancy_cleaned_data.csv'")
```

```
Cleaned dataset saved as 'pregnancy_cleaned_data.csv'
```

DESCRIPTIVE ANALYSIS CLEANING DATASET

In [68]:
```
clean_data.describe()
```

Out[68]:

	Age	SystolicBP	DiastolicBP	BMI
count	949.000000	949.000000	949.000000	949.000000
mean	30.428872	117.692202	77.751949	23.895153
std	9.231804	10.537381	11.323543	3.997908
min	15.000000	89.500000	47.300000	13.000000
25%	22.000000	110.300000	70.100000	21.000000
50%	31.000000	117.800000	78.000000	23.800000
75%	39.000000	124.800000	85.700000	26.900000
max	45.000000	146.400000	108.700000	35.700000

SUMMARY INTERPRETATION OF DESCRIPTIVE ANALYSIS AFTER CLEANING DATASET

Count: There are 949,000 records for each variable (Age, SystolicBP, DiastolicBP, BMI).

Mean: The average age is 30.43 years, systolic blood pressure is 117.69 mmHg, diastolic blood pressure is 77.75 mmHg, and BMI is 23.90.

Standard Deviation (std): The variability is 9.23 years for age, 10.57 mmHg for systolic blood pressure, 11.32 mmHg for diastolic blood pressure, and 3.98 for BMI.

Minimum (min): The lowest values are 15 years for age, 89.5 mmHg for systolic blood pressure, 47.3 mmHg for diastolic blood pressure, and 13.0 for BMI.

25%(Lower Range): Most young and healthy — age 22 or younger, with normal blood pressure (≤110/70 mmHg) and low BMI (≤21).

50%(Median): Half the people are 31 or younger, with slightly higher but still healthy blood pressure (≤117/78 mmHg) and BMI (≤23).

75% (Upper Range): Most are 39 or younger, with borderline blood pressure (≤124/85 mmHg) and BMI (≤26), which may be approaching the overweight range.

Maximum (max): The highest values are 45 years for age, 146 mmHg for systolic blood pressure, 108.7 mmHg for diastolic blood pressure, and 35.7 for BMI.

NOTE: as age increases, blood pressure and BMI also tend to increase gradually, but most values remain within or near healthy limits.

Based on the descriptive analysis of the cleaned dataset, here are three insights that can help a health organization in decision-making

1.Focus on Hypertension Screening for Middle-Aged pregnancy: The mean age is 30.43 years, with most pregnancy (25% to 75%) falling between 22 and 39 years. With mean SystolicBP at 117.69 mmHg and DiastolicBP at 77.75 mmHg, and outliers reaching 146.4 mmHg and 108.7 mmHg respectively, the organization should prioritize regular BP screenings for this age group to detect and manage hypertension early, especially since high BP values align with hypertension outcomes in the dataset.

2.Target Obesity Prevention Programs: The mean BMI is 23.90 kg/m^2, with most pregnancy (25% to 75%) between 21 and 26 kg/m^2, but outliers reach up to 35.7 kg/m^2. This suggests a small but significant group may be obese, a risk factor for hypertension. The organization could implement targeted weight management and education programs to reduce BMI and lower hypertension risk.

3.Monitor and Support Early pregnancy: The age range starts at 15 years, with a minimum BP of 89.5 mmHg (Systolic) and 47.3 mmHg (Diastolic), indicating some young pregnacy may have low or variable BP. The organization could offer routine health check-ups and support for teenagers and young adults to ensure early detection of BP irregularities and promote healthy lifestyles.

THE DATASET OF CATEGORICAL VALUE AND SUMMARY STATISTIC OF EACH GROUP

In [69]:
```
# Load the dataset
df = pd.read_csv('pregnancy_cleaned_data.csv')

# Group by 'Outcome_Hypertension' and calculate summary statistics
summary = df.groupby('Outcome_Hypertension').agg({
    'Age': ['mean', 'median', 'std'],
    'SystolicBP': ['mean', 'median', 'std'],
    'DiastolicBP': ['mean', 'median', 'std'],
    'BMI': ['mean', 'median', 'std']
})

# Clean column names for better readability
summary.columns = [f'{col[0]}_{col[1]}' for col in summary.columns]
summary = summary.reset_index()

# Display summary statistics as a table
print("Summary Statistics by Outcome_Hypertension:\n")
display(summary)  # Works in Jupyter, Google Colab, or IPython environments

# Display count of each group
group_counts = df['Outcome_Hypertension'].value_counts().reset_index()
group_counts.columns = ['Outcome_Hypertension', 'Count']

print("\nCount of Each Group:\n")
display(group_counts)
```

```
Summary Statistics by Outcome_Hypertension:
```

	Outcome_Hypertension	Age_mean	Age_median	Age_std	SystolicBP_mean	SystolicBP_median
0	hypertension	33.701149	36.0	8.837146	129.058046	130.35
1	no hypertension	29.694194	30.0	9.164201	115.140387	115.90

```
Count of Each Group:
```

	Outcome_Hypertension	Count
0	no hypertension	775
1	hypertension	174

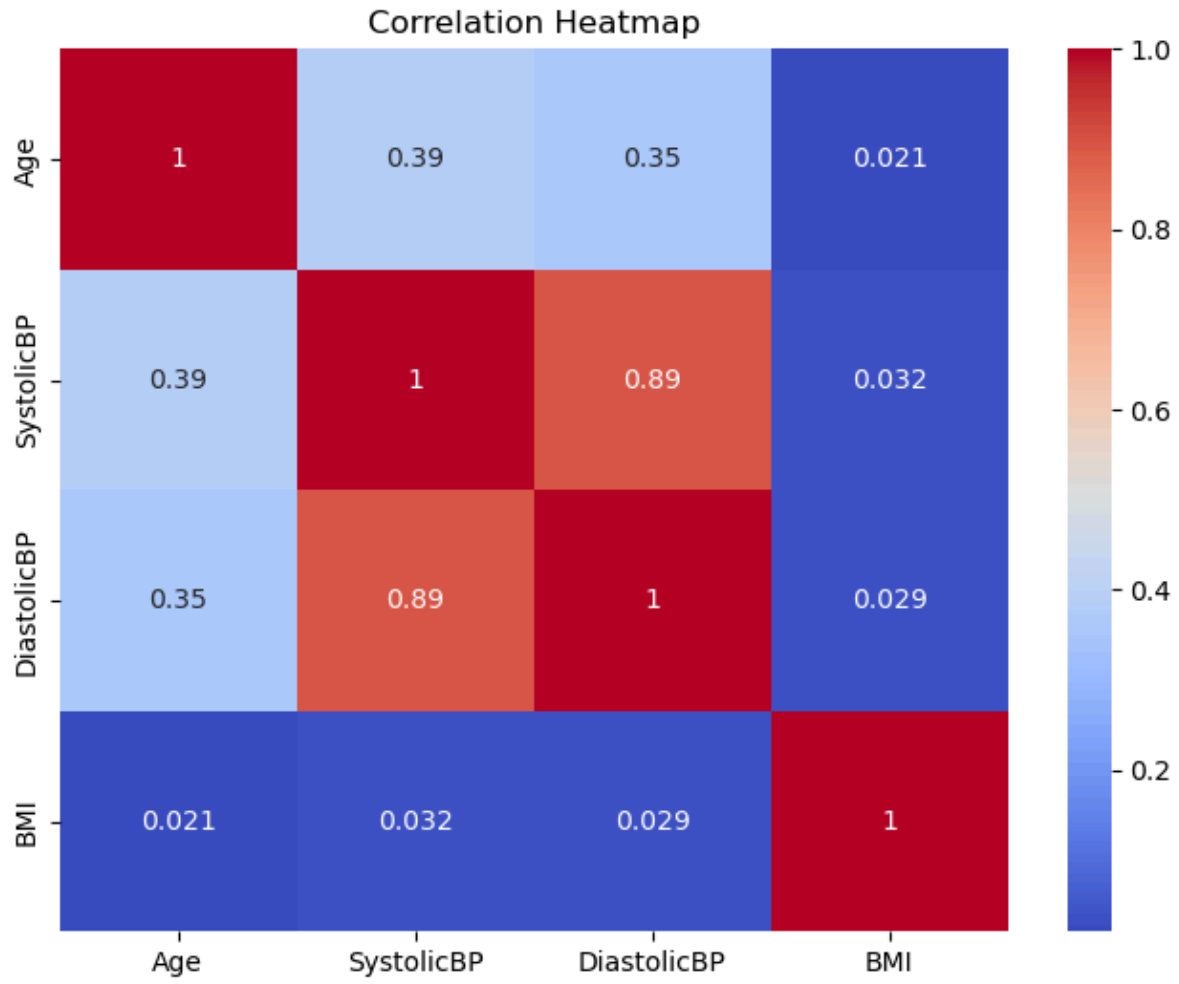
SUMMARY INTERPRETATION INSIGHT FOR CATEGORICAL VALUE

The summary statistics show some clear differences between pregnancy with and without hypertension. Those with hypertension are a bit older, with an average age of 33.7 years compared to 29.7 years for those without, suggesting that age might increase the risk a little. Their blood pressure is much higher, with an average systolic BP of 129 mmHg and diastolic BP of 90.9 mmHg, compared to 115.1 mmHg and 74.8 mmHg for the no-hypertension group, which makes sense since high BP defines hypertension. There are 775 people without hypertension and 174 with it, so most don't have it, but the pregnacy who do show a pattern of higher age and BP. This tells us that focusing on BP checks, especially for pregnacy in their 30s, could help catch and manage hypertension early.

PLOT HEATMAP FOR CORRECTION MATRIX

In [70]:
```
plt.figure(figsize=(8, 6))
sns.heatmap(clean_data.corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```

```
C:\Users\Madina\AppData\Local\Temp\ipykernel_10340\3701086020.py:2: FutureWarning:
The default value of numeric_only in DataFrame.corr is deprecated. In a future ver
sion, it will default to False. Select only valid columns or specify the value of
numeric_only to silence this warning.
  sns.heatmap(clean_data.corr(), annot=True, cmap='coolwarm')
```

