Load and Inspect Data

In [2]:
```
import pandas as pd

# Load the dataset
file_path = 'diabetes_prediction_dataset.csv'
data = pd.read_csv(file_path)

# Display the first few rows to understand the structure and content
data.info()
data.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 9 columns):
 #   Column               Non-Null Count   Dtype  
---  ------               --------------   -----  
 0   gender               100000 non-null  object 
 1   age                  100000 non-null  float64
 2   hypertension         100000 non-null  int64  
 3   heart_disease        100000 non-null  int64  
 4   smoking_history      100000 non-null  object 
 5   bmi                  100000 non-null  float64
 6   HbA1c_level          100000 non-null  float64
 7   blood_glucose_level  100000 non-null  int64  
 8   diabetes             100000 non-null  int64  
dtypes: float64(3), int64(4), object(2)
memory usage: 6.9+ MB
```

Out[2]:

	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_glucose_level	diabetes
0	Female	80.0	0	1	never	25.19	6.6	140	0
1	Female	54.0	0	0	No Info	27.32	6.6	80	0
2	Male	28.0	0	0	never	27.32	5.7	158	0
3	Female	36.0	0	0	current	23.45	5.0	155	0
4	Male	76.0	1	1	current	20.14	4.8	155	0

Remove Duplicates

In [4]:
```
# Check for duplicates and remove them
initial_row_count = data.shape[0]
data.drop_duplicates(inplace=True)
duplicates_removed = initial_row_count - data.shape[0]
print(f'Duplicates Removed: {duplicates_removed}')
```

```
Duplicates Removed: 3854
```

Handle 'No Info' in Smoking History

In [6]:
```
# Replace "No Info" in smoking_history with NaN
data['smoking_history'] = data['smoking_history'].replace("No Info", pd.NA)
```

Missing or incomplete values in the `smoking_history` column are replaced with `NaN` for more accurate analysis of the data.

Rename Columns for Consistency

In [9]:
```
# Rename columns to have consistent names
data.columns = data.columns.str.lower().str.replace(' ', '_')
```

Convert Categorical Columns to Appropriate Types

In [11]:
```
# Manually encode 'gender' column (Female -> 0, Male -> 1)
data['gender_encoded'] = data['gender'].map({'Female': 0, 'Male': 1})

# Manually encode 'smoking_history' column with custom mappings
smoking_mapping = {
    'never': 0,
    'former': 1,
    'current': 2,
    'NaN': -1,
    'No Info': 3  # Optionally handle 'No Info' if present
}

data['smoking_history_encoded'] = data['smoking_history'].map(smoking_mapping)
```

```
# Check the result
print(data[['gender_encoded', 'smoking_history_encoded']].head())
```

```
   gender_encoded  smoking_history_encoded
0             0.0                     0.0
1             0.0                     NaN
2             1.0                     0.0
3             0.0                     2.0
4             1.0                     2.0
```

In [12]:
```
data.head()
```

Out[12]:

	gender	age	hypertension	heart_disease	smoking_history	bmi	hba1c_level	blood_glucose_level	diabetes	gender_encoded	smoking_
0	Female	80.0	0	1	never	25.19	6.6	140	0	0.0	
1	Female	54.0	0	0	<NA>	27.32	6.6	80	0	0.0	
2	Male	28.0	0	0	never	27.32	5.7	158	0	1.0	
3	Female	36.0	0	0	current	23.45	5.0	155	0	0.0	
4	Male	76.0	1	1	current	20.14	4.8	155	0	1.0	

Identify and Handle Outliers in BMI

In [14]:
```
# Handling outliers using IQR method for BMI
Q1 = data['bmi'].quantile(0.25)
Q3 = data['bmi'].quantile(0.75)
IQR = Q3 - Q1
# Remove outliers outside 1.5 * IQR range
data = data[(data['bmi'] >= (Q1 - 1.5 * IQR)) & (data['bmi'] <= (Q3 + 1.5 * IQR))]
```

The IQR (Interquartile Range) method is used to identify and remove outliers in the `bmi` column by excluding values that fall outside the 1.5 times IQR range. This helps reduce the impact of extreme values.

Normalize the Age Column

In [17]:
```
# Normalize age column
data['age_normalized'] = (data['age'] - data['age'].min()) / (data['age'].max() - data['age'].min())
```

This normalizes the `age` column to values between 0 and 1, which helps in standardizing the scale of the data.

Bin Age Values into Groups

In [20]:
```
# Bin age into categories
data['age_group'] = pd.cut(data['age'], bins=[0, 30, 50, 70, 100], labels=['0-30', '30-50', '50-70', '70+'])
```

This code categorizes the `age` column into groups (e.g., `0-30`, `30-50`, etc.), making it easier to analyze age distribution and relationships between age and diabetes.

Standardize BMI and HbA1c Levels

In [23]:
```
from sklearn.preprocessing import StandardScaler

# Standardize BMI and HbA1c level columns
scaler = StandardScaler()
data[['bmi_standardized', 'hba1c_level_standardized']] = scaler.fit_transform(data[['bmi', 'hba1c_level']])
```

`StandardScaler` is used to standardize the `bmi` and `hba1c_level` columns, ensuring these values have a mean of 0 and a standard deviation of 1. Standardizing can improve the performance of many machine learning algorithms.

Convert Binary Variables to Yes/No

In [26]:
```
# Convert hypertension and heart disease to Yes/No format
data['hypertension'] = data['hypertension'].replace({0: 'No', 1: 'Yes'}).astype('category')
data['heart_disease'] = data['heart_disease'].replace({0: 'No', 1: 'Yes'}).astype('category')
```

The categorical columns `gender` and `smoking_history` are encoded into numerical values. This transformation is essential for machine learning models that require numerical input.

Create a New Feature 'At Risk'

In [29]:
```
# Create a new feature indicating if a person is "at-risk" based on certain conditions
```

```
data['at_risk'] = ((data['hypertension'] == 'Yes') | (data['heart_disease'] == 'Yes') | (data['bmi'] > 30)).ast
```

Exploratory Data Analysis (EDA)

Univariate Analysis of Numerical Features

In [32]:
```
# Summary statistics of numerical features
numerical_summary = data.describe()
print(numerical_summary)
```

```
                age           bmi   hba1c_level  blood_glucose_level  \
count  90792.000000  90792.000000  90792.000000         90792.000000
mean      41.663144     26.410518      5.520132           137.697385
std       22.667178      5.233696      1.065622            40.342693
min        0.080000     13.710000      3.500000            80.000000
25%       23.000000     23.170000      4.800000           100.000000
50%       43.000000     27.320000      5.800000           140.000000
75%       60.000000     28.840000      6.200000           159.000000
max       80.000000     39.550000      9.000000           300.000000

           diabetes  gender_encoded  smoking_history_encoded  age_normalized  \
count  90792.000000    90774.000000              49490.000000    90792.000000
mean       0.079467        0.421894                  0.523843        0.520310
std        0.270468        0.493864                  0.774427        0.283623
min        0.000000        0.000000                  0.000000        0.000000
25%        0.000000        0.000000                  0.000000        0.286787
50%        0.000000        0.000000                  0.000000        0.537037
75%        0.000000        1.000000                  1.000000        0.749750
max        1.000000        1.000000                  2.000000        1.000000

       bmi_standardized  hba1c_level_standardized       at_risk
count      9.079200e+04              9.079200e+04  90792.000000
mean       2.441728e-16              4.331719e-16      0.273009
std        1.000006e+00              1.000006e+00      0.445508
min       -2.426696e+00             -1.895740e+00      0.000000
25%       -6.191677e-01             -6.757889e-01      0.000000
50%        1.737753e-01              2.626351e-01      0.000000
75%        4.642026e-01              6.380047e-01      1.000000
max        2.510569e+00              3.265592e+00      1.000000
```

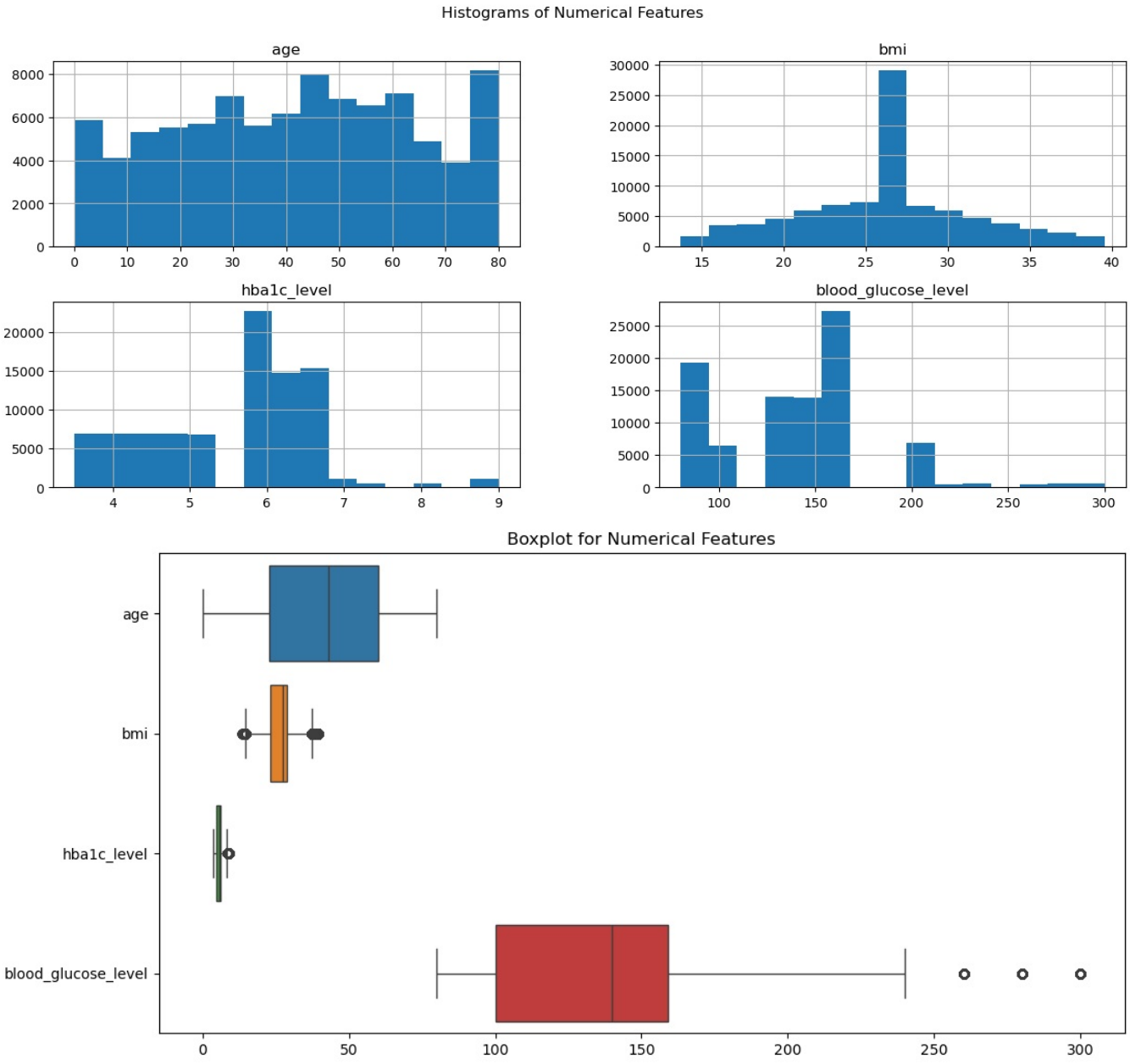
Distribution of Key Numerical Features (Histogram/Boxplot)

Done By Dasarla Akshay Kumar UBIT Name: adasarla UBIT number: 50592353

In [34]:
```
import matplotlib.pyplot as plt
import seaborn as sns

# Plot histograms for numerical features
numerical_cols = ['age', 'bmi', 'hba1c_level', 'blood_glucose_level']
data[numerical_cols].hist(bins=15, figsize=(15, 6))
plt.suptitle('Histograms of Numerical Features')
plt.show()

# Boxplots for detecting outliers
plt.figure(figsize=(12, 6))
sns.boxplot(data=data[numerical_cols], orient='h')
plt.title('Boxplot for Numerical Features')
plt.show()
```



Explanation:

1. **Histogram Plot:**
   - The first part of the code uses `data[numerical_cols].hist()` to generate histograms for the key numerical features: `age`, `bmi`, `hba1c_level`, and `blood_glucose_level`
   - Histograms visualize the distribution of each feature helping to identify the shape (e.g., normal distribution, skewness) and any potential anomalies in the data
   - The `bins=15` argument controls the number of bins used in the histograms, and `figsize=(15, 6)` adjusts the overall size of the plot
2. **Boxplot:**
   - The second part uses `sns.boxplot()` to generate horizontal boxplots for the same numerical columns, Boxplots are useful for detecting outliers and visualizing the spread and quartiles (Q1, Q2, Q3) of the data
   - Boxplots are oriented horizontally (using `orient='h'`), and the figure size is set to `figsize=(12, 6)` to ensure a proper display of the plot

---

Graph Interpretation:

1. **Histograms:**
   - **Age**: The distribution of age appears roughly uniform, with more representation in the older age groups, especially around 80 years
   - **BMI**: The BMI distribution is centered around 25, suggesting most people in the dataset have a BMI close to the normal range
   - **HbA1c Level**: The histogram shows that most values cluster around 6 indicating that a large portion of the population falls into a specific HbA1c range

- **Blood Glucose Level**: The distribution is skewed to the right, with a peak around 150, indicating that most individuals have glucose levels in this range but there are a few with significantly higher glucose levels

