EARLY PREDICTION OF DIABETES RISK

MACHINE LEARNING PROJECT

SAMATRIX CONSULTING PVT LTD
GURUGRAM
contact@samatrix.io

Table of Contents

Early Prediction of Diabetes Risk

Problem Statement:

Diabetes is a major global health concern that affects millions of people every year. Early detection can help individuals take preventive steps and manage their health better. In this project, we use real-world health data from patients to build a machine learning model that can predict whether a person is likely to have diabetes or not. Students will explore the data, clean and prepare it, apply SMOTE to handle imbalance, and build a K-Nearest Neighbors (KNN) classifier to make predictions. The goal is to not only improve technical skills but also understand how data science can support important health decisions.

Learning Objectives

- Explore and clean real-world health datasets to prepare for analysis
- Identify and treat data quality issues such as missing or irrelevant values
- Apply feature scaling techniques to prepare data for machine learning
- Implement oversampling using SMOTE to address class imbalance
- Build and train a K-Nearest Neighbors (KNN) classifier
- Evaluate model performance using accuracy, precision, recall, and F1 score
- Visualize insights using plots like histograms, pair plots, and heatmaps
- Interpret results to derive meaningful conclusions about diabetes risk factors

Dataset

https://samatrix-data.s3.ap-south-1.amazonaws.com/ML/diabetes-data.csv

What to Submit:

Submission Type: Individual

Each student must submit the following:

1. **Jupyter Notebook (.ipynb file) or python filr (.py file)**
   a. Filename: YourFullName_diabetes.ipynb (e.g., AnanyaKumar_ diabetes.ipynb)
   b. Your notebook must follow the steps and structure discussed in class following the instructions in the submission guideline
2. **Word or PDF file**
   a. Answers “Questions for Report” in separate file

Step 1 : Importing the Libraries

To prepare your Python environment by loading the essential tools (called libraries) that help you work with data, perform calculations, and create beautiful graphs.

1. **Learn what each library does:**
   - **NumPy** is like a calculator for large sets of numbers.

- **Pandas** is like Excel for Python — it helps you manage and explore tables of data.
- **Matplotlib** helps you draw simple charts like bar graphs and line plots.
- **Seaborn** makes those charts even more beautiful and easy to understand.

Tip:
If you ever get an error like ModuleNotFoundError, it means the tool isn't installed in your environment. In Colab, it should already be installed. If not, ask your expert or use !pip install (but expert will guide you for that).

Step 2: Load Your CSV

To load a real-world dataset (about diabetes patients) directly from a web link and make it ready for analysis in your project.

1. **What this step is doing:**
   - It connects to the provided website link.
   - It downloads the dataset file.
   - It saves that dataset into a table (called a DataFrame) in your Python notebook.
2. **What kind of data is in the file:**
   - Each row in the file is about a person.
   - Each column is a health-related detail, like blood pressure, insulin level, age, etc.
   - This data is about diabetes and will help you explore how different factors relate to health.

Tip:

- If the dataset does not load properly, check if your internet is working and the link is correct.
- You only need to run this step once. After that, the data is ready to use in your notebook.

Questions for Report

Q1. What is the purpose of loading the dataset at the beginning of your project?
In your own words, explain why we load the dataset first. What does it allow you to do in the rest of the project?

Q2. What kind of information do you think is stored in this dataset?
List at least three types of information (columns) you expect to find in the diabetes dataset based on its name. What kind of questions could this data help you answer?

Q3. If the dataset did not load properly, what possible problems could you check?
List 2–3 things you would troubleshoot if the data didn't load — for example, internet connection, broken link, etc.

Step 3: First Five Rows of Data

To view the top few rows (usually 5) of the dataset so you can get a basic idea of what information is included, what each column represents, and whether the data looks clean and complete.

1. **Give Command: "Show me the top of the table."**
   This step will display the first 5 entries (people) from your dataset. You'll see what columns are included and what kind of numbers or values are present.
2. **Look closely at the column names and think:**
   - What does each column represent?
     For example, "Glucose" may refer to blood sugar level, and "BMI" to body weight status.
   - Are any values missing or suspicious (like lots of zeros)?
3. **Think about the people in the rows:**
   - Each row is one person (or one patient record).
   - Ask: Does the person have diabetes or not? What health indicators do they show?