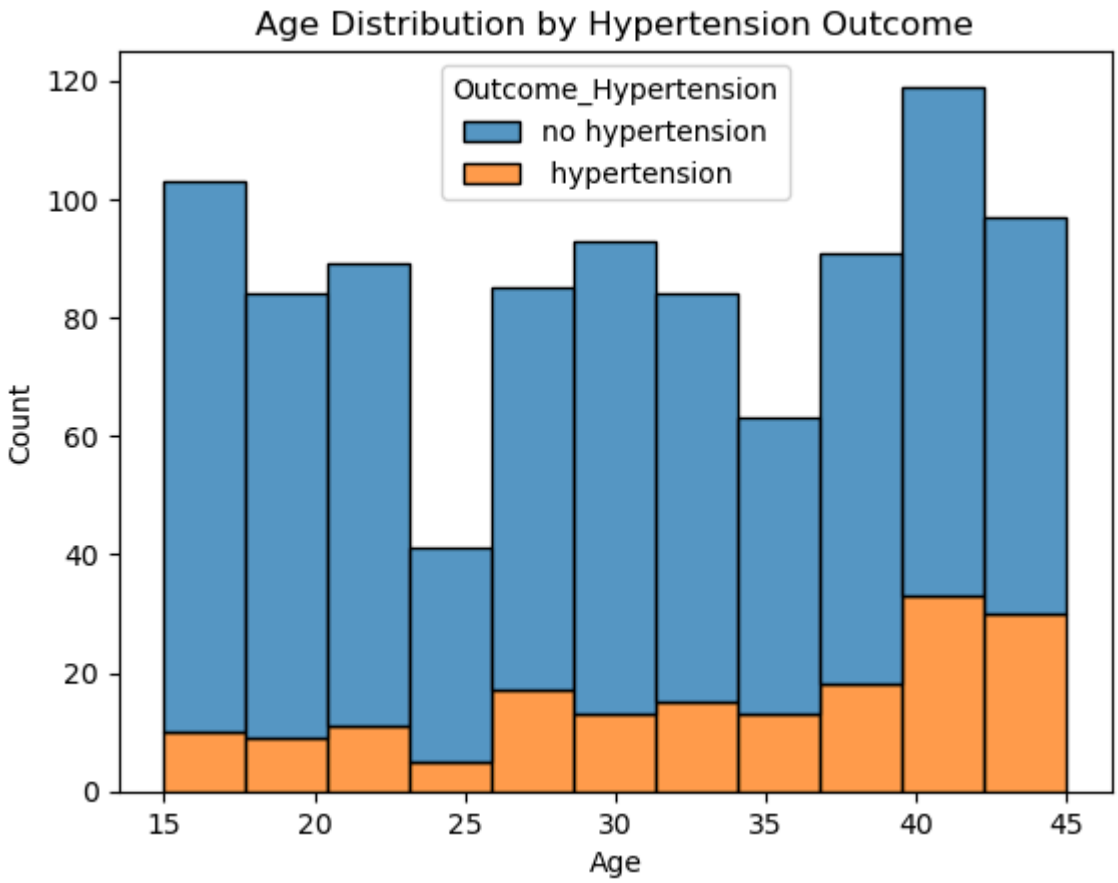


SUMMARY INTERPRETATION FOR HEATMAP

The correlation heatmap analysis examines the relationships between Age, Systolic Blood Pressure (SystolicBP), Diastolic Blood Pressure (DiastolicBP), and Body Mass Index (BMI). The strongest correlations (1.0) are observed within each variable itself, as expected. A significant finding is the strong positive correlation (0.89) between SystolicBP and DiastolicBP, indicating a close physiological link. Age shows moderate positive correlations with SystolicBP (0.39) and DiastolicBP (0.35), suggesting a noticeable influence of aging on blood pressure. In contrast, BMI exhibits very weak correlations with Age (0.021), SystolicBP (0.032), and DiastolicBP (0.029), indicating minimal association with these variables. Overall, the analysis highlights a strong interrelationship between blood pressure measures and a moderate age effect, while BMI appears largely independent.

PLOT HISTOGRAM Age by Outcome_Hypertension

In [71]:
```
# Plot histogram of Age by Outcome_Hypertension
sns.histplot(data=clean_data, x='Age', hue='Outcome_Hypertension', multiple='stack'
plt.title('Age Distribution by Hypertension Outcome')
plt.xlabel('Age')
plt.ylabel('Count')
plt.show()
```