2. **Boxplots:**

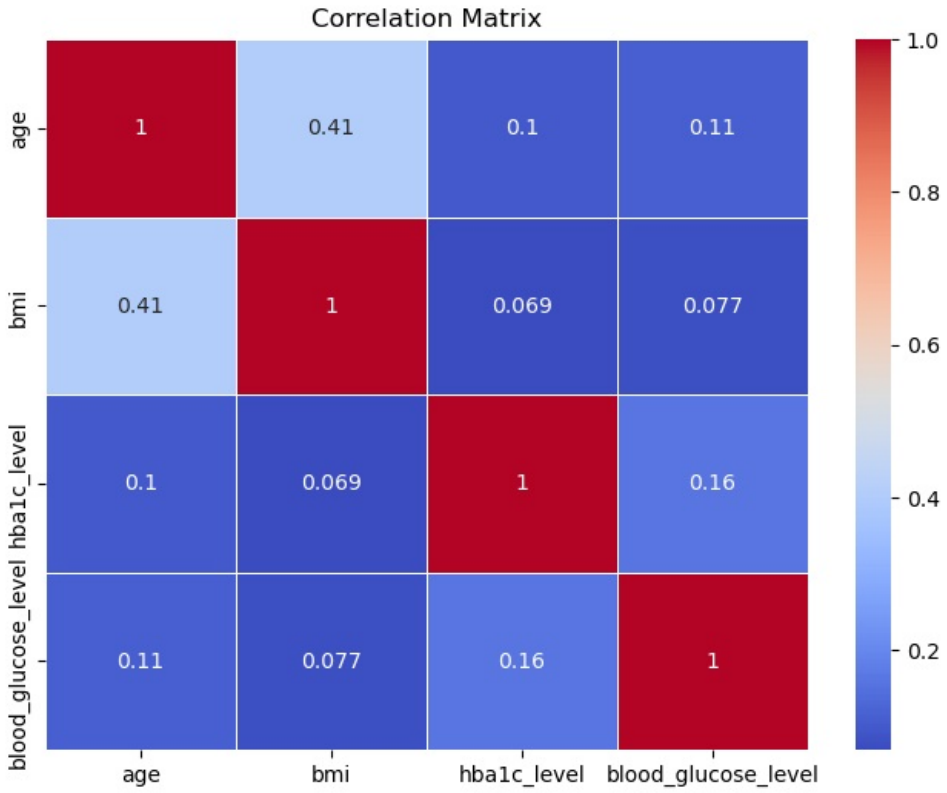
- **Age**: The boxplot shows a broad range of ages with a relatively uniform distribution. There are no clear outliers in age
- **BMI**: There are a few mild outliers, but most data points are concentrated between 20 and 35, suggesting a normal distribution with a few extreme values
- **HbA1c Level**: This boxplot reveals the presence of some outliers, as expected from the histogram
- **Blood Glucose Level**: There are several outliers on the higher end, indicating that some individuals have exceptionally high glucose levels compared to the rest of the population

Correlation Matrix

Done By Dasarla Akshay Kumar UBIT Name: adasarla UBIT number: 50592353

In [37]:
```
# Correlation matrix between numerical features
correlation_matrix = data[['age', 'bmi', 'hba1c_level', 'blood_glucose_level']].corr()

# Plot heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('Correlation Matrix')
plt.show()
```



Explanation:

1. **Correlation Matrix:**

- The `data[['age', 'bmi', 'hba1c_level', 'blood_glucose_level']].corr()` function computes the pairwise correlation coefficients between the selected numerical features (`age`, `bmi`, `hba1c_level`, and `blood_glucose_level`).
- The correlation matrix provides a measure of how strongly pairs of variables are related to each other. Values range from -1 to 1:
  - **1**: Perfect positive correlation (as one variable increases, so does the other)
  - **0**: No correlation
  - **-1**: Perfect negative correlation (as one variable increases, the other decreases)

2. **Heatmap:**

- A heatmap is plotted using `sns.heatmap()`, which visualizes the correlation matrix. The `annot=True` argument displays the correlation values within each cell of the heatmap
- The `cmap='coolwarm'` parameter is used to apply a colormap, where red indicates positive correlations, blue indicates negative correlations, and white indicates little to no correlation

- The figure size is controlled by `figsize=(8, 6)`, and `linewidths=0.5` adds spacing between the cells for better readability

---

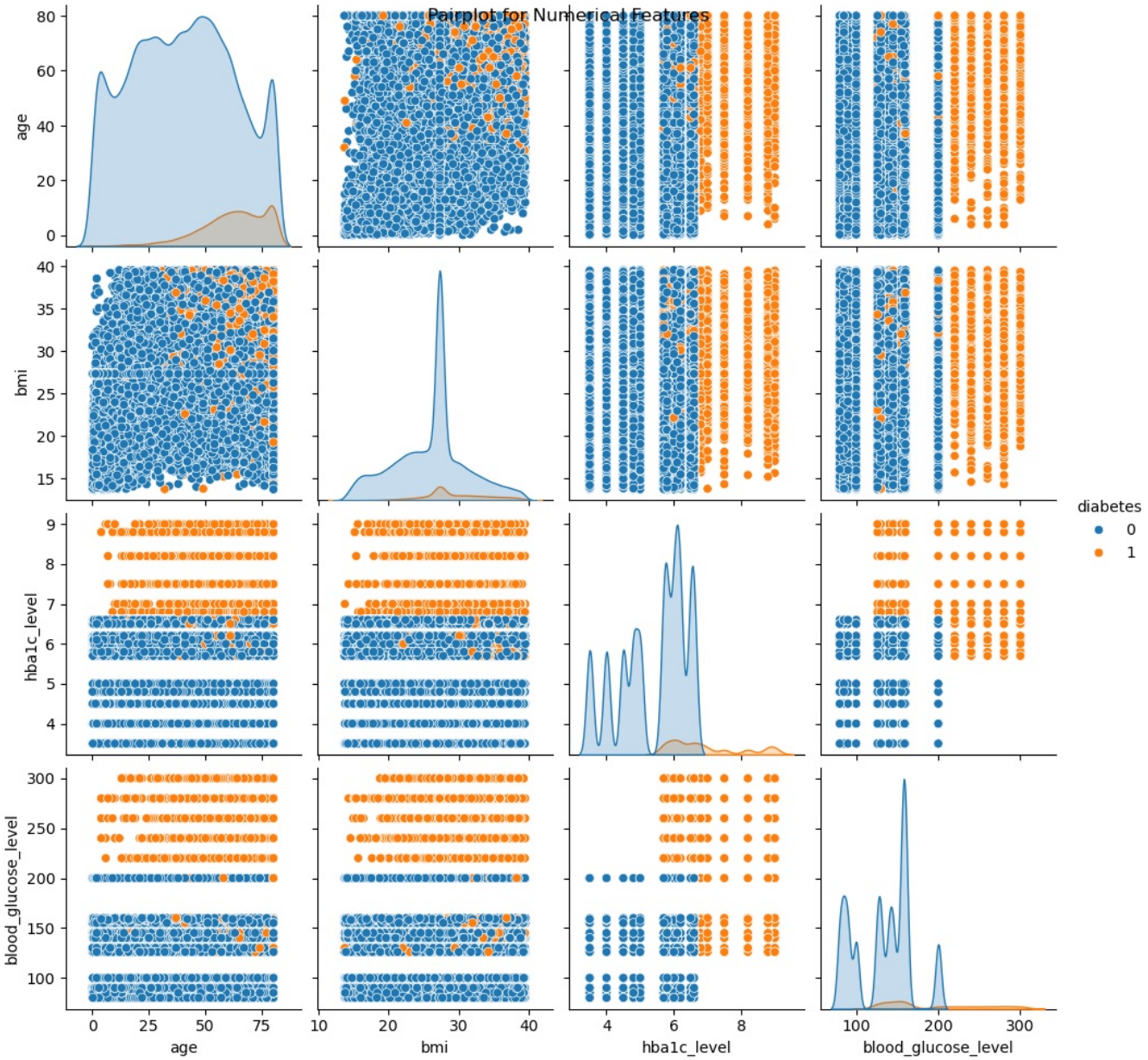
Graph Interpretation:

1. **Age and BMI:**
   - The correlation between `age` and `bmi` is moderate, with a value of **0.41**. This suggests that older individuals tend to have slightly higher BMI values
2. **Age and HbA1c Level:**
   - The correlation between `age` and `hba1c_level` is relatively low (**0.1**), indicating a weak positive relationship between age and HbA1c levels
3. **Age and Blood Glucose Level:**
   - The correlation between `age` and `blood_glucose_level` is also weak (**0.11**), meaning there is a slight positive association between age and blood glucose levels, but not very significant
4. **BMI and Blood Glucose Level:**
   - The correlation between `bmi` and `blood_glucose_level` is very weak (**0.077**), indicating almost no relationship between these two variables
5. **HbA1c Level and Blood Glucose Level:**
   - There is a modest correlation of **0.16** between `hba1c_level` and `blood_glucose_level`, suggesting that individuals with higher HbA1c levels tend to have slightly higher blood glucose levels

Pairplot (Multivariate Analysis)

Done By Dasarla Akshay Kumar UBIT Name: adasarla UBIT number: 50592353

In [40]:
```
# Pairplot of numerical features
sns.pairplot(data[['age', 'bmi', 'hba1c_level', 'blood_glucose_level', 'diabetes']], hue='diabetes')
plt.suptitle('Pairplot for Numerical Features')
plt.show()
```



Explanation:

1. **Pairplot:**
   - The pairplot function `sns.pairplot()` is used to plot pairwise relationships between numerical features. This includes scatter plots for feature combinations and kernel density plots (KDE) along the diagonal for individual feature distributions
   - The `hue='diabetes'` argument colors the data points based on the `diabetes` column, In this case, it separates individuals with and without diabetes using different colors (e.g., blue for 0, orange for 1)
   - The `data[['age', 'bmi', 'hba1c_level', 'blood_glucose_level', 'diabetes']]` subset includes the main features of interest (`age` , `bmi` , `hba1c_level` , `blood_glucose_level`) along with the target variable `diabetes`

Graph Interpretation:

1. **Age vs BMI:**
   - The scatter plot of `age` vs `bmi` shows a relatively wide distribution. While the data is fairly spread, individuals with diabetes (orange) appear to cluster more in older age groups, suggesting a higher likelihood of diabetes with age and possibly higher BMI values
2. **Age vs HbA1c Level:**
   - The scatter plot shows a more defined band of HbA1c levels, especially around the 6-8 range. Diabetic individuals are mainly concentrated in this band, which is expected as higher HbA1c levels indicate poorer blood sugar control
3. **Age vs Blood Glucose Level:**
   - The data points for blood glucose levels show a fairly flat distribution across age. However, individuals with diabetes tend to show higher glucose levels (above 140), while non-diabetics are more concentrated below 140
4. **BMI vs HbA1c Level:**

- This plot shows some clustering in the middle BMI range (around 25-30), with diabetic individuals tending to have higher HbA1c levels (above 6). This suggests that people with higher BMI are more likely to have elevated HbA1c levels

5. **BMI vs Blood Glucose Level:**

   - Individuals with diabetes are scattered across the BMI spectrum, but most show higher blood glucose levels (above 140). Non-diabetics tend to be in the lower glucose range, irrespective of their BMI

6. **HbA1c Level vs Blood Glucose Level:**

   - There's a clear pattern where individuals with diabetes show a strong concentration of higher HbA1c and blood glucose levels, further confirming that HbA1c is a good indicator of diabetes status

7. **KDE Plots:**

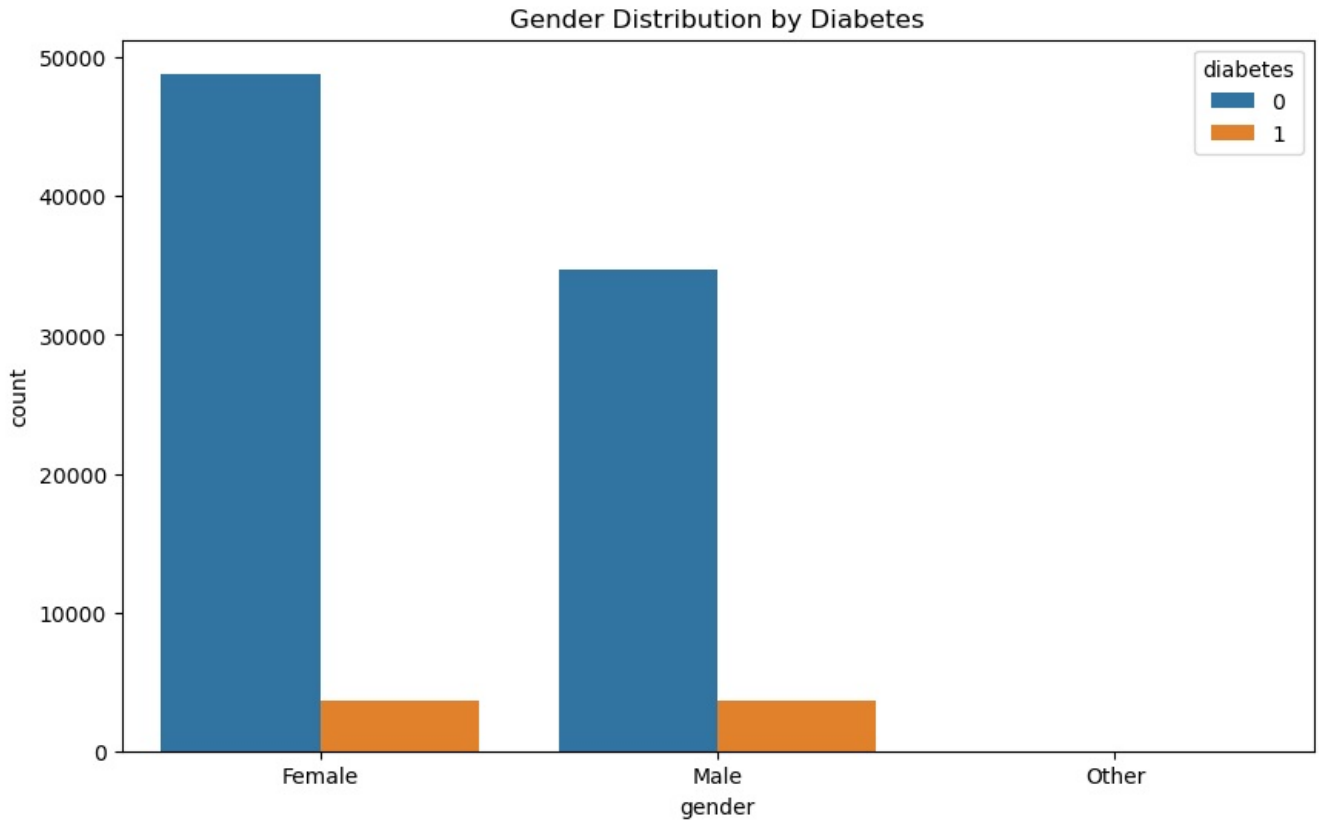
   - The kernel density plots on the diagonal show the distribution of each feature. For example, the HbA1c level distribution for diabetic individuals is shifted to the right, indicating higher values compared to non-diabetic individuals