Why this step:

- Get familiar with your dataset right away.
- Identify which columns are important and which might need cleaning.
- Begin thinking like a data analyst — someone who looks for patterns, problems, and potential insights.

Questions for Report

Q1. What are the names of the columns (features) you saw in the dataset?
List all the column names you saw. What do you think each one means?

Q2. What kind of values do you notice in the first few rows? Are they numbers, text, or something else?
Describe what kind of data you see (e.g., whole numbers, decimals, yes/no answers).

Q3. Which 2 or 3 columns do you think might be most useful for predicting whether a person has diabetes?
Pick a few columns you believe are important and explain why you think they matter.

Q4. Do you notice anything strange, surprising, or possibly wrong in the first few rows of the data?
Are there any missing values, zeros that don't make sense, or something unexpected?

Q5. Based on the first few rows, what kind of questions would you like to answer using this data?
For example: Can glucose level alone predict diabetes? Do older people have more risk?

Step 4: Check Dataset Structure

Samatrix.io

To get a quick and complete summary of your dataset so you can understand how many people (rows) are in it, what kind of data each column contains, and whether any information is missing.

1. **Print a quick summary of your dataset.**
   This step shows the total number of records (rows), the number of columns, and the type of data in each column — like whole numbers or decimal numbers.
2. **Read how many rows are in your data.**
   Each row is one person or case. For example, 768 rows mean there are 768 people whose health details are stored.
3. **Look at the column names.**
   These are the types of information collected about each person, like:
   - Number of pregnancies
   - Blood sugar level (glucose)
   - Body Mass Index (BMI)
   - Whether they have diabetes or not
4. **Check for missing values.**
   Next to each column, you'll see how many values are *not missing*. If all values are present (e.g., 768 out of 768), that means your data is complete for that column.
5. **Understand the type of data in each column.**
   - If it says “int” (short for integer), it means whole numbers (like 25 or 80).
   - If it says “float,” it means decimal numbers (like 23.5 or 31.2).
6. **Look at the memory size.**
   This tells you how much space the dataset is using in your computer. A small file (like 54 KB) means it’s quick and easy to process.

Questions for Report

Q1. How many records (rows) are in the dataset?
Write the total number of rows. What does each row represent in this dataset?

Q2. How many columns are there, and what does each one seem to represent?
List all the columns. For each one, write what kind of information it holds (e.g., Glucose = blood sugar level).

Q3. Did any column have missing data? How do you know?
Check the “Non-Null Count” for each column. Were all values present or were some missing?

Q4. What are the different types of data in the columns? Why does that matter?
Mention which columns are whole numbers (int) and which are decimals (float). Why is it useful to know this?

Q5. Based on the structure, do you think this dataset is ready to use for analysis? Why or why not?
Is the data clean and complete? Do you see anything that might need fixing before analysis?

Step 5: Summary Statistics

To quickly understand the data in each column — including how values are distributed, what the average is, and whether anything looks unusual or needs cleaning.

1. **Tell your tool to summarize the dataset.**
   You'll be asking for a quick overview of every column with numbers — such as age, glucose level, or BMI.
2. **Look at the number of entries (count).**
   This tells you how many people (rows) have values in each column. Ideally, this should match the total number of rows in the dataset.
3. **Check the average (mean) for each column.**
   This gives you a general idea of what is "normal" in your data. For example, the average glucose or BMI value in the dataset.
4. **Understand the minimum and maximum values.**
   These tell you the lowest and highest values in each column. Use this to check for unusual numbers like 0 in glucose or BMI (which may not be realistic).
5. **Pay attention to the 25%, 50%, and 75% values.**
   These are called percentiles and show you how the values are spread out. The 50% value is the median — the middle point of your data.
6. **Watch out for data problems.**
   If any values seem too low (like 0 for insulin) or too high (like 846 for insulin), write these down. You might need to fix or replace them later.
7. **Transpose the table.**
   To make it easier to read, you can flip the table so each column (like Glucose or Age) becomes a row. This is just to improve how it looks and reads.

Questions for Report

Q1. Which column has the highest average (mean) value? What does this tell you about that feature?
Look at the "mean" row and write which column has the biggest number. Why do you think that is?