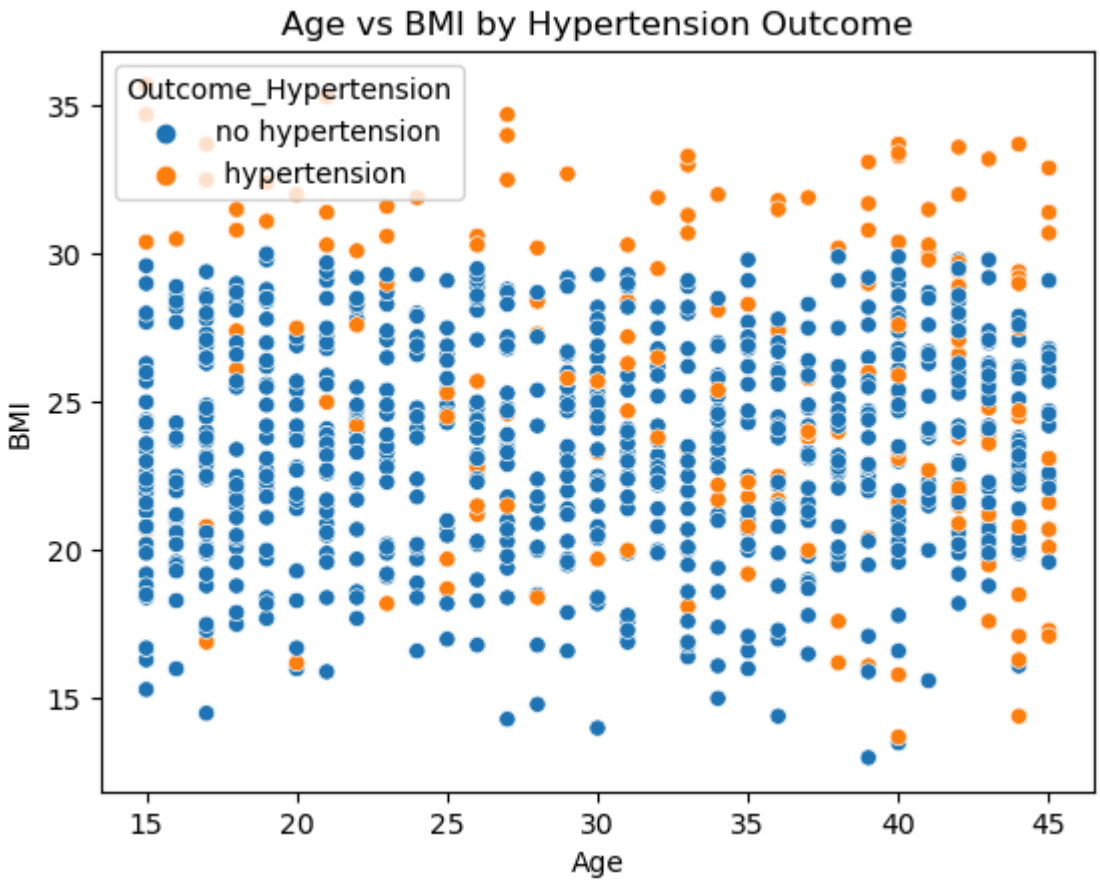


SUMMARY INTERPRETATION FOR HEATMAP Age distribution by Outcome_Hypertension

The histogram illustrates the age distribution of individuals with and without hypertension across ages 15 to 45. The blue bars represent individuals with no hypertension, while the orange bars indicate those with hypertension. The data shows a higher count of individuals without hypertension across all age groups, with peaks around ages 15-20 and 40-45, each nearing 100 counts. Hypertension cases (orange) are consistently fewer, with counts generally below 20 per age group, but they increase slightly in the 35-45 age range. This suggests that hypertension prevalence is lower overall but rises with age, particularly after 35, while most individuals in this dataset remain non-hypertensive across all ages.

SCATTER PLOT OF Age by BMI colored by Outcome_hypertension

In [72]:
```
# Scatter plot of Age vs BMI colored by Outcome
sns.scatterplot(data=clean_data, x='Age', y='BMI', hue='Outcome_Hypertension')
plt.title('Age vs BMI by Hypertension Outcome')
plt.xlabel('Age')
plt.ylabel('BMI')
plt.show()
```



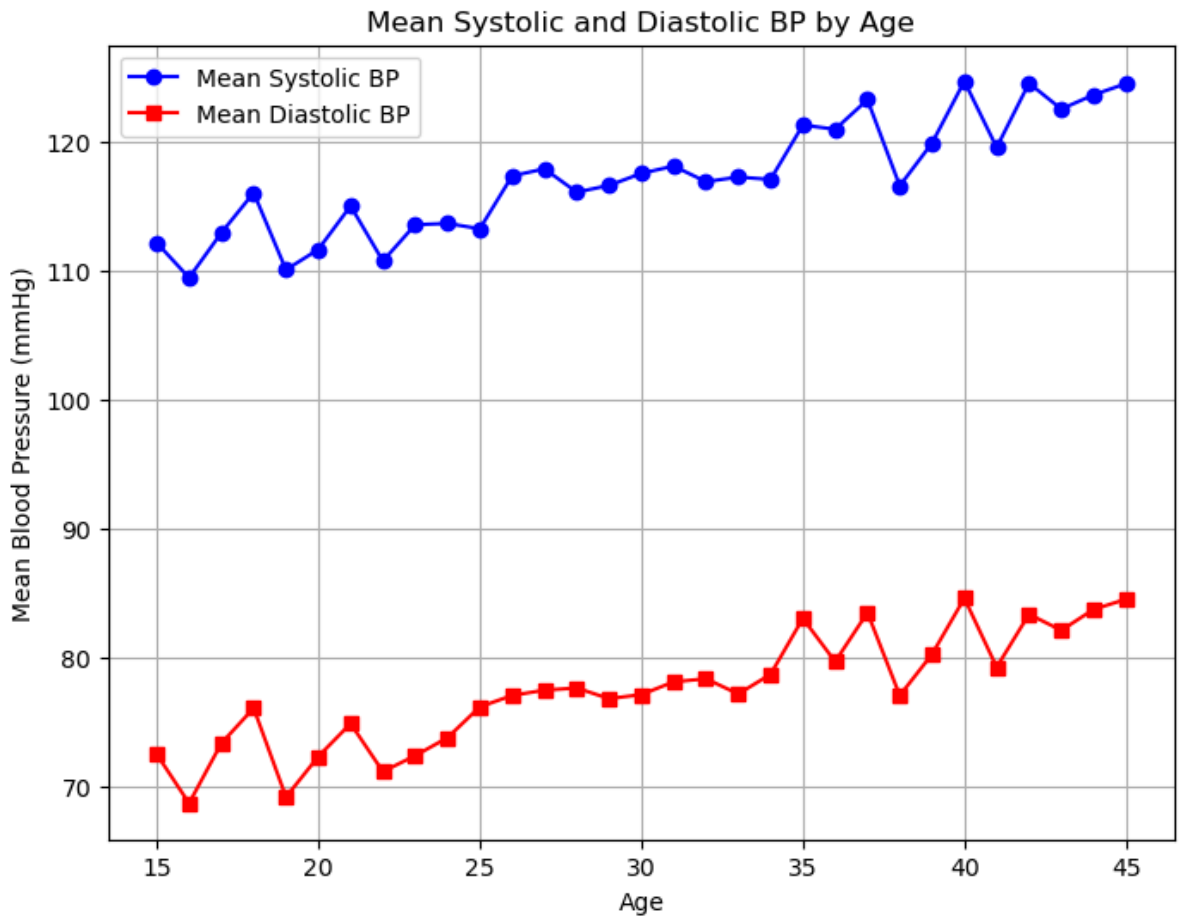
SUMMARY INTERPRETATION Age Vs BMI by Outcome_Hypertension

The scatter plot displays the relationship between Age (15 to 45 years) and BMI (15 to 35) for individuals with and without hypertension. Blue dots represent those with no hypertension, while orange dots indicate those with hypertension. Both groups are distributed across the entire age and BMI range, showing no clear clustering or distinct pattern separating the two outcomes. Individuals with hypertension (orange) are fewer in number compared to those without (blue), consistent across all ages and BMI levels. The plot suggests that neither age nor BMI strongly differentiates hypertension outcomes in this dataset, as both groups overlap significantly without a visible trend or boundary. This aligns with the weak correlations between BMI and other variables observed in the earlier heatmap analysis.