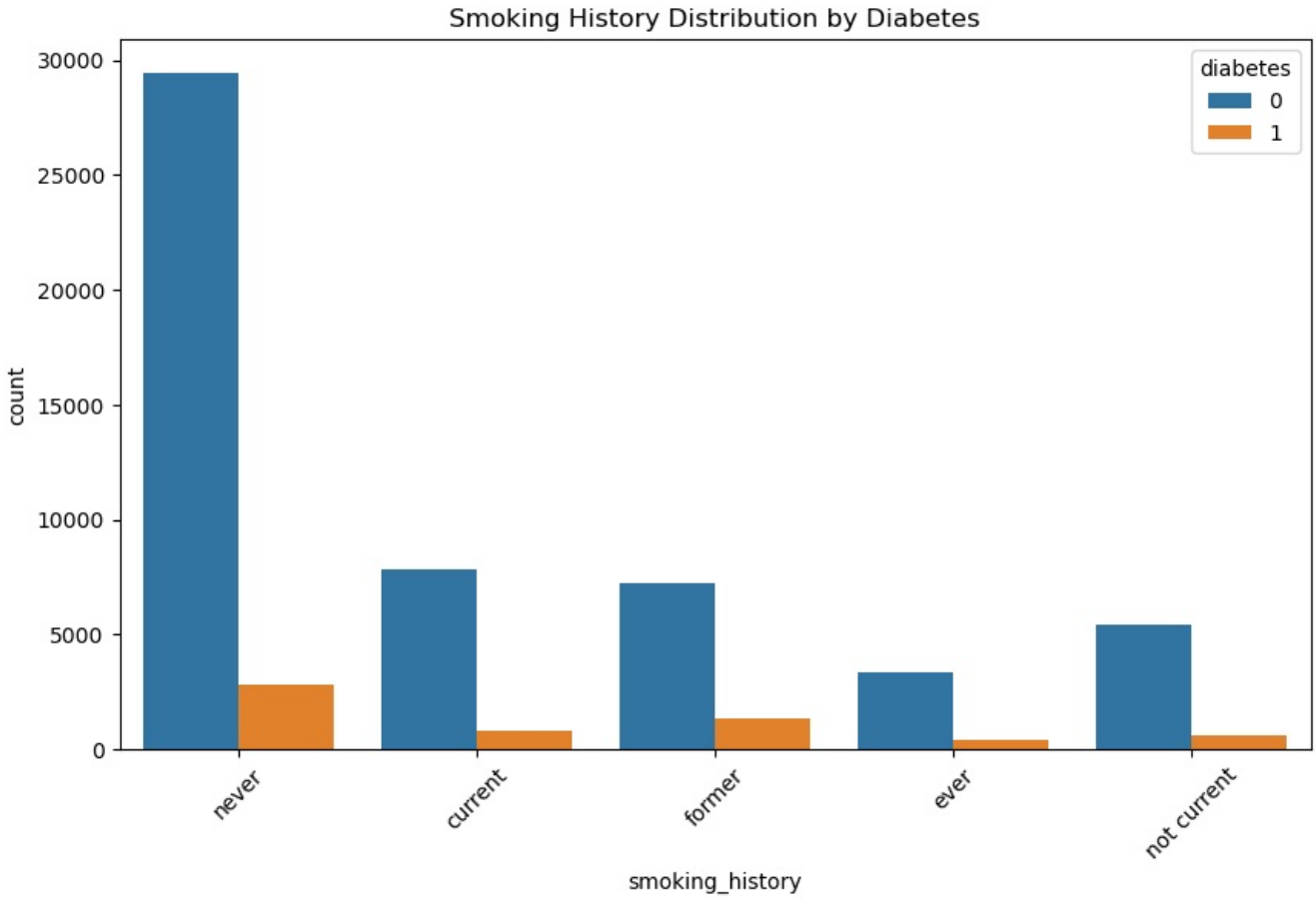
Bar Chart for Categorical Features

Done by KOSARAJU SAI SOHAN

In [43]:
```
# Countplot for categorical features
plt.figure(figsize=(10, 6))
sns.countplot(x='gender', hue='diabetes', data=data)
plt.title('Gender Distribution by Diabetes')
plt.show()

plt.figure(figsize=(10, 6))
sns.countplot(x='smoking_history', hue='diabetes', data=data)
plt.title('Smoking History Distribution by Diabetes')
plt.xticks(rotation=45)
plt.show()
```





Graph 1: Gender Distribution by Diabetes

Explanation:

- The first bar chart visualizes the distribution of individuals across different genders (`Female` , `Male` , `Other`) and how diabetes prevalence (`diabetes` column) differs within each gender group.
- The bars are grouped by diabetes status:
  - **Blue**: Individuals without diabetes (diabetes = 0)
  - **Orange**: Individuals with diabetes (diabetes = 1)

Graph Interpretation:

1. **Female**:
   - The majority of females do not have diabetes, as indicated by the large blue bar
   - A small portion of females (represented by the orange bar) have diabetes, but this number is significantly smaller compared to non-diabetics
2. **Male**:
   - There are fewer males than females in the dataset overall
   - Similar to females, the majority of males do not have diabetes (blue bar), while a smaller number of males have diabetes (orange bar)
3. **Other**:
   - The `Other` gender group has very few individuals, and the distribution shows a low count of both diabetics and non-diabetics

Graph 2: Smoking History Distribution by Diabetes

Explanation:

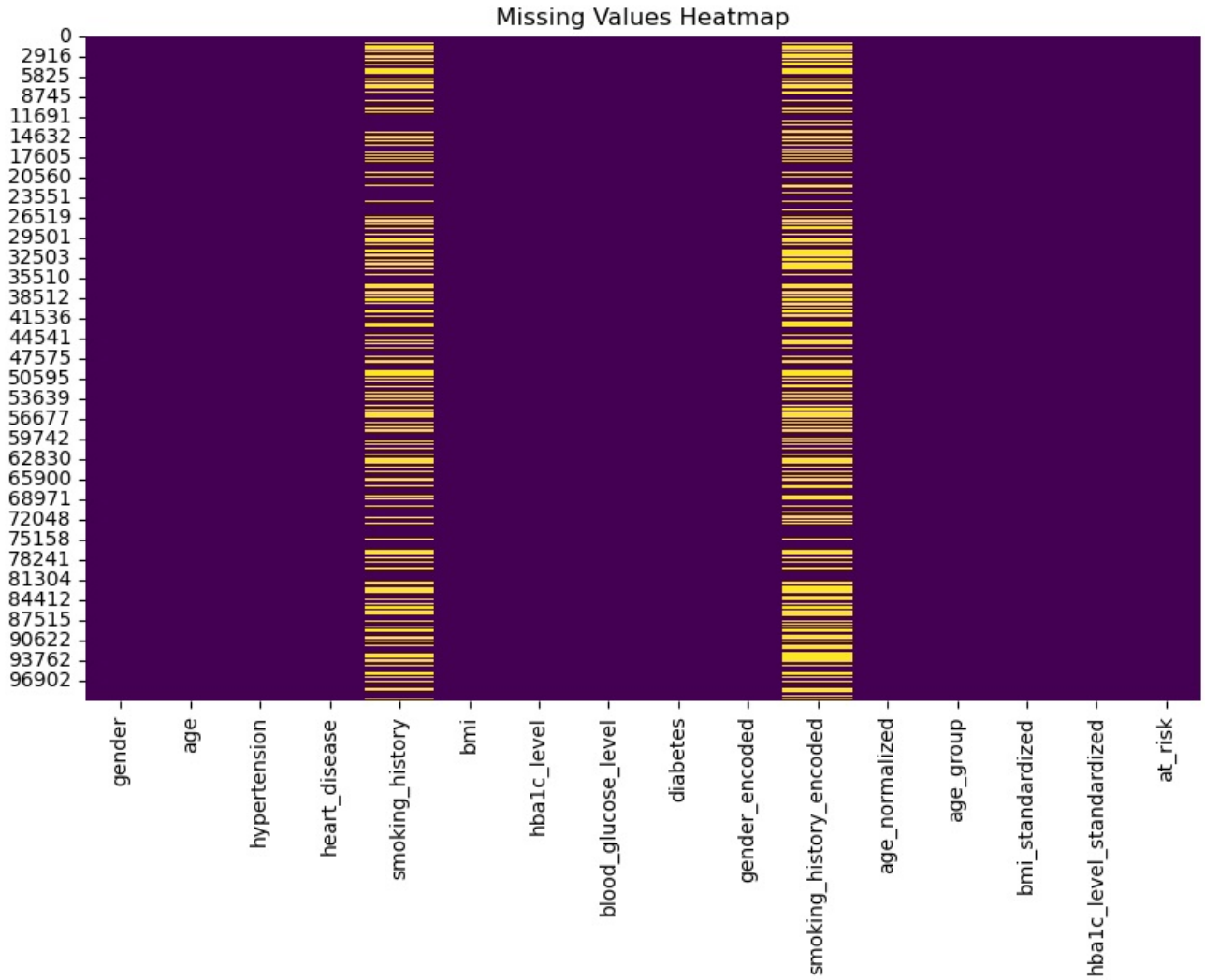
- The second bar chart shows the distribution of individuals by their smoking history and how it relates to their diabetes status
- The bars are grouped by diabetes status:
  - **Blue**: Individuals without diabetes (diabetes = 0)
  - **Orange**: Individuals with diabetes (diabetes = 1)

Graph Interpretation:

1. **Never Smokers**:
   - A large portion of individuals in this category have never smoked, and most of them do not have diabetes (blue bar)
   - A smaller proportion of never smokers have diabetes (orange bar).
2. **Current Smokers**:
   - A noticeable portion of current smokers are non-diabetic (blue bar), while a smaller portion have diabetes (orange bar)
3. **Former Smokers**:
   - The distribution of former smokers follows a similar pattern, with more non-diabetics (blue) than diabetics (orange)
4. **Ever Smokers**:
   - This group is smaller compared to other smoking categories, but the same pattern holds: more non-diabetics than diabetics
5. **Not Current Smokers**:
   - This group has a relatively low number of individuals, with a majority being non-diabetic and a small number being diabetic

KOSARAJU SAI SOHAN

In [46]:
```
# Heatmap for missing values
plt.figure(figsize=(10, 6))
sns.heatmap(data.isnull(), cbar=False, cmap='viridis')
plt.title('Missing Values Heatmap')
plt.show()
```



Missing Values Heatmap

Explanation:

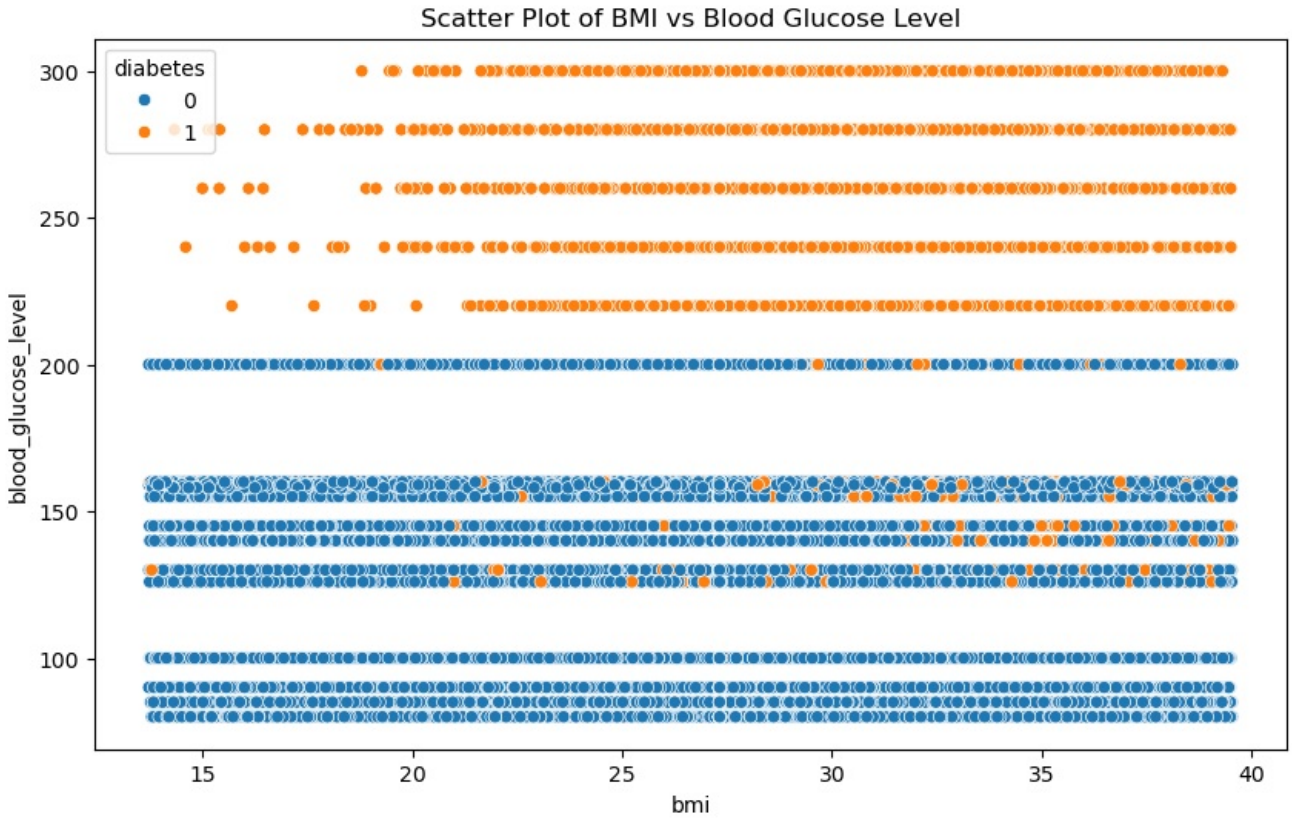
- This heatmap visualizes missing data across different columns in the dataset
- Each row represents a record (or individual) in the dataset, and each column corresponds to a feature (e.g., `gender`, `age`, `smoking_history`)
- The heatmap highlights missing values in **yellow**, while non-missing (available) values are represented by **purple**