Q2. Do you notice any columns that have 0 as the minimum value? Is that realistic?
Some values like Glucose, BloodPressure, or BMI should never be zero. Write down any unrealistic values you noticed and explain why they might be a problem.

Q3. Which columns show the most variation (look at the "std" or standard deviation)? Why does variation matter?
High variation means the values are spread out. Write one or two columns with large standard deviation and what that might tell us.

Q4. Based on the 25%, 50%, and 75% values (percentiles), which columns seem to have balanced distributions?
If the 25%, 50%, and 75% values are not too far apart, the data is well balanced. Pick a column that looks balanced and explain your reason.

Q5. Which features do you think will be most important in predicting the outcome (diabetes or not)? Why?
Based on your understanding of the features and their statistics, write 1–2 features you think will help most in prediction, and explain in simple terms.

Step 6: Create a Safe Copy and Prepare Important Columns

To make a safe copy of your dataset so you don't accidentally damage the original, and to focus on a few important health-related columns for cleaning and analysis.

1. **Make a Safe Copy of Your Dataset**
   - Think of it like making a "Save As" copy of a file.
   - This new copy will be used for all changes and testing.
   - The original file stays untouched in case you make a mistake or want to start over.
2. **Choose Key Health Columns You Want to Focus On**
   - In this project, we will work with:
     - **Glucose** – blood sugar level
     - **BloodPressure** – how high or low your blood pressure is
     - **SkinThickness** – thickness of skin fold (used to estimate body fat)
     - **Insulin** – hormone that controls blood sugar
     - **BMI** – Body Mass Index (weight vs. height)

Questions for Report

Q1. Why is it a good idea to make a separate copy of your dataset before making changes?
Explain what could go wrong if you worked directly on the original data. Why might a copy be safer?

Q2. What do you think could happen if we don't focus on specific columns for cleaning and just use all the columns together?
Think about what kinds of problems might show up — such as wrong values or unrelated data.

Step 7 : Checking for Missing Values

To identify which parts of your data are **missing or empty** so that you can **clean the data properly** before building any machine learning model.

1. **Use the Copy of the Data**
   You should be working on the **copy of your dataset** that you made earlier. This ensures you don't accidentally change the original data.

2. **Ask the Question: What Data is Missing?**

Think of your dataset like a big table or Excel sheet.
Now ask yourself:
Are there any **blank spaces** where information is missing?
For example:

- Did anyone skip their blood pressure reading?
- Is there a row where insulin level is not recorded?
- Did anyone forget to report their BMI?

3. **Use a Simple Command to Count What's Missing**
You'll now check **every column** (like Glucose, Insulin, BMI, etc.) to see **how many cells are empty** in each one.
This tells you:
    - Which columns have missing values
    - How many missing entries are there

Why This Step

- Missing data can make your analysis incorrect
- This is the first step in **data cleaning**
- Machine learning models cannot work well if important data is missing
- Checking now saves you time and effort later

Questions for Report

Q1. Why is it important to check for missing values in your dataset before doing analysis or prediction?
Explain how missing information might affect the results or decisions made by a machine learning model.

Q2. Which columns in your dataset had missing values? How many were missing in each?
Write down the actual output you saw and name the columns that were affected.

Q3. Do you think it's okay for health-related data like "Glucose" or "BloodPressure" to have a value of 0 or be empty? Why or why not?
Share your thoughts on whether such values are realistic in a medical dataset.

Q4. If you ignored missing values completely and trained a machine learning model, what problems might happen?
Try to imagine how the final prediction could be affected if the model sees incorrect or incomplete data.

Step 8: Visualizing Data Distributions with Histograms

To help you **understand the spread and distribution** of values for each feature (column) in the diabetes dataset by using **histogram charts**. These charts give you a visual idea of what the data looks like, what's common, and what might be unusual.

1: Create Histograms

- Use a function to create **one histogram for each numeric column** in the dataset.
- Each histogram will show how frequently different values appear.

2: Adjust the Plot Size

- Make your entire chart area larger (like increasing paper size), so you can see all histograms clearly in a grid layout.