PLOT LINE PLOT Mean Systolic and Diastolic BP by Age

In [73]:
```
# Group by Age and calculate mean BP
mean_bp = df.groupby('Age')[['SystolicBP', 'DiastolicBP']].mean().reset_index()

# Create line plot
plt.figure(figsize=(8, 6))
plt.plot(mean_bp['Age'], mean_bp['SystolicBP'], marker='o', color='blue', label='Me
plt.plot(mean_bp['Age'], mean_bp['DiastolicBP'], marker='s', color='red', label='Me
plt.xlabel('Age')
plt.ylabel('Mean Blood Pressure (mmHg)')
plt.title('Mean Systolic and Diastolic BP by Age')
plt.legend()
```

```
plt.grid(True)
plt.show()
```



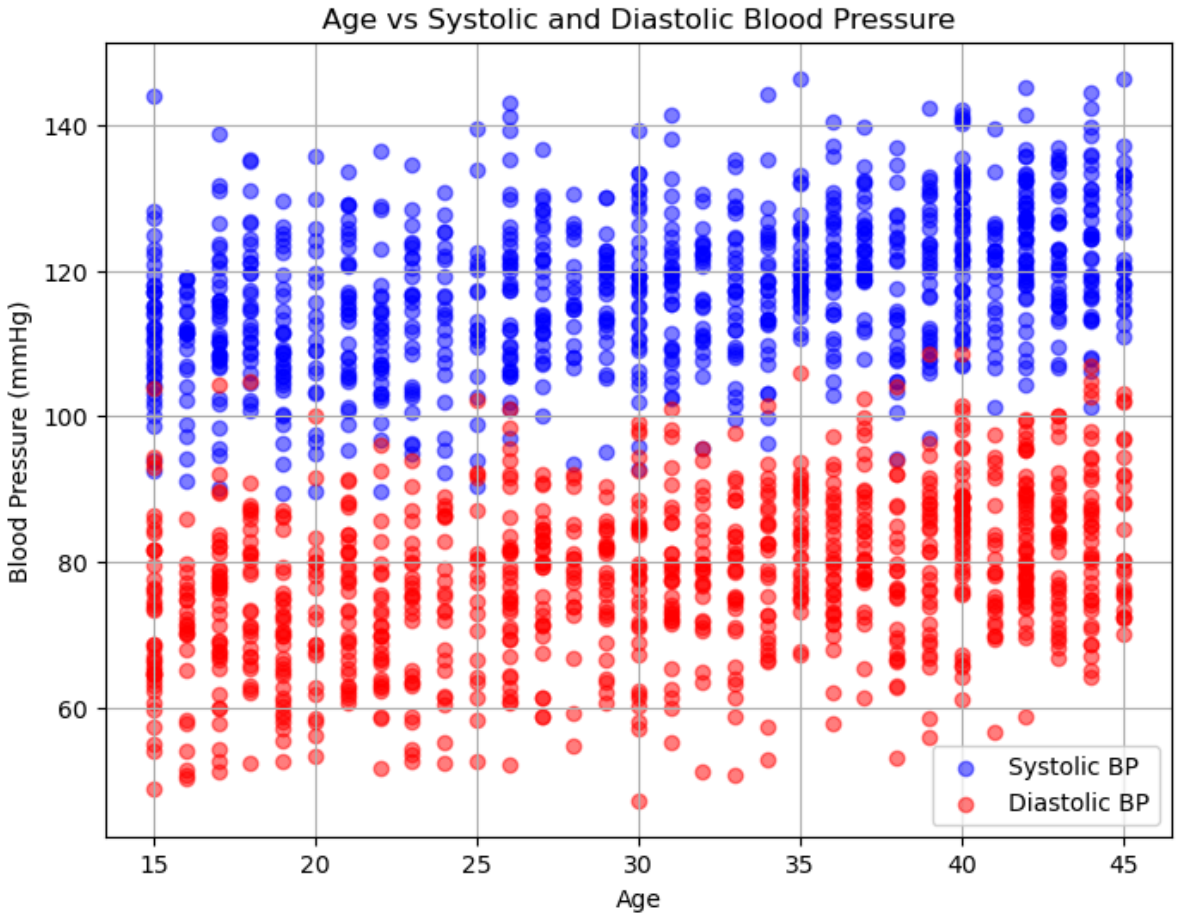
SUMMARY INTERPRETATION Mean SystoliBP and DiastolicBP by Age

The line plot illustrates the average Systolic and Diastolic Blood Pressure (BP) across ages 15 to 45 in the pregnancy hypertension dataset. Systolic BP (blue line) ranges from approximately 100-120 mmHg, while Diastolic BP (red line) ranges from 70-90 mmHg. Both show an upward trend with age, with notable peaks around 35-45 years, suggesting a potential increase in hypertension risk.

The mean BP values smooth out individual variations, highlighting a general rise in BP with age, which may reflect physiological changes or higher hypertension prevalence in older groups.

PLOT SCATTER Age vs SystolicBP and DiastolicBP

In [74]:
```
# Create scatter plot for Age vs SystolicBP and DiastolicBP
plt.figure(figsize=(8, 6))
plt.scatter(df['Age'], df['SystolicBP'], color='blue', label='Systolic BP', alpha=0
plt.scatter(df['Age'], df['DiastolicBP'], color='red', label='Diastolic BP', alpha=
plt.xlabel('Age')
plt.ylabel('Blood Pressure (mmHg)')
plt.title('Age vs Systolic and Diastolic Blood Pressure')
plt.legend()
plt.grid(True)
plt.show()
```

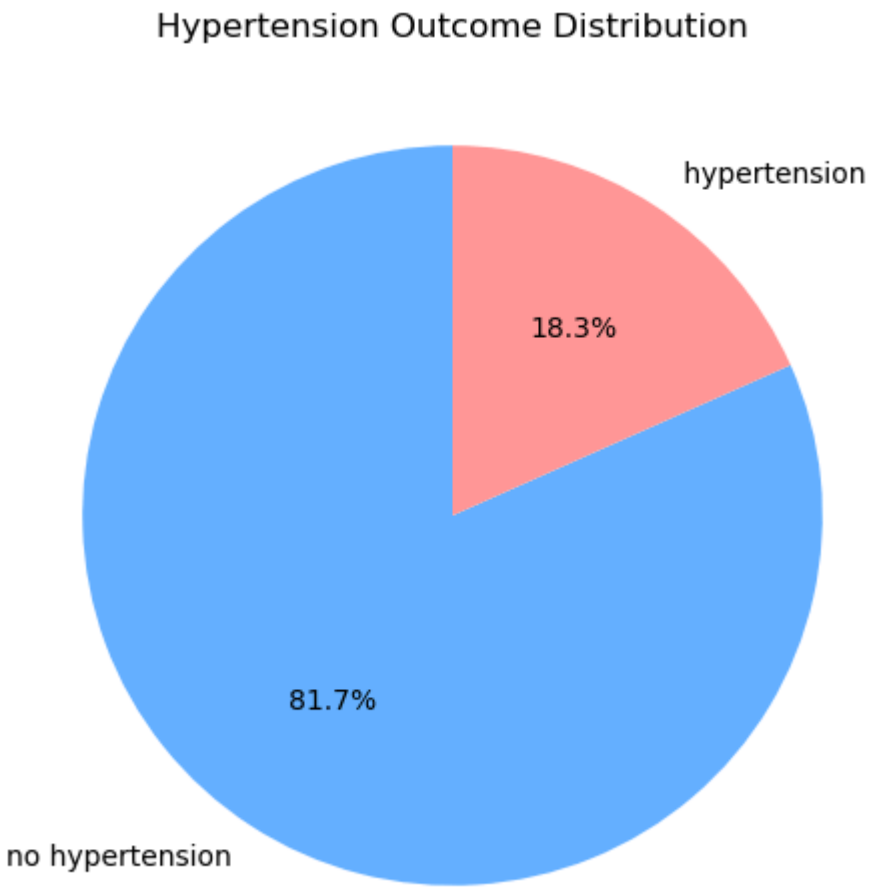


SUMMARY INTERPRETATION Age vs Systolic and Diastolic Blood Pressure

This scatter plot displays individual Systolic Blood Pressure (BP) (blue dots) and Diastolic BP (red dots) across ages 15 to 45 in the pregnancy hypertension dataset. The x-axis represents age, and the y-axis shows BP in mmHg, ranging from 60 to 140. The plot reveals a wide spread of BP values at each age, with Systolic BP generally higher (around 90-140 mmHg) and Diastolic BP lower (around 50-100 mmHg). A slight upward trend with age is visible, suggesting a potential increase in BP, possibly linked to hypertension risk in older groups.