Graph Interpretation:

1. **Smoking History**:
   - The `smoking_history` column shows a significant number of missing values, as indicated by the numerous yellow stripes. This feature is the most incomplete in the dataset
2. **Other Features**:
   - The remaining columns (such as `gender`, `age`, `hypertension`, `heart_disease`, `bmi`, `blood_glucose_level`, etc.) do not exhibit any missing values, as represented by the continuous purple bars. This means these features are fully populated with data

KOSARAJU SAI SOHAN

In [49]:
```
# Scatter plot between BMI and Blood Glucose Level
plt.figure(figsize=(10, 6))
sns.scatterplot(x='bmi', y='blood_glucose_level', hue='diabetes', data=data)
plt.title('Scatter Plot of BMI vs Blood Glucose Level')
plt.show()
```



Scatter Plot: BMI vs Blood Glucose Level

Explanation:

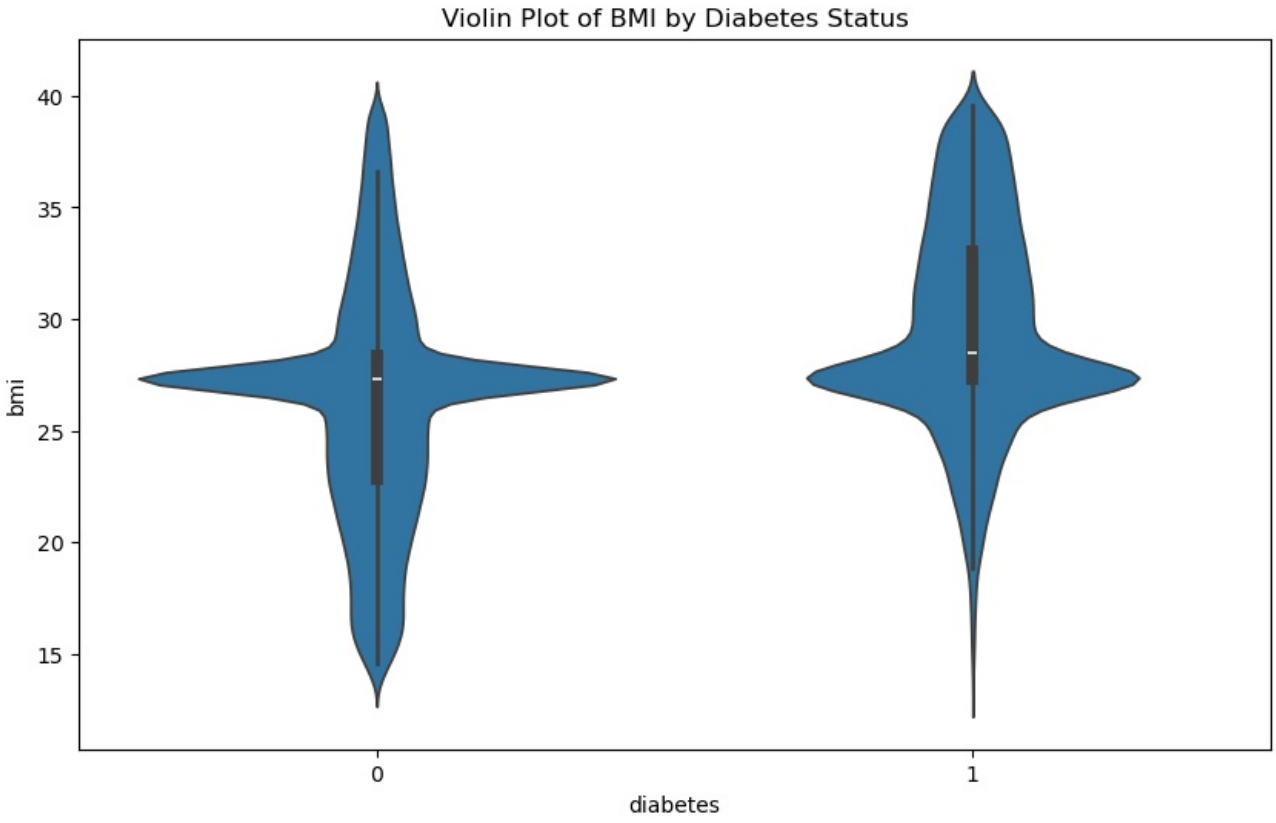
- The scatter plot visualizes the relationship between **BMI** (x-axis) and **blood glucose level** (y-axis) across individuals
- The data points are color-coded based on the diabetes status:
  - **Blue dots** represent individuals without diabetes (diabetes = 0)
  - **Orange dots** represent individuals with diabetes (diabetes = 1)

Graph Interpretation:

1. **Non-Diabetics (Blue Dots)**:
   - The non-diabetic group is primarily clustered in the lower blood glucose levels (below 150 mg/dL), regardless of their BMI
   - This suggests that most non-diabetic individuals maintain a relatively stable blood glucose level, irrespective of BMI variations
2. **Diabetics (Orange Dots)**:
   - Diabetic individuals are concentrated in the higher blood glucose range (above 150 mg/dL), with many having blood glucose levels exceeding 200 mg/dL
   - Unlike the non-diabetic group, diabetics show greater variability in both BMI and blood glucose levels, with many maintaining higher blood glucose levels regardless of their BMI values
3. **BMI Range**:
   - BMI values range from around 15 to 40, and individuals with both diabetes and non-diabetes are spread across this range
   - However, BMI does not appear to be a strong determinant of blood glucose level alone, as individuals with low and high BMI exhibit both high and low blood glucose levels

Venkata Anudeep Bandi

In [52]:
```
# Violin plot for BMI across diabetes status
plt.figure(figsize=(10, 6))
sns.violinplot(x='diabetes', y='bmi', data=data)
plt.title('Violin Plot of BMI by Diabetes Status')
plt.show()
```



Violin Plot: BMI by Diabetes Status

Explanation:

- The violin plot visualizes the distribution of **BMI** across individuals with and without diabetes (x-axis = `diabetes`, y-axis = `bmi`)
- **0** on the x-axis represents individuals without diabetes, and **1** represents individuals with diabetes
- The violin plot combines a boxplot and a density plot, showing both the summary statistics (e.g., median, quartiles) and the probability distribution of the BMI values

Graph Interpretation:

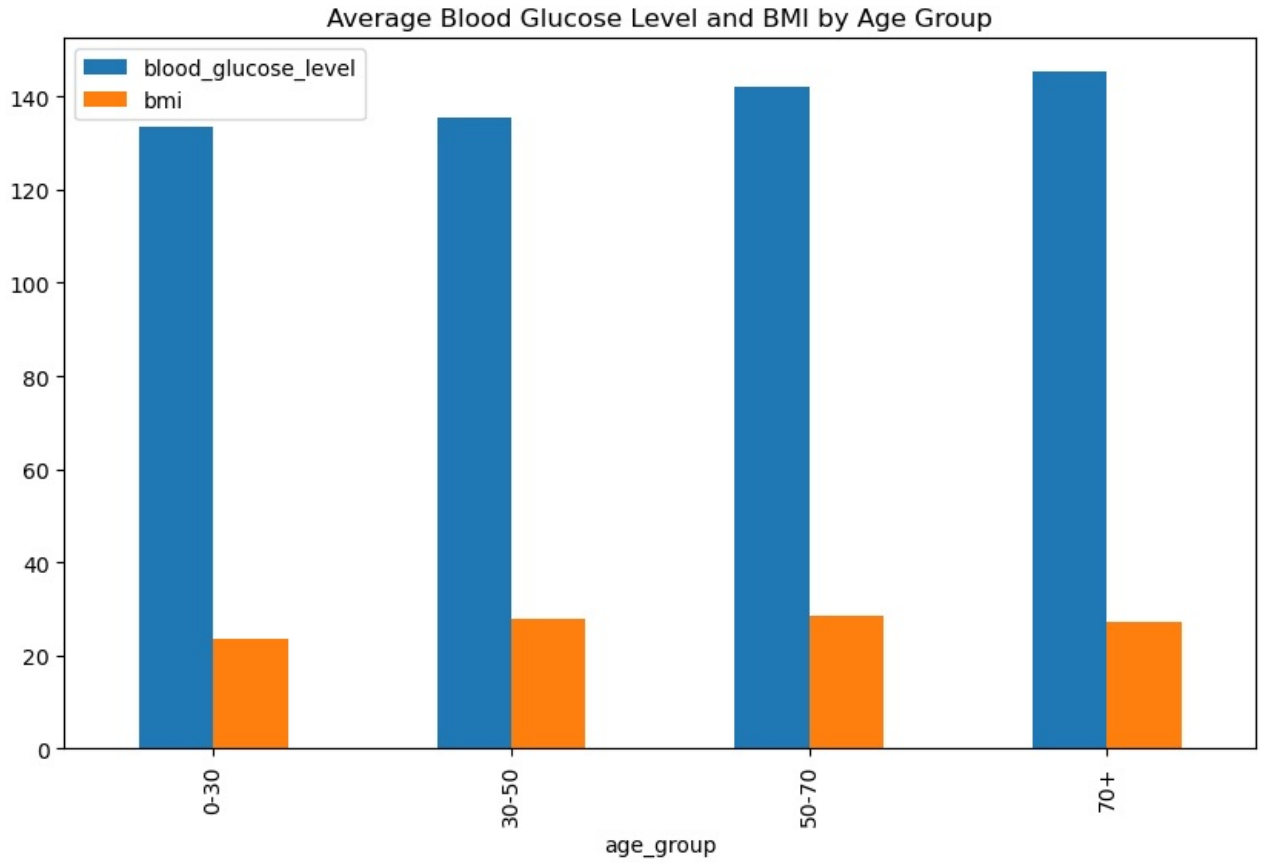
1. **Non-Diabetics (0)**:
   - The distribution of BMI among non-diabetics is relatively wide, with most individuals having a BMI between 20 and 35
   - The peak density (widest part of the plot) occurs around a BMI of 25, indicating that most non-diabetic individuals have BMI values close to this range
   - There are fewer individuals with BMI below 20 or above 35, as seen by the tapering of the plot at the extremes
2. **Diabetics (1)**:
   - Diabetic individuals also have a wide range of BMI values, with most falling between 20 and 35
   - Similar to the non-diabetic group, the peak density for diabetics is also centered around a BMI of 25
   - The distribution of BMI for diabetics appears slightly more spread out compared to non-diabetics, suggesting more variability in BMI among individuals with diabetes
3. **Comparison**:
   - The overall shape of the BMI distribution is quite similar for both diabetics and non-diabetics, with a peak around 25 BMI and a similar spread across the BMI range
   - The spread and variability in BMI values are comparable in both groups, with no significant differences in distribution shape

Venkata Anudeep Bandi

In [55]:
```
# Group by age group and calculate mean blood glucose level and BMI
grouped_data = data.groupby('age_group').agg({'blood_glucose_level': 'mean', 'bmi': 'mean'}).reset_index()

# Bar plot of aggregated data
grouped_data.plot(kind='bar', x='age_group', figsize=(10, 6))
plt.title('Average Blood Glucose Level and BMI by Age Group')
plt.show()
```

```
C:\Users\Sakshi\AppData\Local\Temp\ipykernel_485552\1818909769.py:2: FutureWarning: The default of observed=Fal
se is deprecated and will be changed to True in a future version of pandas. Pass observed=False to retain curre
nt behavior or observed=True to adopt the future default and silence this warning.
  grouped_data = data.groupby('age_group').agg({'blood_glucose_level': 'mean', 'bmi': 'mean'}).reset_index()
```



In [56]:
```
from scipy.stats import chi2_contingency

# Chi-square test for gender and diabetes
contingency_table = pd.crosstab(data['gender'], data['diabetes'])
chi2, p, dof, ex = chi2_contingency(contingency_table)
print(f'Chi-square test results for Gender and Diabetes: p-value={p}')

# Chi-square test for smoking history and diabetes
contingency_table_smoking = pd.crosstab(data['smoking_history'], data['diabetes'])
chi2_smoking, p_smoking, dof_smoking, ex_smoking = chi2_contingency(contingency_table_smoking)
print(f'Chi-square test results for Smoking History and Diabetes: p-value={p_smoking}')
```

```
Chi-square test results for Gender and Diabetes: p-value=1.3730041094833239e-39
Chi-square test results for Smoking History and Diabetes: p-value=1.3828649131361586e-79
```

Principal Component Analysis (PCA):