3: Show the Plots

- Display the histograms on the screen.
- You should see 9 charts — one for each of the following:
    - Pregnancies
    - Glucose
    - Blood Pressure
    - Skin Thickness
    - Insulin
    - BMI
    - Diabetes Pedigree Function
    - Age
    - Outcome (whether a person has diabetes or not)

What to Look For in the Output:

- Are there any values like zero in Glucose, Blood Pressure, Insulin, etc.? (These might be missing or incorrect.)
- Are most people in a certain age group or BMI range?
- Is the dataset balanced in the Outcome column? (Do we have similar numbers of people with and without diabetes?)

Tips:

- Think of each bar in the histogram as showing how many people fall into that category (e.g., how many people had BMI between 30–35).
- If a column shows a big spike at 0, it may need to be cleaned or fixed later.
- This is a powerful way to find problems in data before building any machine learning model.

Questions for Report

Q1. What can you say about the distribution of the 'Glucose' and 'BloodPressure' values?
Are they mostly centered around a certain value? Are there any unusually low values like zero?

Q2. Which columns seem to have many zero values (e.g., Insulin, SkinThickness)?
Do you think zero is a valid value for these health measurements? What might zero represent in this dataset?

Q3. Look at the 'Age' and 'BMI' histograms. What age group or BMI range do most patients fall under?
Are there more young or older individuals? Does BMI look normally distributed or skewed?

Q4. Check the 'Outcome' column. Are there more diabetic or non-diabetic patients in the dataset?
Why might this imbalance matter when building a prediction model?

Q5. What data quality issues do you think should be addressed before using this data in a machine learning model?
Mention any possible outliers, missing values, or unrealistic numbers you noticed in the plots.

Step 9: Fix Zero Values & Understand Data

Sometimes, our dataset has **zero values** in columns where 0 doesn't make real-world sense — like **0 BMI, 0 blood pressure, or 0 insulin**. These are likely errors or missing data. In this step, we'll replace those **unrealistic zero values** with more reasonable values, and then **draw graphs (histograms)** to see how the data looks.

1. Understand which columns can't have zero
Look at these columns:

- Glucose
- Blood Pressure
- Skin Thickness
- Insulin
- BMI

Think about it — a living person cannot have:

- **0 glucose (blood sugar)**
- **0 blood pressure (that means no blood flow!)**
- **0 BMI (body mass index)**

So we decide these 0s are not real — they must be **errors** or **missing values**.

2. Replace those zero values with median

- The **median** is the middle value in a column.
- It's better than the average because it's not affected by extreme numbers.
- For example, instead of guessing a number for missing insulin values, we just **replace all 0s with the median insulin value**.

Do this for all five columns above.

3. Draw Histograms

- A histogram is a type of graph that shows **how many times each value appears**.
- It helps us **see the shape** of the data.
  For example:
    - Is it balanced like a hill?
    - Does it have too many low or high values?
    - Are there any weird spikes or gaps?

Draw one histogram for **each column**.

Questions for Report

1. Why is it important to replace zero values in health-related columns like Blood Pressure, Glucose, or BMI?
Explain with real-life meaning. For example, what does a "0 BMI" mean, and why can't it be correct?

2. What method did you use to replace the zero values, and why is it a good choice?
Hint: Did you use the median or something else? Why not the average (mean)?

3. After cleaning the data, what changes did you observe in the histograms of any two columns?
Pick any two graphs and explain: Did the shape change? Do the values look more realistic now?

4. What would happen if you didn't fix the zero values? How would it affect future analysis or predictions?
Think about how wrong values could confuse a machine learning model or give incorrect results.

5. Which column do you think has the most unusual values, and why?
Look at the graphs — are there columns that are heavily skewed or look very different from the rest? What might be the reason?

Step 10: Check the Size of Dataset

To find out how many rows (people) and columns (health information types) are present in your dataset.

1. **Think about what do we want to know?**
   - We want to check **how big** this table is.
   - Specifically, we want to know:
     - How many **rows** are in it (each row is one person or patient).
     - How many **columns** are in it (each column is a type of health data like age, glucose, etc.).
2. **How to do it**
   - Run a simple command:
     "Hey, how many rows and columns does the data table have?"
3. **What you will get**
   - You'll see something like:
     (768, 9)
   - This means:
     - **768 rows**: Your dataset has 768 people (or data entries).
     - **9 columns**: There are 9 types of health data for each person.
4. **Before you move ahead**
   - Make a note of the shape (number of rows and columns) in your project notebook or report.