PLOT PIE CHART Outcome_hypertension distribution

In [75]:
```
# Count the occurrences of each outcome
counts = clean_data['Outcome_Hypertension'].value_counts()

# Plot pie chart
plt.figure(figsize=(6,6))
plt.pie(counts, labels=counts.index, autopct='%1.1f%%', startangle=90, colors=['#66
plt.title('Hypertension Outcome Distribution')
plt.show()
```



SUMMARY INTREPRETATION Hypertension Outcome Distribution

The pie chart illustrates the distribution of hypertension outcomes in the dataset. It shows that 81.7% of individuals have no hypertension (represented by the blue section), while 18.3% are diagnosed with hypertension (represented by the pink section). This indicates a significantly higher proportion of individuals without hypertension, suggesting that hypertension is less prevalent in this population. The clear disparity highlights that the majority of the sample does not suffer from hypertension, which aligns with the trends observed in the previous visualizations where non-hypertensive individuals consistently outnumbered those with hypertension across age and BMI distributions.

FIRST WE IMPORT IMPORTANT LIBRARY

In [76]:
```
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
import seaborn as sns
```

LOAD AND READ CLEANED DATASET

In [77]:
```
df= pd.read_csv("pregnancy_cleaned_data.csv")
```

VIEW DATASET

In [78]:
```
df.head(5)
```

Out[78]:

	Age	SystolicBP	DiastolicBP	BMI	Outcome_Hypertension
0	21	116.4	81.3	26.8	no hypertension
1	34	118.6	85.3	23.0	no hypertension
2	43	126.8	83.4	18.8	no hypertension
3	29	119.8	74.0	25.1	no hypertension
4	25	90.5	52.7	24.9	no hypertension

In [79]:
```
df.tail(5)
```

Out[79]:

	Age	SystolicBP	DiastolicBP	BMI	Outcome_Hypertension
944	32	116.2	75.2	22.3	no hypertension
945	29	117.9	82.0	23.0	no hypertension
946	31	120.4	72.9	30.3	hypertension
947	25	110.5	64.2	18.2	no hypertension
948	40	135.5	90.4	27.6	hypertension

NORMALIZATION OF THE FEATURES TO PREVENT BIAS

In [80]:
```
scaler = StandardScaler()
scaled_features = scaler.fit_transform(features)
```

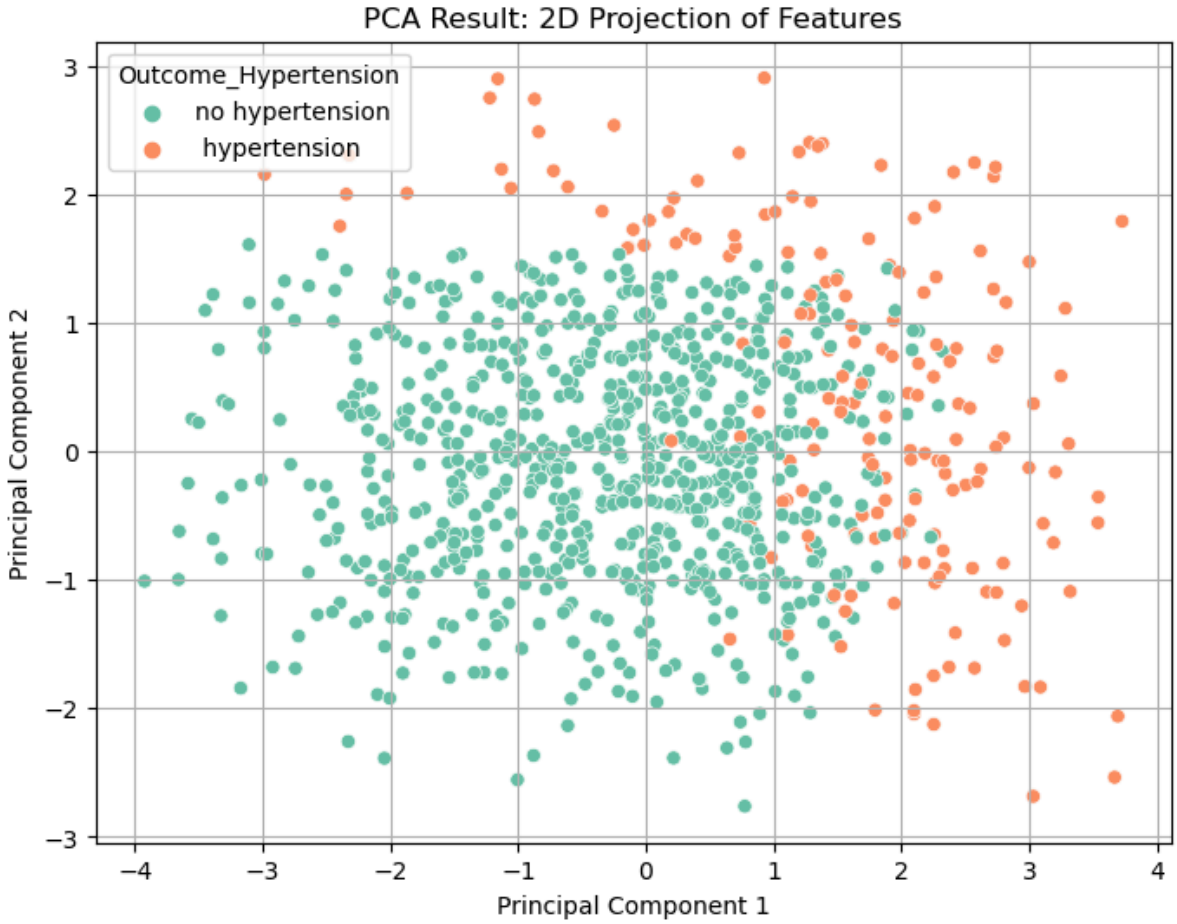
APPLY PCA

In [81]:
```
# Reduce to 2 components for visualization
pca = PCA(n_components=2)
principal_components = pca.fit_transform(scaled_features)

# Create DataFrame with PCA results
pca_df = pd.DataFrame(data=principal_components, columns=['PCA1', 'PCA2'])

# Optional: Add back target for coloring
pca_df['Outcome_Hypertension'] = target
```

TO VISUALIZE THE RESULT

In [82]:
```
plt.figure(figsize=(8, 6))
sns.scatterplot(data=pca_df, x='PCA1', y='PCA2', hue='Outcome_Hypertension', palett
plt.title('PCA Result: 2D Projection of Features')
```

```
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.grid(True)
plt.show()
```



SUMMARY INTERPRETATION OF PCA IN 2DIMENSION FEATURES

The PCA plot shows a 2D projection of the pregnancy dataset, reducing many features into two main components. Each point represents a pregnancy, colored by hypertension outcome—green for no hypertension and red/orange for hypertension. There is a visible pattern where those with hypertension tend to cluster more to the left, while those without are spread more to the center and right. However, there's still overlap between the groups, meaning PCA captures some, but not all, of the variation related to hypertension. This suggests limited separation, useful for visual insight but not sufficient for full classification.