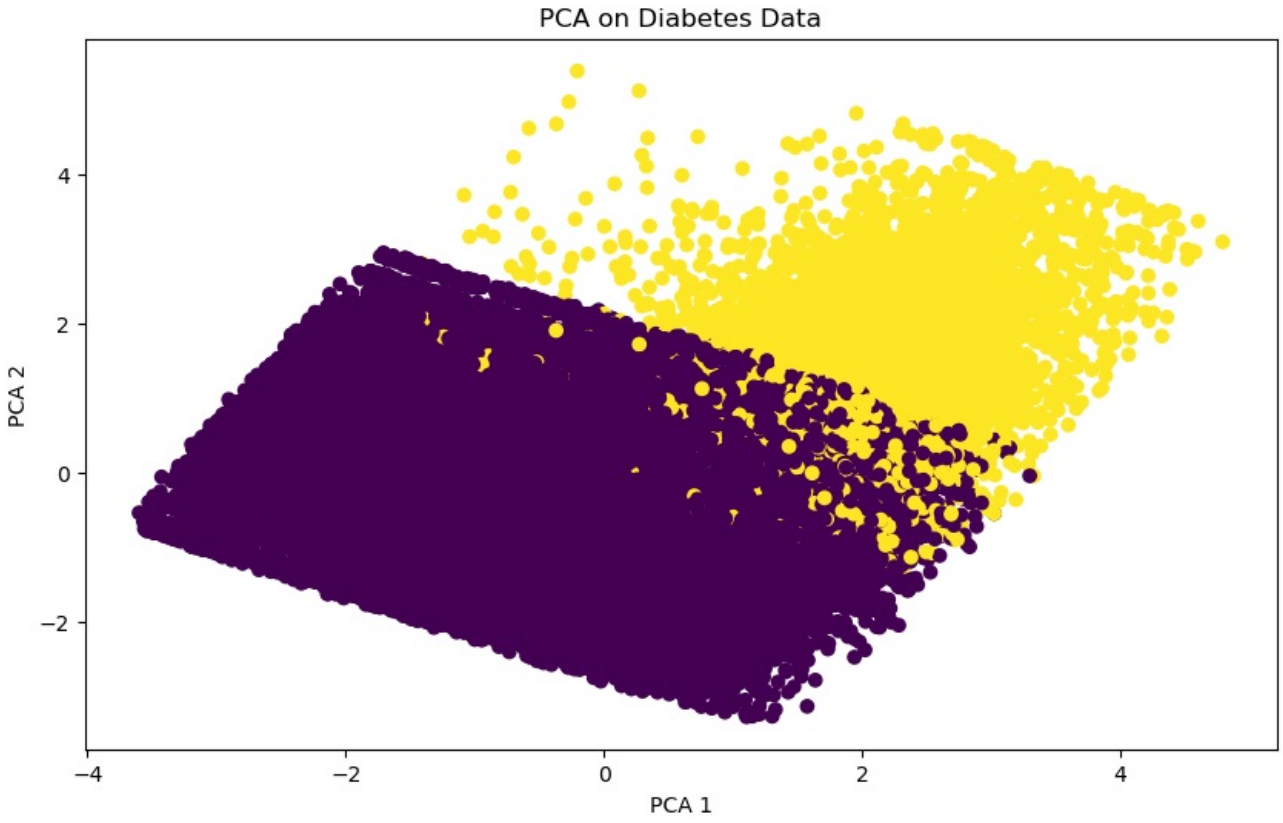
Done By Bhanu Teja Veeramachanei UBIT Name: bveerama UBIT number: 50606694

In [58]:
```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

# Scale the features
scaler = StandardScaler()
scaled_data = scaler.fit_transform(data[['age', 'bmi', 'hba1c_level', 'blood_glucose_level']])

# Perform PCA
pca = PCA(n_components=2)
pca_result = pca.fit_transform(scaled_data)

# Plot PCA results
plt.figure(figsize=(10, 6))
plt.scatter(pca_result[:, 0], pca_result[:, 1], c=data['diabetes'], cmap='viridis')
plt.title('PCA on Diabetes Data')
plt.xlabel('PCA 1')
plt.ylabel('PCA 2')
plt.show()
```



Bar Plot: Average Blood Glucose Level and BMI by Age Group

Explanation:

- This bar plot compares the **average blood glucose level** (blue bars) and **average BMI** (orange bars) across different age groups
- The x-axis represents the age groups: `0-30`, `30-50`, `50-70`, and `70+`
- The y-axis represents the average values for blood glucose levels and BMI for each age group

Graph Interpretation:

1. **Blood Glucose Level**:
   - Across all age groups, the average blood glucose level remains same, hovering around 140 mg/dL.
   - There is minimal variation in blood glucose levels between the different age groups, indicating that age does not affect average blood glucose levels
2. **BMI**:
   - The average BMI is much lower than the average blood glucose level and shows slight variation across the age groups.
   - The BMI for the `0-30` age group is slightly higher than that of the older age groups (`30-50`, `50-70`, and `70+`), but the difference is not substantial
   - Overall, BMI values remain relatively constant across all age groups, staying between 20 and 25
3. **Comparison**:
   - The blood glucose levels are significantly higher than BMI values across all age groups
   - Both BMI and blood glucose levels show limited variation between different age groups, with blood glucose levels being consistently higher in all groups

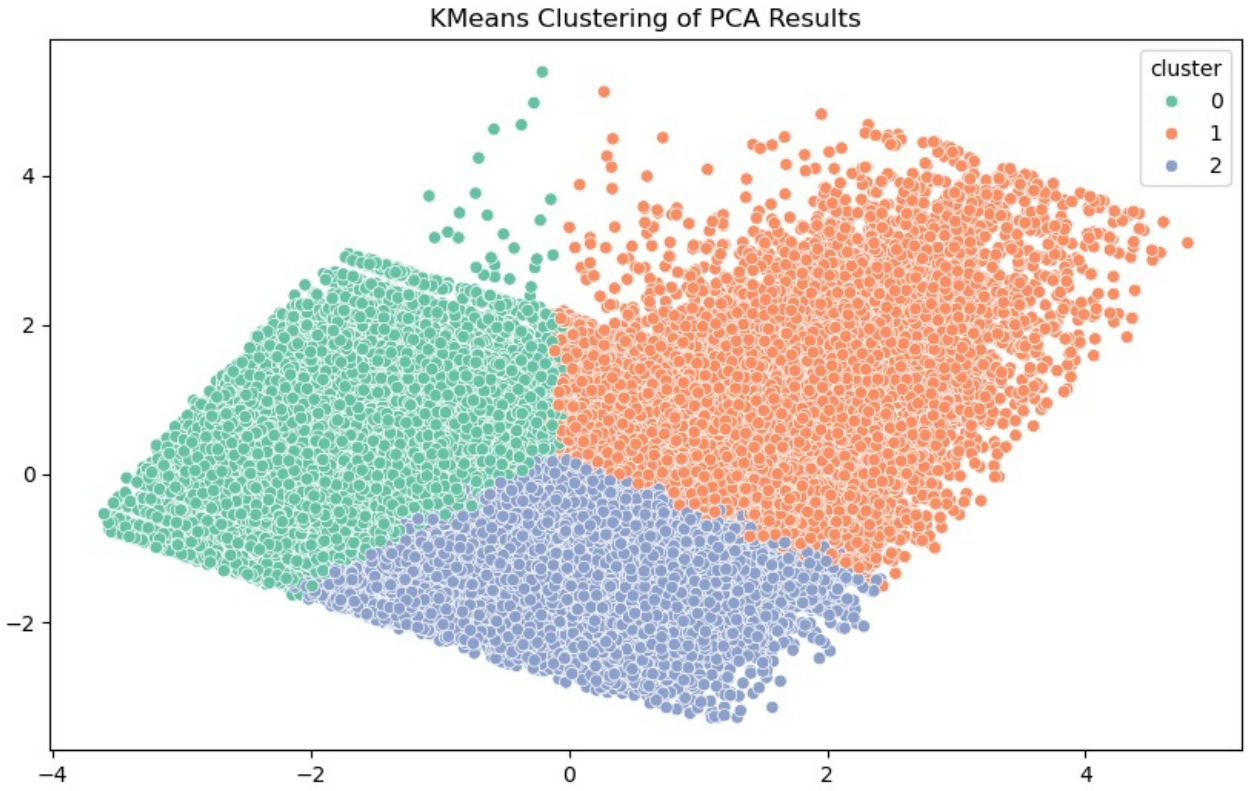
Clustering (KMeans):

Done By Bhanu Teja Veeramachanei UBIT Name: bveerama UBIT number: 50606694

In [61]:
```
from sklearn.cluster import KMeans

# Apply KMeans clustering
kmeans = KMeans(n_clusters=3)
data['cluster'] = kmeans.fit_predict(scaled_data)

# Scatter plot of clusters
plt.figure(figsize=(10, 6))
sns.scatterplot(x=pca_result[:, 0], y=pca_result[:, 1], hue=data['cluster'], palette='Set2')
plt.title('KMeans Clustering of PCA Results')
plt.show()
```



PCA Plot: Diabetes Data

Explanation:

- In this plot:
  - The x-axis represents the first principal component (**PCA 1**).
  - The y-axis represents the second principal component (**PCA 2**).
  - The colors represent different classes in the dataset:
    - **Purple** for one class (likely non-diabetic individuals).
    - **Yellow** for the other class (likely diabetic individuals).

Graph Interpretation:

1. **Separation Between Classes**:
   - The plot shows a clear separation between the two classes (purple and yellow), indicating that the PCA was successful in capturing the variance between the diabetic and non-diabetic individuals.
   - The purple points (likely representing non-diabetic individuals) are more concentrated towards the negative PCA 1 and PCA 2 regions, while the yellow points (diabetic individuals) spread out towards the positive PCA 1 and PCA 2 regions.
2. **Cluster of Diabetics**:
   - The yellow points (diabetics) are more dispersed in the upper-right region of the plot, suggesting a greater variability in the dataset for individuals with diabetes.
3. **Cluster of Non-Diabetics**:
   - The purple points (non-diabetics) form a tighter cluster in the lower-left corner, suggesting less variability in this group compared to diabetics.
4. **Overlap**:
   - Although there is some overlap between the two classes, the PCA was able to create a noticeable distinction between the diabetic and non-diabetic groups, which may be useful for classification purposes.

Outlier Detection (Isolation Forest):

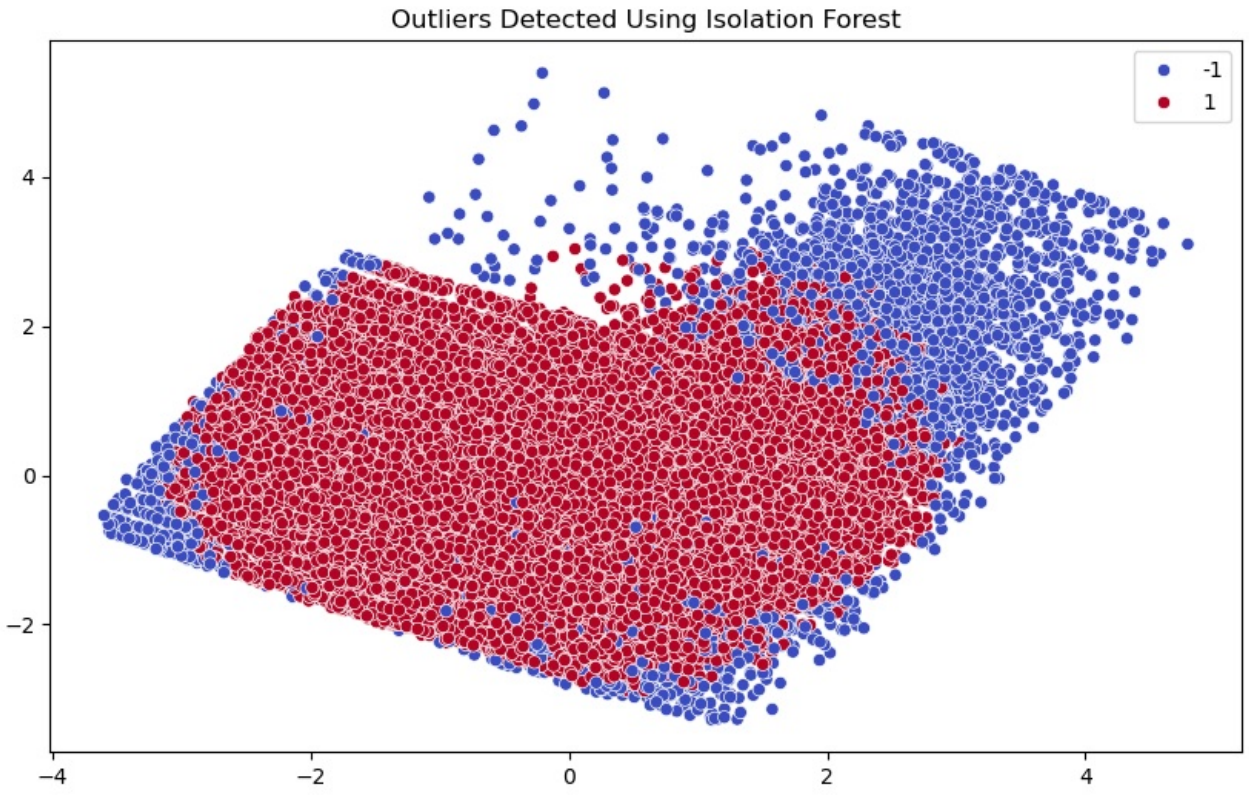
Done By Bhanu Teja Veeramachanei UBIT Name: bveerama UBIT number: 50606694

In [64]:
```
from sklearn.ensemble import IsolationForest

# Apply Isolation Forest for outlier detection
iso_forest = IsolationForest(contamination=0.05)
outliers = iso_forest.fit_predict(scaled_data)

# Scatter plot of outliers
```

```
plt.figure(figsize=(10, 6))
sns.scatterplot(x=pca_result[:, 0], y=pca_result[:, 1], hue=outliers, palette='coolwarm')
plt.title('Outliers Detected Using Isolation Forest')
plt.show()
```