- If the shape looks wrong (for example, 0 rows or fewer columns than expected), go back and check your data loading step.

Questions for Report

Q1. **How many rows and columns are present in your dataset? What does each row represent? What does each column represent?**
(Use the output of the shape command to answer. Relate it to real-world examples like rows = people, columns = health data.)

Q2. **Why is it important to check the shape of a dataset before doing any analysis or machine learning?**
(Think about whether you have enough data and whether any data might be missing.)

Q3. **If the shape showed fewer columns than expected, what possible issues could have caused this?**
Try to list any one or two reasons why data might be missing or loaded incorrectly.

Q4. **Imagine you are working with a dataset that has 5,000 rows and only 2 columns. Do you think this is good enough for training a model? Why or why not?**
(This helps you think about data size and feature richness.)

Q5. **If your dataset had 0 rows after loading, what would be your first steps to fix the issue?**
(Try to think what you would check: the file name, the URL, or the data file itself.)

Step 11: Check How Many People Have Diabetes

To find out how many people in the dataset have diabetes and how many do not.

1. **Understand what we're checking**
   You are looking at a column called **"Outcome"** in your dataset.
   This column tells us if the person has diabetes (value = 1) or does not have diabetes (value = 0).
2. **Why we need to check this**
   - Before building a machine learning model, it's important to know if your data is balanced.
   - If one group (like non-diabetic people) is much larger than the other group, it could make the model unfair or inaccurate.
   - This step gives you a clear count of how many people fall into each category.
3. **What to do**
   - Use a command that checks how many times each value appears in the "Outcome" column.
   - You will get a small summary showing two numbers:
     - One for people who **do not** have diabetes (Outcome = 0)
     - One for people who **do** have diabetes (Outcome = 1)

4. **How to interpret the result**
    - Look at the numbers and compare them.
    - If both values are somewhat close, your data is balanced.
    - If one number is much larger than the other, your data is imbalanced and might need extra handling later.
5. **What to do next**
    - Write down or save the result.
    - Think: Is the number of diabetic and non-diabetic people balanced? Why might this matter for prediction?
    - Use this information in your project report to explain how well your data represents both classes.

Questions for Report

Q1. **How many people in the dataset have diabetes and how many do not?**
(Use the result of the count from the "Outcome" column.)

Q2. **Which group is larger in the dataset – people with diabetes or without diabetes?**
(Write a short explanation comparing the two groups.)

Q3. **Why do you think it's important to check the number of diabetic and non-diabetic cases before building a model?**

Q4. **If one group is much bigger than the other, what kind of problem could that cause in prediction?**

Q5. **How would you describe this dataset to someone who wants to build a diabetes prediction app?**
(Give a short summary of the data's health outcome based on the counts.)

Step 12: Plot Scatter Matrix

Create a large visual chart (called a *scatter matrix*) to explore and understand how different numerical columns in your dataset relate to each other.

1: Understand What a Scatter Matrix Is
- A scatter matrix is a grid of small charts.
- It compares every column in your dataset with every other column using tiny scatter plots.
- It helps you see which variables might be related.

2: Generate the Scatter Matrix
- Use your data table to create the scatter matrix plot.
- Make the chart large so it's easy to read — for example, a square of size 25x25.

3: Understand the Output
- Each square shows the relationship between two features (like Glucose vs. BMI).

- The diagonal line of squares shows histograms — they show how values are spread for one column.
- The other squares show how two columns change together.

4: **How to Read the Plot**
- If you see dots forming a straight line, it may mean the two variables are strongly related.
- If the dots are spread out randomly, those variables may not be related.
- Look for patterns, trends, or unusual dots (called outliers).

5: Use the Chart for Decision Making
- This chart can help you choose which features to use when building a prediction model.
- You can also decide if any columns are too similar or not useful.

Questions for Report

Q1. **Which two features (columns) in the scatter matrix seem to have the strongest positive relationship?**
Explain what kind of pattern you saw in the scatter plot and why you think the features are related.

Q2. **Do you notice any features that do not seem related to others?**
Pick one or two and describe what their scatter plots look like when compared to other features.