TO SEPARATE THE TARGET BECAUSE USE FOR PREDICTION AS LABEL (Outcome_Hypertension)

In [83]:
```
target = df['Outcome_Hypertension']
features = df.drop(columns=['Outcome_Hypertension'])
```

In [84]:
```
print(df.columns)
```

```
Index(['Age', 'SystolicBP', 'DiastolicBP', 'BMI', 'Outcome_Hypertension'], dtype
='object')
```

ENCODE LABEL TO NUMERICAL

In [85]:
```
#Load the cleaned dataset
df = pd.read_csv("pregnancy_cleaned_data.csv")

#Clean the label column
df['Outcome_Hypertension'] = df['Outcome_Hypertension'].astype(str).str.lower().str

#Map the values manually
df['Outcome_Hypertension'] = df['Outcome_Hypertension'].map({
    'no hypertension': 0,
    'hypertension': 1
})

#Check the result
print("Unique values in Outcome_Hypertension after encoding:")
print(df['Outcome_Hypertension'].unique())

print("\nCheck if there are any NaNs after mapping:")
print(df['Outcome_Hypertension'].isna().sum())
```

```
Unique values in Outcome_Hypertension after encoding:
[0 1]

Check if there are any NaNs after mapping:
0
```

In [86]:
```
#to check first few rows
df.head(5)
```

Out[86]:

	Age	SystolicBP	DiastolicBP	BMI	Outcome_Hypertension
0	21	116.4	81.3	26.8	0
1	34	118.6	85.3	23.0	0
2	43	126.8	83.4	18.8	0
3	29	119.8	74.0	25.1	0
4	25	90.5	52.7	24.9	0

AFTER ENCODE WE NEED TO DO OVERSAMPLING FOR CLASS LABEL HYPERTENSION HAVE FEW ROWS WITHOUT THIS CAN LED CLASS BIAS

In [87]:
```
!pip install -U imbalanced-learn
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: imbalanced-learn in c:\users\madina\appdata\roaming
\python\python311\site-packages (0.13.0)
Requirement already satisfied: numpy<3,>=1.24.3 in c:\programdata\anaconda3\lib\si
te-packages (from imbalanced-learn) (1.24.3)
Requirement already satisfied: scipy<2,>=1.10.1 in c:\programdata\anaconda3\lib\si
te-packages (from imbalanced-learn) (1.10.1)
Requirement already satisfied: scikit-learn<2,>=1.3.2 in c:\users\madina\appdata\r
oaming\python\python311\site-packages (from imbalanced-learn) (1.6.1)
Requirement already satisfied: sklearn-compat<1,>=0.1 in c:\users\madina\appdata\r
oaming\python\python311\site-packages (from imbalanced-learn) (0.1.3)
Requirement already satisfied: joblib<2,>=1.1.1 in c:\programdata\anaconda3\lib\si
te-packages (from imbalanced-learn) (1.2.0)
Requirement already satisfied: threadpoolctl<4,>=2.0.0 in c:\users\madina\appdata
\roaming\python\python311\site-packages (from imbalanced-learn) (3.6.0)
```

In [88]:
```
#import library
from imblearn.over_sampling import SMOTE
```

In [92]:
```
# Import necessary libraries
import pandas as pd
from imblearn.over_sampling import SMOTE

# Load the cleaned dataset
df = pd.read_csv('pregnancy_cleaned_data.csv')

# Clean and encode the Outcome_Hypertension column
df['Outcome_Hypertension'] = df['Outcome_Hypertension'].astype(str).str.lower().str
df['Outcome_Hypertension'] = df['Outcome_Hypertension'].map({
    'no hypertension': 0,
    'hypertension': 1
})

# Check for any NaN values after encoding
print("Unique values in Outcome_Hypertension after encoding:")
print(df['Outcome_Hypertension'].unique())
print("\nCheck if there are any NaNs after mapping:")
print(df['Outcome_Hypertension'].isna().sum())

# Define features (X) and target (y)
X = df.drop('Outcome_Hypertension', axis=1)
y = df['Outcome_Hypertension']

# Apply SMOTE
smote = SMOTE(sampling_strategy={1: 631}, random_state=42)
X_resampled, y_resampled = smote.fit_resample(X, y)

# Create a new balanced DataFrame
balanced_df = pd.DataFrame(X_resampled, columns=X.columns)
balanced_df['Outcome_Hypertension'] = y_resampled

# Check the new class distribution
print("Class distribution after SMOTE:")
print(balanced_df['Outcome_Hypertension'].value_counts())

# Save the balanced dataset
balanced_df.to_csv('balanced_pregnancy_data.csv', index=False)
```

```
Unique values in Outcome_Hypertension after encoding:
[0 1]

Check if there are any NaNs after mapping:
0
Class distribution after SMOTE:
0    775
1    631
Name: Outcome_Hypertension, dtype: int64
```

NOW WE TRAIN THE ML MODEL USE OF LOGISTIC REGERESSION, SVM AND NAIIVE BAYES

In [ ]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn.metrics import classification_report, confusion_matrix

# To load the balanced dataset
df = pd.read_csv('balanced_pregnancy_data.csv')
X = df.drop('Outcome_Hypertension', axis=1)
y = df['Outcome_Hypertension']

# Scale features aim to improve model performance
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Train/test split process
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, ran

# Logistic Regression
log_reg = LogisticRegression(C=1.0, penalty='l2', solver='liblinear', random_state=
log_reg.fit(X_train, y_train)
y_pred_log = log_reg.predict(X_test)
log_acc = accuracy_score(y_test, y_pred_log)
precision_log = precision_score(y_test, y_pred_log)
recall_log = recall_score(y_test, y_pred_log)
f1_log = f1_score(y_test, y_pred_log)

# Naive Bayes
nb = GaussianNB()
nb.fit(X_train, y_train)
y_pred_nb = nb.predict(X_test)
nb_acc = accuracy_score(y_test, y_pred_nb)
precision_nb = precision_score(y_test, y_pred_nb)
recall_nb = recall_score(y_test, y_pred_nb)
f1_nb = f1_score(y_test, y_pred_nb)

# SVM
svc = SVC(kernel='rbf', C=1.0, gamma='scale', random_state=42)
svc.fit(X_train, y_train)
y_pred_svc = svc.predict(X_test)
svc_acc = accuracy_score(y_test, y_pred_svc)
precision_svc = precision_score(y_test, y_pred_svc)
recall_svc = recall_score(y_test, y_pred_svc)
```

```
f1_svc = f1_score(y_test, y_pred_svc)

print("Metrics without PCA:")
print(f"Logistic Regression - Accuracy: {log_acc:.4f}, Precision: {precision_log:.4
print(f"Naive Bayes         - Accuracy: {nb_acc:.4f}, Precision: {precision_nb:.4f}
print(f"SVM                 - Accuracy: {svc_acc:.4f}, Precision: {precision_svc:.4

print("\nClassification report for Logistic Regression:\n")
print(classification_report(y_test, y_pred_log))
```

SUMMARY INTERPRETATION

Model Performance Summary: Logistic Regression: Accuracy ~84%, Precision ~82%, Recall ~84%, F1-score ~83% Balanced performance for both classes.

Naive Bayes: Accuracy ~81%, Precision ~76%, Recall ~85%, F1-score ~80% Good recall but lower precision, meaning more false positives.

SVM: Accuracy ~96%, Precision ~95%, Recall ~98%, F1-score ~96% Best overall performance with high precision and recall.

Recommendation: The SVM model performs the best across all metrics, showing high accuracy and excellent balance between precision and recall. It is the most reliable choice for predicting hypertension outcomes in your dataset.

In [ ]:

```
# Bar chart to show comparison on model performance before PCA using
models = ['Logistic Regression', 'Naive Bayes', 'SVM']
metrics = ['Accuracy', 'Precision', 'Recall', 'F1-score']
data = {
    'Logistic Regression': [0.8440, 0.8244, 0.8372, 0.8308],
    'Naive Bayes': [0.8085, 0.7586, 0.8527, 0.8029],
    'SVM': [0.9645, 0.9474, 0.9767, 0.9618]
}

# Set width of bars
bar_width = 0.2
index = np.arange(len(metrics))

# Create bars
plt.figure(figsize=(8, 6))
for i, model in enumerate(models):
    plt.bar(index + i * bar_width, data[model], bar_width, label=model)