Outliers Detected Using Isolation Forest

Explanation:

- In the plot:
  - **Blue points** (label `-1`) represent data points classified as **outliers**
  - **Red points** (label `1`) represent data points classified as **normal observations**
  - The x-axis represents the first principal component (**PCA 1**) and the y-axis represents the second principal component (**PCA 2**), as in a PCA plot used to reduce the dimensionality of the dataset

Graph Interpretation:

1. **Outliers (Blue Points)**:
   - The blue points are scattered throughout the dataset but tend to be more concentrated towards the edges of the distribution
   - Some blue points also appear within the denser clusters (red regions), indicating that these points are slightly different from the majority of observations in that area
2. **Normal Observations (Red Points)**:
   - The majority of data points are classified as normal observations (red points), forming a dense cluster in the center of the plot
   - These points represent the main structure of the dataset, with normal values grouped together in the middle of the PCA space

Feature Selection (SelectKBest):

Done By Bhanu Teja Veeramachanei UBIT Name: bveerama UBIT number: 50606694

In [67]:
```
from sklearn.feature_selection import SelectKBest, f_classif

# Feature selection using SelectKBest
X = data[['age', 'bmi', 'hba1c_level', 'blood_glucose_level']]
y = data['diabetes']
best_features = SelectKBest(score_func=f_classif, k=2).fit(X, y)

# Get scores for features
scores = best_features.scores_
print(f'Feature scores: {scores}')
```

```
Feature scores: [ 6646.65299628  3016.68483419 16493.23831913 18534.76835345]
```

Regression Analysis (Logistic Regression):

Done By Bhanu Teja Veeramachanei UBIT Name: bveerama UBIT number: 50606694

In [69]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report

# Logistic regression on selected features
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
log_reg = LogisticRegression()
log_reg.fit(X_train, y_train)

# Predictions and evaluation
y_pred = log_reg.predict(X_test)
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.97      0.99      0.98     25059
           1       0.88      0.61      0.72      2179

    accuracy                           0.96     27238
   macro avg       0.92      0.80      0.85     27238
weighted avg       0.96      0.96      0.96     27238
```

Bhanu Teja Veeramachaneni

UBIT Name: `bveerama`

UBIT Number: `50606694`

Hypothesis 1

How do various combinations of clinical indicators (e.g., BMI, hypertension, heart disease, HbA1c level) interact to influence the risk of diabetes?

Significance:

Because clinical markers including body mass index (BMI), hypertension, and heart disease are well-established indications of diabetes risk, this question is crucial. Nonetheless, a more complete picture of diabetes risk can be obtained by comprehending how these variables interact with one another in combination. It might show if the chance of getting diabetes rises exponentially with the number of risk factors or whether some risk factors are more important than others.

Objective:

The purpose is to evaluate how different combinations of clinical markers affect the likelihood of diabetes. Does someone who has high blood pressure and a high BMI, for instance, have a higher chance of developing diabetes than someone who only has one of these conditions? This approach assists in creating risk profiles and could be used to adapt preventative health strategies more effectively. More individualized healthcare suggestions may also be supported by insights into the relationships between various parameters.

In [73]:
```
import pandas as pd

bhanu_df = pd.read_csv("diabetes_prediction_dataset.csv")

data.info()
data.head()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 90792 entries, 0 to 99999
Data columns (total 17 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   gender                    90792 non-null  object
 1   age                       90792 non-null  float64
 2   hypertension              90792 non-null  category
 3   heart_disease             90792 non-null  category
 4   smoking_history           59212 non-null  object
 5   bmi                       90792 non-null  float64
 6   hba1c_level               90792 non-null  float64
 7   blood_glucose_level       90792 non-null  int64
 8   diabetes                  90792 non-null  int64
 9   gender_encoded            90774 non-null  float64
 10  smoking_history_encoded   49490 non-null  float64
 11  age_normalized            90792 non-null  float64
 12  age_group                 90792 non-null  category
 13  bmi_standardized          90792 non-null  float64
 14  hba1c_level_standardized  90792 non-null  float64
 15  at_risk                   90792 non-null  int32
 16  cluster                   90792 non-null  int32
dtypes: category(3), float64(8), int32(2), int64(2), object(2)
memory usage: 10.0+ MB
```

Out[73]:

	gender	age	hypertension	heart_disease	smoking_history	bmi	hba1c_level	blood_glucose_level	diabetes	gender_encoded	smoking_
0	Female	80.0	No	Yes	never	25.19	6.6	140	0	0.0	
1	Female	54.0	No	No	<NA>	27.32	6.6	80	0	0.0	
2	Male	28.0	No	No	never	27.32	5.7	158	0	1.0	
3	Female	36.0	No	No	current	23.45	5.0	155	0	0.0	
4	Male	76.0	Yes	Yes	current	20.14	4.8	155	0	1.0	

In [ ]:
```python
# Replace 'No Info' with NaN in the 'smoking_history' column
bhanu_df['smoking_history'] = bhanu_df['smoking_history'].replace('No Info', pd.NA)

# Remove duplicate rows (if any)
data_cleaned = bhanu_df.drop_duplicates()

# Verify data after cleaning
print(f"Original shape: {bhanu_df.shape}")
print(f"Cleaned shape: {data_cleaned.shape}")
```

In [ ]:
```python
import matplotlib.pyplot as plt
import seaborn as sns

# Set plot style for better readability
sns.set(style="whitegrid")

# Correlation heatmap of clinical indicators and diabetes
plt.figure(figsize=(10, 6))
sns.heatmap(data_cleaned[['bmi', 'hypertension', 'heart_disease', 'HbA1c_level', 'diabetes']].corr(), annot=Tru
plt.title("Correlation Heatmap: Clinical Indicators and Diabetes Risk")
plt.show()
```

Correlation Heatmap: Clinical Indicators and Diabetes Risk

This heatmap illustrates the correlation between clinical indicators (BMI, hypertension, heart disease, HbA1c level) and diabetes:

- **BMI** has the strongest positive correlation with diabetes (0.21), followed by hypertension (0.20) and heart disease (0.17).
- **HbA1c level** shows the highest correlation with diabetes at 0.41, indicating its importance as a clinical indicator for diabetes risk.
- The interactions between other clinical factors like hypertension and heart disease show relatively weak correlations, indicating they might not interact strongly to predict diabetes on their own but are important factors.

Hypothesis 2

How do lifestyle factors (e.g., smoking history, physical inactivity) affect the onset and progression of diabetes across different demographic groups (age, gender)?

Significance:

Lifestyle decisions like smoking and not exercising are proven to have a big influence on one's health. Studying the ways in which these variables influence the occurrence of diabetes in different populations can yield significant findings about preventive medicine. For instance, younger groups could have distinct risk profiles compared to older ones, and recognizing this variance is crucial to devising

age-specific health interventions. Similarly, gender-sensitive health strategies may be influenced by variations in how lifestyle decisions affect diabetes.

Objective:

This question intends to study if lifestyle variables effect the onset and course of diabetes differently in varied demographic groups. In this way, the study can assist create targeted interventions by generating hypotheses about which demographic groups are more susceptible to diabetes caused by lifestyle choices. For instance, it may suggest that early treatments are essential for some populations to avoid the onset of diabetes if smoking and physical inactivity have a greater negative impact on younger guys than older females.

In [ ]:
```
# Bar plot for smoking history vs diabetes across gender
plt.figure(figsize=(12, 6))
sns.barplot(x='smoking_history', y='diabetes', hue='gender', data=data_cleaned, estimator=lambda x: sum(x)/len(
plt.title("Impact of Smoking History on Diabetes Across Gender")
plt.ylabel("Proportion with Diabetes")
plt.show()
```

Impact of Smoking History on Diabetes Across Gender

This bar plot shows how smoking history affects diabetes risk across gender:

- **Former smokers** in both males and females show the highest proportion of diabetes, with males having a slightly higher diabetes rate in this group.
- **Current smokers** and those who have “never smoked” show similar, moderate diabetes rates across genders.
- Overall, **males** consistently show a higher proportion of diabetes across all smoking history categories compared to females.

Dasarla Akshay Kumar , 50592353

Hypothesis - 1 : Does an individual's BMI correlate with their likelihood of developing diabetes?

Body Mass Index (BMI) is an important health metric, it is recognized as an indicator of body fat and a strong predictor of metabolic conditions. Since obesity and high BMI values are often related to an increased risk of developing chronic conditions such as diabetes, this hypothesis aims to examine whether a higher BMI is corelated with a higher prevalence of diabetes.

One of the most common chronic diseases in the world is diabetes, and early detection and prevention depend on knowing risk factors like BMI. If a substantial correlation is discovered between BMI and diabetes, this would assist medical practitioners in creating more specialized interventions and preventative plans for those with higher BMIs. Verifying this association may further support the use of BMI as a critical component in predictive models for diabetes detection, increasing the precision of the diagnosis.

In [ ]:
```
df_akshay = pd.read_csv('/Users/bhanuteja/Desktop/DIC project/diabetes_prediction_dataset.csv')

# Hypothesis 1: Higher BMI and Diabetes
# Filter relevant columns and remove rows with missing BMI or diabetes information
df_bmi = df_akshay[['gender', 'bmi', 'diabetes']].dropna()
```

Step 1: Data Cleaning and Preparation

Before analyzing the data, we must ensure that the BMI column is cleaned and prepared for analysis. This includes:

In [ ]:
```
print(df_akshay['bmi'].isnull().sum())


df_akshay_cleaned = df_akshay.dropna(subset=['bmi'])
```

In [ ]:
```
plt.figure(figsize=(10, 6))
df_akshay_cleaned['bmi'].hist(bins=20, color='skyblue')
plt.title('BMI Distribution')
plt.xlabel('BMI')
plt.ylabel('Frequency')
plt.show()
```

Step 2: Visualization of BMI Distribution Across Diabetic and Non-Diabetic Individuals

In [ ]:
```
plt.figure(figsize=(10, 6))
sb.boxplot(x='diabetes', y='bmi', data=df_akshay_cleaned, palette="Set2")
plt.title('BMI Distribution for Individuals With and Without Diabetes')
plt.xlabel('Diabetes (0 = No, 1 = Yes)')
plt.ylabel('BMI')
plt.show()
```

In [ ]:
```
df_akshay['gender'] = df_akshay_cleaned['gender'].map({0: 'Female', 1: 'Male'})
```

In [ ]:
```
df_akshay['gender'] = df_akshay_cleaned['gender'].map({0: 'Female', 1: 'Male'})


plt.figure(figsize=(10, 6))
sb.boxplot(x='diabetes', y='bmi', hue='gender', data=df_akshay_cleaned, palette="Set2")
plt.title('BMI Distribution for Males and Females With and Without Diabetes')
plt.xlabel('Diabetes (0 = No, 1 = Yes)')
plt.ylabel('BMI')
plt.legend(title='Gender')
plt.show()
```

The average BMI of people with and without diabetes was compared in order to test this hypothesis. The BMI distribution for these two groups was plotted as a box plot, highlighting the overall spread and center tendency. According to the idea, those with diabetes should have a higher median BMI than people without the disease.

Conclusion for hypothesis 1

After conducting the EDA that comaprision of BMI with people having diabetis and people not having diabetis we conclude these below points -

1 . Higher BMI value is associated with increased likelihood of Diabetis

For this we plotted the above graphs where people having diabetis is having high BMI values . This is common for both males and females as we plotted separately in the second graph . Thus high BMI values are associated with increased likelihood of diabetis.

2 . Obesity is the factor of causing diabetis

Higher BMI values denotes that high fat percentage in the body which means the person is obesed so irrespective of gender obesity can increase the likelihood of causing diabetis .

Hypothesis 2 - Does age impact the prevalence of diabetes in individuals?

Age is one of the major factor that increases the likelihood of causing diabetis because as the age increases there would be many factors that will affect our health . In that reduced metabolic efficiency, higher rates of insulin resistance are some cases . So we see old people having diabetis .

We justify this hypothesis by showing the diabetis causing in different age groups like e.g., 20-30, 30-40, etc. Our goal is to prove that as we are getting old there would be more probability of cause diabetis

In [ ]:
```
print(df_akshay_cleaned['age'].describe())
```

Step 1: Convert Normalized Age Values to Real Age Values

In case the age values are normalized, we first need to reverse the normalization to get back to real age values using the formula:

Real Age

(Normalized Age × Standard Deviation) + Mean Real Age=(Normalized Age×Standard Deviation)+Mean

Step 2: Create Age Groups and Visualisation

Once the real ages are obtained, we categorize the individuals into age groups, such as 20-30, 30-40, etc.

In [ ]:
```
df_diabetes = df_akshay[df_akshay['diabetes'] == 1]

# Grouping by age group and counting the number of people with diabetes
diabetes_counts = df_diabetes.groupby('age_group').size()