Q3. **Were there any unusual points (outliers) visible in the scatter matrix?**
If yes, mention which feature(s) had outliers and what could be the reason for those values.

Q4. **What do the histograms on the diagonal line of the scatter matrix tell you about the data?**
Pick any one feature (e.g., Age or BMI) and describe whether the values are mostly low, high, or spread out.

Q5. **Based on the scatter matrix, which features do you think might be most useful for predicting diabetes?**
List 1 or 2 features and explain your reasoning using the visual patterns from the scatter matrix.

Step 13: Plot Scatter Matrix (With `hue=outcome`)

To understand how different health factors are related to each other, and how these relationships differ between people **with** and **without diabetes**.

1. **Understand the goal:**
   - We want to visually explore the dataset to identify which health indicators (like glucose, BMI, age, etc.) may help **predict diabetes**.

- A pairplot is like a **visual summary of the entire dataset**, comparing every pair of features side by side.

2. **What we will do:**
   - Create a **pairplot** where each point is a person.
   - The **color** of each point will show whether that person has diabetes (Outcome = 1) or not (Outcome = 0).
   - The **diagonal plots** will show how each feature is distributed (e.g., how BMI values are spread).
3. **What will happen after running this step:**
   - You'll get a big grid of scatterplots and curves.
   - Each small square shows how two features (like Glucose and Age) relate.
   - Colors (orange and blue) show the two groups: with and without diabetes.
4. **What to look for in the plot:**
   - Which features have **different patterns** between diabetic and non-diabetic people?
   - For example: Do people with higher Glucose tend to be diabetic?
   - Are there any **strong visible patterns** that stand out?
   - This can help you pick **important features** for your prediction model later.

Tip:
This step is **not about building a model**, but about **seeing the data with your eyes**. It helps you think like a detective — looking for clues in the patterns.

Questions for Report

Q1. **Which features show a clear difference between people with and without diabetes?**
Look at the scatterplots and histograms. Which features (like Glucose, BMI, Age) look clearly different between the two outcome groups?

Q2. **Is there any feature that does not help much in separating diabetic and non-diabetic individuals?**
From the plots, are there any variables where both diabetic and non-diabetic people have similar patterns?

Q3. **What extra insight do you gain by using hue='Outcome' in the pairplot that you do not get in the plain pairplot?**
Explain with an example (e.g., Glucose or BMI) how the color-coded plot helps you see differences between people with and without diabetes more clearly.

Q4. **If you had to choose only one pairplot to include in your final project report, which one would you choose and why?**
Discuss your reasoning. Consider things like how easy it is to understand, how much information it shows, and how helpful it is for choosing important features.

Q5**. How can the pairplot with hue='Outcome' help you decide which features might be important for predicting diabetes?**

Give an example of a feature where the two classes (diabetes and non-diabetes) appear clearly separated, and explain why that might be useful for building a prediction model.

Step 14: Plot a Heatmap

To understand how different health-related features are related to each other and to the diabetes outcome using a color-coded visual map (heatmap).

1. **What is a heatmap?**
   - A heatmap is a colorful chart that shows how strongly different features are connected or related.
   - Each small square in the chart compares two columns and tells us if they move together or not.
2. **Plotting tool:**
   - You will now open a tool that helps draw pictures from data (like graphs and maps).
   - Set the figure size large enough so that all labels and boxes are visible clearly (for example, 12 units wide and 10 units tall).
3. **Ask the tool to compare all columns:**
   - You're asking it to calculate how strongly each feature (like Glucose or BMI) is connected to every other feature.
   - It compares these pairs and gives you a number between -1 and 1.
     - Close to 1 = they rise and fall together.
     - Close to -1 = one goes up while the other goes down.
     - Close to 0 = no relationship.
4. **Display the heatmap with color:**
   - The squares are filled with colors to make it easier to read:
     - Green means strong connection.
     - Red means weak or negative connection.
     - Yellow/orange is medium.
5. **Add labels and values:**
   - Show the exact numbers in each square so you can see the actual relationship (for example, 0.47 means a strong positive relationship).
   - Label each row and column with the names of the features (e.g., Age, BMI, Outcome).
6. **View and analyze:**
   - Look at the final chart. Try to notice which features are strongly connected to the Outcome.
   - Ask yourself: Which features seem important? Which ones may not be useful?