# Customize the plot
plt.xlabel('Metrics')
plt.ylabel('Score')
plt.title('Model Performance Before PCA')
plt.xticks(index + bar_width, metrics)
plt.legend()
plt.ylim(0, 1.0)
plt.grid(True, axis='y', linestyle='--', alpha=0.7)

# Show plot
plt.tight_layout()
plt.show()
```

THIS FOR EVALUTION MATRIX OF LABEL USING HEATMAP PLOT

In [ ]:
```
# Define models
models = {
    "Logistic Regression": LogisticRegression(solver='liblinear', random_state=42),
    "Naive Bayes": GaussianNB(),
    "SVM": SVC(kernel='rbf', C=1.0, gamma='scale', random_state=42)
}

for name, model in models.items():
    print(f"\n{name} Evaluation:\n")

    # Step 5: Train the model and make predictions
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    # Step 6: Classification report
    print(classification_report(y_test, y_pred, target_names=["No Hypertension", "H

    # Step 7: Confusion matrix
    cm = confusion_matrix(y_test, y_pred)

    # Step 8: Plot confusion matrix heatmap
    plt.figure(figsize=(6, 4))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
                xticklabels=["No Hypertension", "Hypertension"],
                yticklabels=["No Hypertension", "Hypertension"])
    plt.title(f"{name} Confusion Matrix Heatmap")
    plt.xlabel("Predicted Label")
    plt.ylabel("True Label")
    plt.tight_layout()
    plt.show()
```

SUMMARY INTERPREATION

Logistic Regression Model Evaluation Report:

The Logistic Regression model achieved an overall accuracy of 84% in classifying hypertension outcomes. The precision and recall for the "No Hypertension" class were both 86% and 85% respectively, indicating the model is good at correctly identifying patients without hypertension. For the "Hypertension" class, precision was 82% and recall was 84%, showing the model also performs well in detecting hypertensive cases with relatively balanced sensitivity and specificity. The confusion matrix heatmap confirms that most predictions fall along the diagonal, demonstrating strong agreement between predicted and true labels.

Interpretation:The darker blue squares (130 and 108) show where the model is most accurate, with a strong ability to correctly identify both No Hypertension and Hypertension cases. The lighter areas (21 and 23) indicate fewer misclassifications, but there's still some confusion, especially with 23 false positives and 21 false negatives.

Naive Bayes Model Evaluation Report:

The Naive Bayes model achieved an overall accuracy of 81% on the test set. It showed a higher recall (85%) for the "Hypertension" class, meaning it successfully identified most hypertensive cases, though the precision for this class was slightly lower at 76%, indicating some false positives. For the "No Hypertension" class, precision was strong at 86%, but recall

was lower at 77%, suggesting some missed cases in this group. Overall, the model is fairly balanced but slightly favors detecting hypertension cases over non-hypertension.

Interpretation:The darker blue squares (118 and 110) indicate strong accuracy in correctly identifying No Hypertension and Hypertension cases. The lighter areas (19 and 35) show more misclassifications, with 35 false positives (over-predicting Hypertension) and 19 false negatives (missing some Hypertension cases).

SVM Model Evaluation Report:

The SVM model delivered the best performance with an impressive accuracy of 96%. It achieved very high precision and recall values for both classes above 95% indicating excellent capability to correctly classify both hypertensive and non-hypertensive patients. The F1-scores of 97% for "No Hypertension" and 96% for "Hypertension" further demonstrate strong, balanced performance. The confusion matrix heatmap for SVM reflects very few misclassifications, making it the most reliable model among the three for this dataset.

Interpretation: The dark blue squares (146 and 126) show excellent accuracy in correctly identifying both No Hypertension and Hypertension cases. The very light areas (3 and 7) indicate minimal misclassifications, with only 7 false positives and 3 false negatives, reflecting the model's high precision and recall.

USING PCA FOR TRAINING ML MODELS

In [ ]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

# To load dataset
df = pd.read_csv('balanced_pregnancy_data.csv')
X = df.drop('Outcome_Hypertension', axis=1)
y = df['Outcome_Hypertension']

# Scale features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Apply PCA (retain 95% variance)
pca = PCA(n_components=0.95, random_state=42)
X_pca = pca.fit_transform(X_scaled)

# Train/test split process
X_train, X_test, y_train, y_test = train_test_split(X_pca, y, test_size=0.2, random

# Logistic Regression
log_reg = LogisticRegression(C=1.0, penalty='l2', solver='liblinear', random_state=
log_reg.fit(X_train, y_train)
y_pred_log = log_reg.predict(X_test)
log_acc = accuracy_score(y_test, y_pred_log)
```

```
precision_log = precision_score(y_test, y_pred_log)
recall_log = recall_score(y_test, y_pred_log)
f1_log = f1_score(y_test, y_pred_log)

# Naive Bayes
nb = GaussianNB()
nb.fit(X_train, y_train)
y_pred_nb = nb.predict(X_test)
nb_acc = accuracy_score(y_test, y_pred_nb)
precision_nb = precision_score(y_test, y_pred_nb)
recall_nb = recall_score(y_test, y_pred_nb)
f1_nb = f1_score(y_test, y_pred_nb)

# SVM
svc = SVC(kernel='rbf', C=1.0, gamma='scale', random_state=42)
svc.fit(X_train, y_train)
y_pred_svc = svc.predict(X_test)
svc_acc = accuracy_score(y_test, y_pred_svc)
precision_svc = precision_score(y_test, y_pred_svc)
recall_svc = recall_score(y_test, y_pred_svc)
f1_svc = f1_score(y_test, y_pred_svc)

print("Metrics WITH PCA:")
print(f"Logistic Regression - Accuracy: {log_acc:.4f}, Precision: {precision_log:.4
print(f"Naive Bayes         - Accuracy: {nb_acc:.4f}, Precision: {precision_nb:.4f}
print(f"SVM                 - Accuracy: {svc_acc:.4f}, Precision: {precision_svc:.4

print("\nClassification report for Logistic Regression with PCA:\n")
print(classification_report(y_test, y_pred_log))
```

In [ ]:

```
# model comparison perfomance after PCA apply
models = ['Logistic Regression', 'Naive Bayes', 'SVM']
metrics = ['Accuracy', 'Precision', 'Recall', 'F1-score']
data = {
    'Logistic Regression': [0.8191, 0.8095, 0.7907, 0.8000],
    'Naive Bayes': [0.8688, 0.8239, 0.9070, 0.8635],
    'SVM': [0.9362, 0.9111, 0.9535, 0.9318]
}

# Set width of bars
bar_width = 0.2
index = np.arange(len(metrics))

# Create bars
plt.figure(figsize=(8, 6))
for i, model in enumerate(models):
    plt.bar(index + i * bar_width, data[model], bar_width, label=model)

# Customize the plot
plt.xlabel('Metrics')
plt.ylabel('Score')
plt.title('Model Performance After PCA')
plt.xticks(index + bar_width, metrics)
plt.legend()
plt.ylim(0, 1.0)
plt.grid(True, axis='y', linestyle='--', alpha=0.7)

# Show plot
plt.tight_layout()
plt.show()
```

SUMMARY INTERPREATION

Model Performance Summary with PCA: Logistic Regression Accuracy: 81.9% Precision: 80.9%, Recall: 79.1%, F1-score: 80.0% → Performs consistently, but slightly lower compared to without PCA.

Naive Bayes Accuracy: 86.9% Precision: 82.4%, Recall: 90.7%, F1-score: 86.4% → High recall makes it good for detecting hypertension cases.