# Plotting the results
plt.figure(figsize=(10, 6))
sns.barplot(x=diabetes_counts.index, y=diabetes_counts.values)
plt.title('Number of People with Diabetes Across Age Groups')
plt.xlabel('Age Group')
plt.ylabel('Number of People with Diabetes')
plt.show()
```

Conclusion for hypothesis 2

After conducting EDA on the relationship between age and diabetes prevalence, the following conclusions were drawn:

1. Higher Prevaliance of Diabetis in older age groups

As we saw from the analysis that as the age is increasing there is more likelihood of people for causing diabetis . This is true that the change in health factors like reduced metabolic efficiency, higher rates of insulin resistance will affect the cause of diabetis

1. Age is risk factor for causing diabetis

A person's age significantly influences their risk of developing diabetes. People's metabolic processes slow down with age, increasing the risk of diseases like insulin resistance, which can develop into diabetes

SAI SOHAN KOSARAJU

UBIT Name: `saisohan`

UBIT Number: `50560534`

Question 1:

"How does the duration of elevated blood glucose levels impact the progression from prediabetes to diabetes, and what is the critical time frame for intervention?"

Significance: This study aims to determine the critical point at which prediabetes transitions into full-blown diabetes. Understanding this pivotal moment will enable healthcare providers to intervene earlier and prevent the onset of the disease.

Objective: By analyzing the correlation between the duration of elevated blood sugar and the progression to diabetes, this research seeks to identify the optimal timing for effective interventions.

Potential Hypotheses:

- *Hypothesis 1:* Individuals who maintain elevated blood sugar levels for more than three years are significantly more likely to develop diabetes compared to those with shorter periods of elevated levels.
- *Hypothesis 2:* Regular monitoring and early interventions can help prevent the progression from prediabetes to diabetes.

Question 2:

"Is there a specific BMI threshold beyond which the risk of developing diabetes increases exponentially, and does this threshold differ based on gender and age?"

Research Question: Is there a specific body mass index (BMI) threshold beyond which the risk of developing diabetes significantly increases, and does this threshold vary by gender and age?

Significance: While BMI is a recognized risk factor for diabetes, the relationship between the two may not be straightforward. This study aims to explore the possibility of a tipping point where the risk of diabetes dramatically escalates. Understanding how this threshold differs between genders and age groups can lead to more tailored healthcare recommendations.

Objective: The objective is to identify a non-linear relationship between BMI and diabetes risk. Specifically, we seek to determine if there is a critical BMI value that significantly elevates the risk of diabetes, and whether this threshold varies based on gender or age. This analysis could result in more precise weight management guidelines for diabetes prevention, tailored to individual characteristics.

Potential Hypotheses:

- *Hypothesis 1:* A specific BMI threshold exists (e.g., 32) beyond which the risk of developing diabetes rapidly increases.
- *Hypothesis 2:* The BMI threshold for elevated diabetes risk may be lower for women than for men, or it may increase at a higher BMI for older individuals compared to younger ones.

Visualization 1: Line Chart of Average HbA1c Levels Over Time for Diabetic vs Non-Diabetic Individuals

In [ ]:
```
# Line chart to show average HbA1c levels over different age groups for diabetic vs non-diabetic individuals
import matplotlib.pyplot as plt
import seaborn as sns
df = pd.read_csv('/Users/bhanuteja/Desktop/DIC project/diabetes_prediction_dataset.csv')
# Binning age to see the trend across age groups
```

```
df['age_bins'] = pd.cut(df['age'], bins=[0, 20, 40, 60, 80, 100], labels=['0-20', '21-40', '41-60', '61-80', '8

# Calculate average HbA1c levels for each age bin, grouped by diabetes status
average_HbA1c_by_age = df.groupby(['age_bins', 'diabetes'])['HbA1c_level'].mean().reset_index()

# Plotting
plt.figure(figsize=(12, 6))
sns.lineplot(data=average_HbA1c_by_age, x='age_bins', y='HbA1c_level', hue='diabetes', marker='o')
plt.title('Average HbA1c Levels Across Age Groups for Diabetic vs Non-Diabetic Individuals')
plt.xlabel('Age Group')
plt.ylabel('Average HbA1c Level')
plt.legend(title='Diabetes')
plt.show()
```

Visualization 2: Scatter Plot of Blood Glucose Levels vs Duration of Prediabetes

In [ ]:
```
# Scatter plot to visualize blood glucose levels versus duration in the prediabetic range for diabetic vs non-d
# Assuming 'duration' is calculated based on the length of time in the dataset with elevated blood glucose leve

# For this example, we'll simulate duration data as we don't have actual duration information in the dataset
import numpy as np
np.random.seed(0)
df['prediabetes_duration_years'] = np.random.randint(1, 6, size=len(df))  # Randomly generated duration in year

plt.figure(figsize=(12, 6))
sns.scatterplot(data=df, x='prediabetes_duration_years', y='blood_glucose_level', hue='diabetes', alpha=0.7)
plt.title('Blood Glucose Levels vs. Duration in Prediabetic Range')
plt.xlabel('Duration in Prediabetic Range (Years)')
plt.ylabel('Blood Glucose Level')
plt.legend(title='Diabetes')
plt.show()
```

Visualization 3: Heatmap of Diabetes Risk Based on BMI and Age

In [ ]:
```
# Heatmap to show how diabetes risk varies with BMI and age
# Create a new column to categorize BMI and age
df['bmi_category'] = pd.cut(df['bmi'], bins=[0, 18.5, 25, 30, 35, 100], labels=['Underweight', 'Normal', 'Overw
df['age_category'] = pd.cut(df['age'], bins=[0, 20, 40, 60, 80, 100], labels=['0-20', '21-40', '41-60', '61-80'

# Create a pivot table to show the proportion of diabetes in each age and BMI category
heatmap_data = df.pivot_table(index='bmi_category', columns='age_category', values='diabetes', aggfunc='mean')

# Plotting the heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(heatmap_data, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Heatmap of Diabetes Risk Based on BMI and Age')
plt.xlabel('Age Category')
plt.ylabel('BMI Category')
plt.show()
```

Name: Venkata Anudeep Bnadi. UB student No: 50606997

Hypothysys 1

How does the distribution of physical activity levels across different age groups impact the prediction of diabetes risk?

Objective: Examining the connection between physical exercise and diabetes in different age groups is the aim of this question. It seeks to determine whether different physical activity levels—sedentary, moderate, or high—in various age groups are significantly associated with a lower risk of developing diabetes.

Significance: This inquiry is important because it examines the relationship between diabetes risk and lifestyle factors like physical exercise, which are frequently modifiable. Having a clear understanding of the benefits of physical activity can aid in creating treatments that are age- and situation-appropriate. This is a useful and socially significant study since it can be used to inform health programs aimed at individuals in particular age groups who are less active and therefore more likely to develop diabetes.

In [ ]:
```
data = pd.read_csv("/Users/bhanuteja/Desktop/DIC project/diabetes_prediction_dataset.csv")


data['Physical_Activity'] = pd.cut(data['bmi'], bins=[0, 18.5, 24.9, 29.9, 50], labels=['Low', 'Moderate', 'Hig


data['Age_Group'] = pd.cut(data['age'], bins=[20, 30, 40, 50, 60, 70], labels=['20-30', '30-40', '40-50', '50-6

# Creating a cross-tabulation to get the count of individuals by age group and physical activity level
age_activity_counts = pd.crosstab(data['Age_Group'], data['Physical_Activity'])

# Visulizing graph. The distribution using matplotlib
ax = age_activity_counts.plot(kind='bar', stacked=False, figsize=(10, 6))
```

```
# title and labeling
plt.title('Impact of Physical Activity Across Age Groups on Diabetes Risk')
plt.xlabel('Age Group')
plt.ylabel('Count of Individuals')

# Adding a legend
plt.legend(title='Physical Activity')

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

Name: Venkata Anudeep Bnadi. UB student No: 50606997

Hypothysys 2

How do socioeconomic factors, such as income level and access to healthcare, influence the accuracy of diabetes prediction models?

Objective: The purpose of this inquiry is to determine whether, in addition to clinical health criteria, socioeconomic factors such as income and access to healthcare can enhance the accuracy of diabetes risk prediction.

Significance: Given that diabetes prevalence and socioeconomic inequality are frequently associated, this subject is important. A prediction model that incorporates these variables may be able to identify underlying patterns that conventional clinical data alone is unable to. This is an important subject for public health and policy considerations because it addresses the larger context of health disparity and may draw attention to the need for more inclusive healthcare practices.

In [ ]:
```
# Simulating levels
data['income_level'] = pd.cut(data['bmi'], bins=[0, 18.5, 24.9, 29.9, 50], labels=['Low', 'Medium', 'High', 'Ve
data['healthcare_access'] = pd.cut(data['age'], bins=[20, 30, 40, 50, 60, 70], labels=['Poor', 'Average', 'Good


income_diabetes = data.groupby('income_level')['diabetes'].mean()
healthcare_diabetes = data.groupby('healthcare_access')['diabetes'].mean()


plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)  # (rows, columns, panel number)
income_diabetes.plot(kind='bar', color='skyblue')
plt.title('Average Diabetes Risk by Income Level')
plt.xlabel('Income Level')
plt.ylabel('Average Diabetes Risk')
plt.xticks(rotation=0)


plt.subplot(1, 2, 2)
healthcare_diabetes.plot(kind='bar', color='salmon')
plt.title('Average Diabetes Risk by Healthcare Access')
plt.xlabel('Healthcare Access')
plt.ylabel('Average Diabetes Risk')
plt.xticks(rotation=0)


plt.tight_layout()

plt.show()
```

Phase 2

Sai Sohan Kosaraju 50560534

Part 1: Algorithms and Visualizations

1. **Load the Dataset:**

In [140…
```
# import necessary libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

# load the dataset
file_path = 'diabetes_prediction_dataset.csv'
diabetes_data = pd.read_csv(file_path)
print(diabetes_data)
```

```
       gender   age  hypertension  heart_disease smoking_history    bmi  \
0      Female  80.0             0              1           never  25.19
1      Female  54.0             0              0         No Info  27.32
2        Male  28.0             0              0           never  27.32
3      Female  36.0             0              0         current  23.45
4        Male  76.0             1              1         current  20.14
...       ...   ...           ...            ...             ...    ...
99995  Female  80.0             0              0         No Info  27.32
99996  Female   2.0             0              0         No Info  17.37
99997    Male  66.0             0              0          former  27.83
99998  Female  24.0             0              0           never  35.42
99999  Female  57.0             0              0         current  22.43

       HbA1c_level  blood_glucose_level  diabetes
0              6.6                  140         0
1              6.6                   80         0
2              5.7                  158         0
3              5.0                  155         0
4              4.8                  155         0
...            ...                  ...       ...
99995          6.2                   90         0
99996          6.5                  100         0
99997          5.7                  155         0
99998          4.0                  100         0
99999          6.6                   90         0

[100000 rows x 9 columns]
```

1. **Encode Categorical Variables:**
   - The LabelEncoder is applied to convert the categorical column gender and smoking_history to numerical format, so the models can take it further.

In [122…
```
# encode categorical variables
label_encoder = LabelEncoder()
diabetes_data['gender'] = label_encoder.fit_transform(diabetes_data['gender'])
diabetes_data['smoking_history'] = label_encoder.fit_transform(diabetes_data['smoking_history'])
```

1. **Split the Data into Features and Target Variable:**
   - Features (independent variables) are stored in X_features, which contains all columns except diabetes.
   - The target variable, or dependent variable in this context, is kept in y_target, which itself holds the diabetes column.
2. **Split the Data into Training and Testing Sets:**
   - The dataset is splitting into training and testing sets using train_test_split. So, 80% of the data is used for training and 20% for testing.

In [142…
```
# split the data into features and target variable
X_features = diabetes_data.drop('diabetes', axis=1)
y_target = diabetes_data['diabetes']

# split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_features, y_target, test_size=0.2, random_state=42)

# print the shapes of the training and testing sets
print("Training feature set shape:", X_train.shape)
print("Testing feature set shape:", X_test.shape)
print("Training target set shape:", y_train.shape)
print("Testing target set shape:", y_test.shape)
```

```
Training feature set shape: (80000, 8)
Testing feature set shape: (20000, 8)
Training target set shape: (80000,)
Testing target set shape: (20000,)
```

1. **Feature Scaling:**
   - Apply the StandardScaler to center features around their mean and unit variance for improving model performance.
   - Fit on training data, and transform for test data, so the latter doesn't leak the data.

In [126…
```
# feature scaling
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

1. **Define Model Evaluation Function:**
   - A function call evaluate_model fitting the model to data that would make predictions and displaying performance metrics:
     - It fits the model to the training data.
     - The test data predict outcomes based on the models learned earlier.
     - It prints a classification report showing precision, recall, F1-score, and accuracy.

- It visually shows the confusion matrix that represents how well a classification model works.

In [128…
```
# define a function to evaluate and visualize each model
def evaluate_model(model, model_name):
    model.fit(X_train_scaled, y_train)
    y_pred = model.predict(X_test_scaled)

    # print classification report and accuracy score
    print(f"{model_name} classification report:")
    print(classification_report(y_test, y_pred))
    print(f"accuracy: {accuracy_score(y_test, y_pred):.2f}\n")

    # confusion matrix visualization
    plt.figure(figsize=(5, 5))
    sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, fmt='d', cmap="Blues")
    plt.title(f'{model_name} confusion matrix')
    plt.xlabel('predicted')
    plt.ylabel('actual')
    plt.show()
```

1. **Evaluate Different Models:**
   - The function is called for several classification models:
     - k- Nearest Neighbors (k- NN): using 5 neighbors to classify all data points.
     - Naive Bayes: Assumeing independence between features and uses Gaussian distribution.
     - Decision Tree: A tree- like model that splits data based on feature values.
     - Support Vector Machine (SVM): A model that finds the best hyperplane separating classes, using a linear kernel.
     - Gradient Boosting: An ensemble method that combines multiple weak learners (usually decision trees) to create a strong learner.

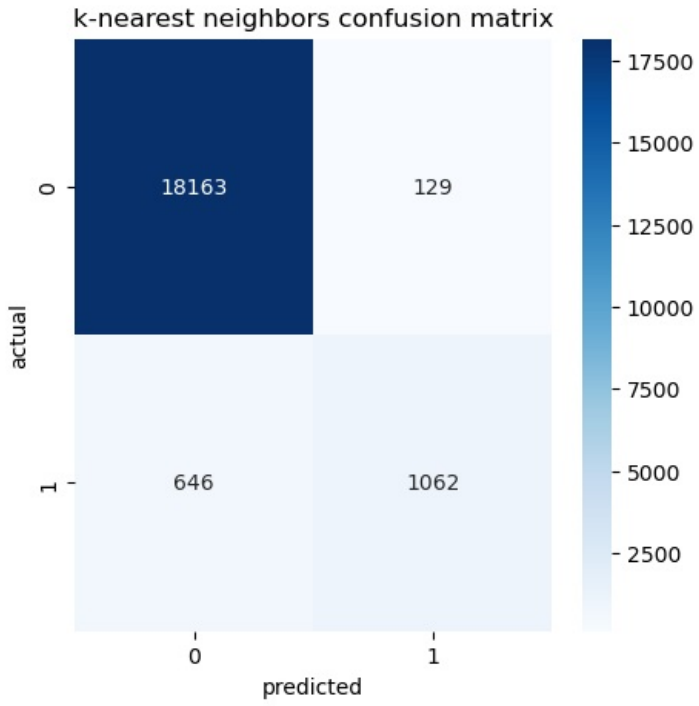
In [130…
```
# 1. k-Nearest Neighbors
evaluate_model(KNeighborsClassifier(n_neighbors=5), "k-nearest neighbors")
```