Questions for Report

Q1. **Which two features in the dataset show the strongest positive relationship?**
Hint: Look for the pair with the highest positive value (close to 1) in the heatmap.

Q2. **Which feature shows the strongest relationship with the Outcome (diabetes)?**
Why do you think this feature might be important for predicting diabetes?

Q3. **Were there any pairs of features that showed little or no relationship (value close to 0)?**
What does it mean if two features have no strong connection?

Q4. **What insights did the heatmap give you that you could not easily see in the pair plot?**
Compare the heatmap with the earlier pairplot. Which one helped you understand the data better, and why?

Q5. **How can this heatmap help in choosing the right features for building a prediction model?**
Based on your heatmap, which 2 or 3 features would you choose as most useful in predicting whether someone has diabetes?

Step 15: Scaling the Data Before Modeling

Prepare the health dataset for machine learning by scaling the data, so that all health measurements are on a similar scale.

1. **Understand the Need for Scaling**
    - Different health indicators like glucose levels and BMI are measured in different units (e.g., mg/dL vs. kg/m²).
    - Some values might be very large (e.g., insulin levels), while others are small (e.g., pedigree score).
    - Models like logistic regression or k-nearest neighbors work better when all numbers are on a similar scale.
2. **Remove the Outcome Column Temporarily**
    - "Outcome" is the result we want to predict — whether a person has diabetes or not.
    - Before scaling, we temporarily set this column aside because we only want to scale the health inputs (not the answer).
3. **Use a Standard Scaling Technique**
    - Standard scaling is like adjusting each column so that:
        - The average becomes zero
        - The spread of values fits into a standard range
    - This helps the model treat all features equally, no matter their original units.
4. **Transform the Data**
    - Apply the scaling process to all columns like Glucose, BMI, BloodPressure, etc.
    - After scaling, each column will have values that are easier to compare.
5. **Create a New Table with the Scaled Data**
    - Once all the health values are scaled, we put them back into a clean and organized table.
    - This new table looks just like the original one but now contains scaled values.
    - Each column still has a name (like 'Age', 'BMI', etc.), which makes analysis and reporting easier.

Example:
Imagine you're comparing heights in centimeters and weights in kilograms. If you plot them together without adjusting, the numbers will look very different and hard to compare. Scaling is like adjusting both to a common scale, so a machine can "see" and "understand" the relationships between them more clearly.

Questions for Report

Q1. **Why did we need to scale the data before building a machine learning model?**
Hint: Think about how health features like glucose and insulin have very different value ranges.

Q2. **Which columns were selected for scaling, and which column was left out? Why?**
Hint: Consider what "Outcome" represents in the dataset.

Q3. **What do you think would happen if we skipped the scaling step? How might it affect the model's performance?**
Hint: Think about how the model would treat very large numbers vs. small ones.

Q4. **After scaling, do the actual values of the features (like BMI or Glucose) still have the same meaning? Explain.**
Hint: Consider how scaling changes the values, but not the information they carry.

Q5. **How is the new table after scaling different from the original one? What's one benefit of creating a separate table instead of replacing the original?**
Hint: Think about data safety and keeping the original dataset untouched.

Step 16: Test Train Data Split

We want to divide our dataset into two parts:

- One part to **train the machine** (learn from existing data)
- Another part to **test the machine** (check how well it has learned)

1. **Identify what you are trying to predict**
   Look at the dataset and decide which column contains the final outcome or result.
   In this case, the **Outcome** column tells us whether a person has diabetes (1) or not (0).
   This is the column we want the machine to predict.
2. **Separate your data into two parts**
   - One part (called X) contains all the features (like Glucose, Age, BMI, etc.)
   - The other part (called y) contains just the Outcome column.
3. **Use a tool to split the data**
   Use a special function that helps you divide the data into four pieces:
   - Features for training (X_train)
   - Features for testing (X_test)
   - Target labels for training (y_train)
   - Target labels for testing (y_test)

4. **Decide how much data to use for testing**
   Usually, we use 20% or 30% of the data to test the model.
   In this project, you will use **one-third** (about 33%) of the data to test.
5. **Make sure the diabetes cases are balanced in both sets**
   We don't want all diabetes cases to be in just one part.
   Use a setting that keeps the percentage of diabetic and non-diabetic cases similar in both training and testing data.
6. **Set a random seed to make your result repeatable**
   This ensures that if you run the code again, you get the same split.
   It helps your teacher or teammates reproduce your work.