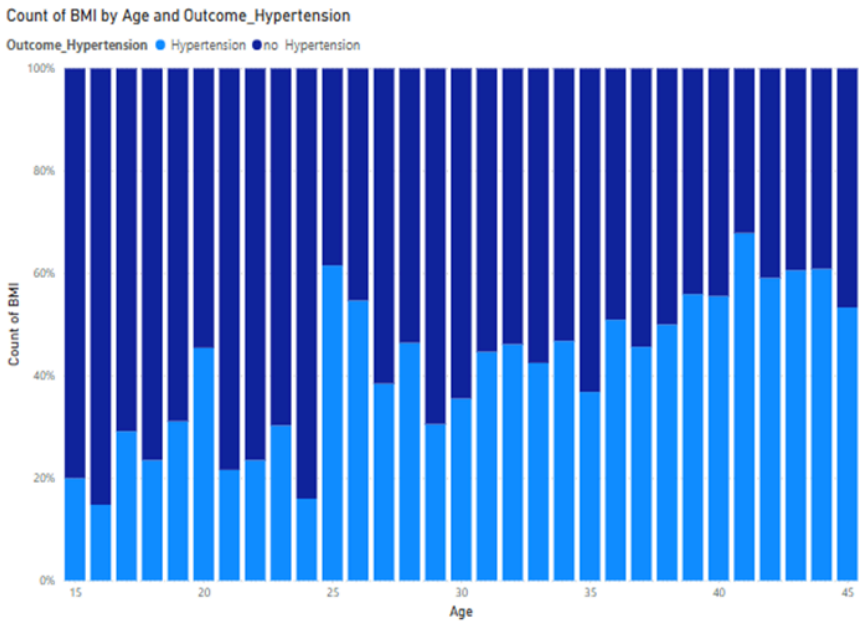
SVM Accuracy: 93.6% Precision: 91.1%, Recall: 95.4%, F1-score: 93.2% → Best performer overall with strong precision and recall.

Recommendation: The SVM model with PCA gives the best overall performance and remains the top choice. However, Naive Bayes with PCA also shows significant improvement and is a good lightweight alternative, especially if computation speed is important.

OVERALL SUMMARY INTERPRETATION COMPARE BEFORE PCA AND AFTER

In this analysis, three models Logistic Regression, Naive Bayes, and Support Vector Machine (SVM) were evaluated before and after applying Principal Component Analysis (PCA). Before PCA, SVM achieved the highest accuracy (96.5%) with strong precision and recall, followed by Logistic Regression and Naive Bayes. After applying PCA to reduce dimensionality and noise, SVM remained the top performer (93.6%), while Naive Bayes showed improvement in recall and overall F1-score. Logistic Regression performance slightly dropped after PCA. Overall, SVM consistently outperformed the other models in both scenarios, indicating its robustness with and without dimensionality reduction.

VISUALIZATION BY POWER BI

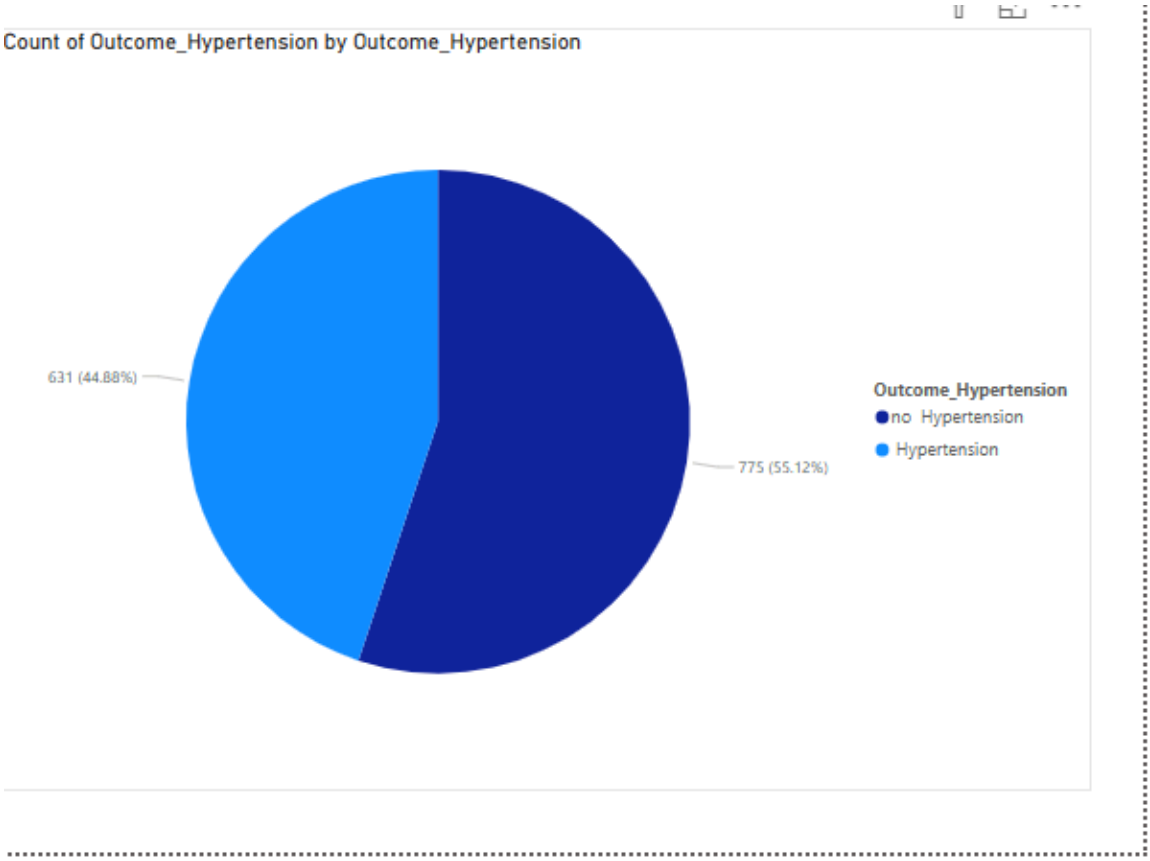


Count of BMI by Age and Outcome_Hypertension interpretation

This 100% stacked column chart shows the distribution of hypertension outcomes across different ages based on BMI counts. Each bar represents an age group, with colours indicating the proportion of individuals with or without hypertension. Light Blue: Individuals diagnosed with hypertension Dark Blue: Individuals without hypertension

In younger age groups (15–25), most individuals do not have hypertension, as shown by the dominance of dark blue. From age 30 onward, the proportion of hypertension cases gradually increases, indicating a rising risk with age. By age 40+, hypertension becomes more prevalent, with some age groups showing almost equal or higher hypertensive cases.

This chart effectively communicates how age may be associated with increased hypertension risk, and how BMI counts vary by outcome. It's useful for health risk profiling and age-specific interventions.



Interpretation of pie chart

This pie chart illustrates the distribution of hypertension outcomes within the dataset. It reveals that the majority of individuals, specifically 55.12% (or 775 individuals), are categorized as having 'Hypertension'. Conversely, 44.88% (or 631 individuals) are reported as having 'no Hypertension'. This clearly shows that hypertension is slightly more prevalent than its absence in the represented population.

Based on the model here are insights that can help a health organization in decision-making

Insights for Health Organization Decision-Making Based on SVM Model

Prioritize Hypertension Screening for women Aged 30 and Above

- The SVM model highlights that women aged 30 and older have a higher prevalence of hypertension, with mean systolic and diastolic blood pressures of 129 mmHg and 90.9 mmHg, respectively, in the hypertensive group. Health organizations should implement targeted blood pressure screening programs for pregnant women in this age group to enable early detection and management of hypertension, reducing risks to maternal and fetal health.

Implement Obesity Prevention Programs

- The model indicates a link between higher BMI and hypertension, with some hypertensive cases having BMIs up to 35.7 kg/m^2. Health organizations should develop nutrition and weight management programs for pregnant women to maintain healthy BMI levels, thereby reducing the risk of hypertension and associated complications during pregnancy.

Allocate Resources for Hypertension Management

- With 55.12% of the dataset's population exhibiting hypertension, the SVM model underscores the need for adequate resource allocation. Health organizations should ensure the availability of hypertension medications and support services, particularly for high-risk groups, to enhance maternal health outcomes and streamline care