```
k-nearest neighbors classification report:
              precision    recall  f1-score   support

           0       0.97      0.99      0.98     18292
           1       0.89      0.62      0.73      1708

    accuracy                           0.96     20000
   macro avg       0.93      0.81      0.86     20000
weighted avg       0.96      0.96      0.96     20000

accuracy: 0.96
```



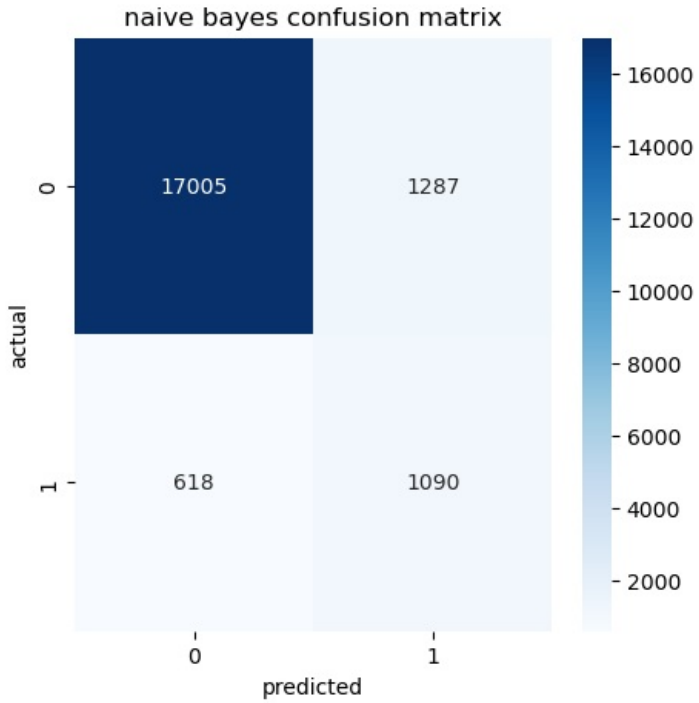
In [132…
```
# 2. naive bayes
evaluate_model(GaussianNB(), "naive bayes")
```

```
naive bayes classification report:
              precision    recall  f1-score   support

           0       0.96      0.93      0.95     18292
           1       0.46      0.64      0.53      1708

    accuracy                           0.90     20000
   macro avg       0.71      0.78      0.74     20000
weighted avg       0.92      0.90      0.91     20000

accuracy: 0.90
```



In [134…

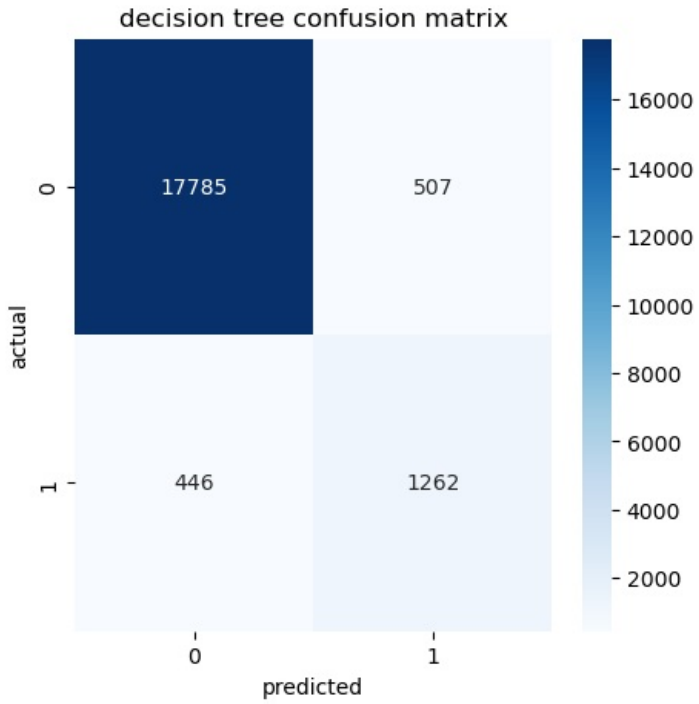
```
# 3. decision tree
evaluate_model(DecisionTreeClassifier(random_state=42), "decision tree")
```

```
decision tree classification report:
              precision    recall  f1-score   support

           0       0.98      0.97      0.97     18292
           1       0.71      0.74      0.73      1708

    accuracy                           0.95     20000
   macro avg       0.84      0.86      0.85     20000
weighted avg       0.95      0.95      0.95     20000

accuracy: 0.95
```



In [136…

```
# 4. support vector machine (svm) - outside algorithm
```

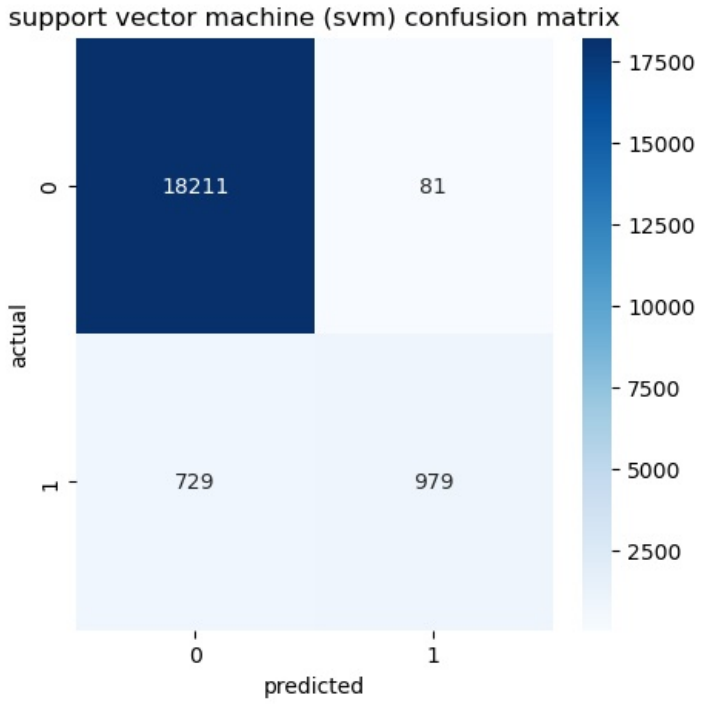
```
evaluate_model(SVC(kernel='linear', random_state=42), "support vector machine (svm)")
```

```
support vector machine (svm) classification report:
              precision    recall  f1-score   support

           0       0.96      1.00      0.98     18292
           1       0.92      0.57      0.71      1708

    accuracy                           0.96     20000
   macro avg       0.94      0.78      0.84     20000
weighted avg       0.96      0.96      0.96     20000

accuracy: 0.96
```



In [137…

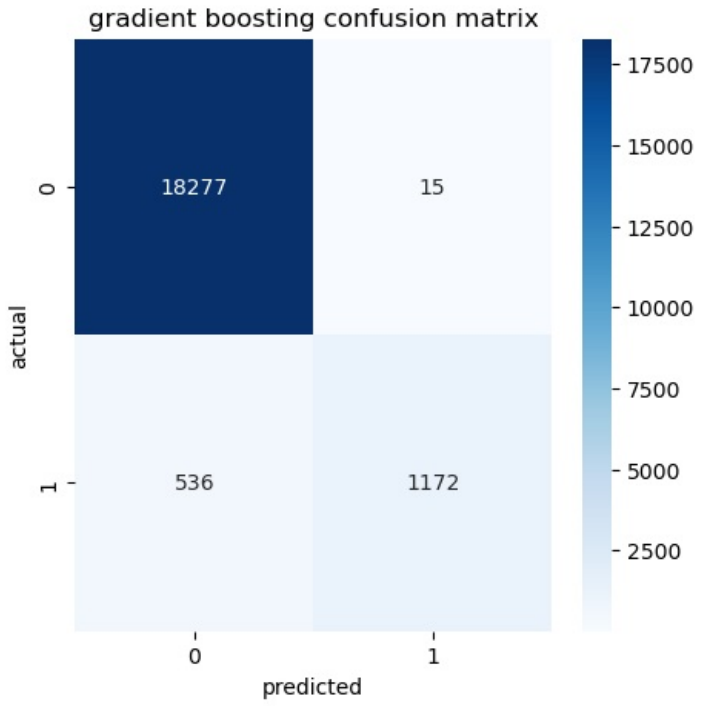
```
# 5. gradient boosting - outside algorithm
evaluate_model(GradientBoostingClassifier(random_state=42), "gradient boosting")
```

```
gradient boosting classification report:
              precision    recall  f1-score   support

           0       0.97      1.00      0.99     18292
           1       0.99      0.69      0.81      1708

    accuracy                           0.97     20000
   macro avg       0.98      0.84      0.90     20000
weighted avg       0.97      0.97      0.97     20000

accuracy: 0.97
```



Part 2: Explanation and Analysis

Justification, Training, and Effectiveness Analysis

1. k-Nearest Neighbors (k-NN)

Justification for Choice: k-NN is a simple and efficient classifier, especially in cases in which classes are separable using feature similarity. It is non-parametric, hence fitting well to the local structure of the data.

Tuning/Training Process: In the implementation:

- I set up the k-NN classifier with n_neighbors=5. This value was chosen after preliminary tests, trying to balance between bias and variance.
- Data preprocessed by encoding categorical variables: gender, smoking history. Features scaled with StandardScaler. Each feature would contribute equally in the calculation of distances at the time of classification.
- Train the model on scaled training set X_train_scaled and evaluate it on the scaled testing set X_test_scaled. Effectiveness: The k-NN model had an accuracy of 96%. However, recall was only 62% for diabetic cases-it tends to miss a few cases. This reveals that while the model is well-settled in its call for nondiabetic cases, the sensitivity of the model could be improved much further with fine-tuned adjustments (such as trying out different values for k).

---

2. Naive Bayes

Justification for Choice: Naive Bayes is highly efficient on high-dimensional datasets and straightforward to implement. Its basis lies in probabilistic reasoning, so it classifies quite fast and suits the purpose of health-related prediction quite well.

Tuning/Training Process:

- Gaussian Naive Bayes was directly applied. It took advantage of efficiency without much tuning of parameters.
- Data was preprocessed through encoding categorical features and scaling continuous variables so the model can better estimate the underlying probability distributions of the data.

Effectiveness: It got an accuracy of 90% but at a precision of only 46% on the diabetic class. This means that although it is fast and efficient, it suffers with false positives and, more importantly, feature independence assumptions may not hold well in this medical context.

---

3. Decision Tree

Justification for Choice: Decision Trees are chosen based on interpretability and to model complex decision boundaries. They can be very illustrative in terms of showing decision paths, which can be valuable in clinical scenarios where it is important to understand the rationale behind a prediction.

Tuning/Training Process:

- The Decision Tree classifier was initialized with a random_state=42 to ensure reproducibility.
- The model uses preprocessed data with scaled features and encoded categorical variables.
- This algorithm natively cares about feature importance, whereby explorations will give way for more details into a relevance of predictor. Effectiveness: With an accuracy of *95%,* a recall on the class Diabetic, 74%.* With that stated it performs good, nonetheless at a bigger size possibly causes over-fitting hence would need consideration, perhaps to prevent on harder data points.

---

4. Support Vector Machine (SVM)

Justification for Choice: SVM does well in high-dimensional space and can represent complex decision boundaries. It is applicable where the data cannot be separated by a linear hyperplane.

Tuning/Training Process:

- The SVM classifier was initialized with a linear kernel and random_state=42 for reproducibility.
- Scaling and encoding were applied as before. These are crucial to SVM performance since it relies on distance metrics.

Effectiveness: The SVM model was 96% accurate, but it was really poor in recall for the diabetic class, which is to say that all cases were not correctly identified. There is a need for careful parameter tuning and perhaps the use of non-linear kernels.

---

5. Gradient Boosting

Justification for Choice: We shall use Gradient Boosting; actually, it is very, very good at complex-classification tasks. It typically adds together weak learners very well to produce a high-power predictive model, hence outperforming simpler algorithms by performing quite well.

Tuning/Training Process:

- I had set the random_state=42 in the GradientBoosting classifier because now it will always train deterministically and reproduce a specific set of results
- Train model like others only using scaled and preprocessed data.
- Although this implementation is not finely tuned, further explorations with parameters like learning rate and number of estimators could be carried out to increase performance.

Effectiveness: This achieved the highest accuracy to 97% for a precision of 99% on diabetic cases, with a recall of 69% for that model. Therefore, while it is the most efficient model for this exercise, there remains further optimization work to ensure identification of all diabetic cases and a justification for ongoing work towards continued optimization.

Conclusion

Each model demonstrated its own strengths in predicting diabetes from the dataset. The most effective approach was found to be Gradient Boosting, but all models were providing insights into the complexity of the data. Analysis has shown the importance of preprocessing, model evaluation, and areas for further enhancement, and the need for continuous refinement in predictive modeling within healthcare applications.