Step 17: Balancing the Dataset using SMOTE

To balance the training data so that both diabetic and non-diabetic cases are equally represented. This helps the machine learning model learn from both classes fairly.

1. **Understand the Problem:**
   - In the diabetes dataset, you might notice that there are more non-diabetic cases (Outcome = 0) than diabetic cases (Outcome = 1).
   - If you train your model on this unbalanced data, it may learn to always predict "not diabetic", which is not helpful.
2. **Learn About SMOTE:**
   - SMOTE stands for *Synthetic Minority Over-sampling Technique*.
   - It creates **artificial but realistic data points** for the smaller class (diabetic cases) by studying the existing data patterns.
   - The goal is to make the number of diabetic and non-diabetic cases **equal** in the training data.
3. **Apply SMOTE to Training Data Only:**
   - Use SMOTE only on your **training data**.
   - Do not apply SMOTE to your test data.
   - This ensures your test results remain realistic and unbiased.
4. **Store the New Balanced Data:**
   - After applying SMOTE, you will have a **new version** of your training data.
   - This new data has equal numbers of diabetic and non-diabetic samples.
5. **Check Before and After:**
   - Print or view the number of diabetic and non-diabetic cases before SMOTE.
   - Then check again after SMOTE to confirm they are now equal.
6. **Why This Step is Important:**
   - Balancing helps the model learn better and prevents it from becoming biased.
   - A balanced model performs better in real-life situations where detecting the minority class (like diabetes or fraud) is very important..

Questions for Report

Q1. **Why do you think it is important to balance the number of diabetic and non-diabetic cases in the dataset before training a model?**

Q2. **What problems could arise if we did not apply SMOTE and trained the model on unbalanced data? Explain with a simple example.**

Q3. **After applying SMOTE, what changes did you observe in the number of samples for each class? Write the numbers before and after.**

Q4. **Do you think synthetic (artificial) data created by SMOTE is as useful as real data? Why or why not?**

Q5. **How does balancing the data help improve the fairness or accuracy of your final machine learning model?**

Step 18: Finding the Best K for KNN Classifier

In this activity, you will explore how changing the number of neighbors (called *k*) affects the performance of a K-Nearest Neighbors (KNN) classification model.

Follow these steps carefully and make sure you understand the purpose of each step.

1. **Create Two Empty Lists**
    - One list will store how well the model performs on the training data (what it learned from).
    - Another list will store how well the model performs on the testing data (new data it hasn't seen before).
2. **Try Different Numbers of Neighbors (K values)**
    - Use a loop that repeats from k = 1 to 14.
    - In each round of the loop, build a new KNN model with that specific value of k (e.g., 1 neighbor, 2 neighbors, … up to 14 neighbors).
3. **Train the Model on Training Data**
    - Teach the model using the balanced training data (after applying SMOTE).
    - The model will learn what patterns are common in people with and without diabetes.
4. **Measure Accuracy on Training Data**
    - After training, check how many predictions the model got right using the same data it learned from.
    - Save this result in your training score list.
5. **Measure Accuracy on Test Data**
    - Then test the same model on new data that the model has never seen.
    - Save this result in your testing score list.
6. **Compare All Results**
    - After the loop finishes, you will have two sets of results: one for training accuracy and one for testing accuracy.
    - You can now compare which k value gives you the best performance without overfitting.

Tips

- A very low value of k (like 1) may be too specific and perform poorly on new data.
- A very high value of k may be too general and miss useful patterns.
- The best k is often somewhere in the middle, where both training and testing accuracy are reasonably high.

Questions for Report

Q1. **Why do we test the KNN model with different values of k (number of neighbors)? What does changing k help us understand?**

Q2. **Based on your results, which value of k gave the best performance on the test data? Was this value also the best for training data? Why might they be different?**

Q3. **Did you notice any k values where the model performed very well on training data but poorly on testing data? What could this tell us about the model?**

Q4. **Why is it important to compare both training accuracy and testing accuracy when building a machine learning model? What could go wrong if we only looked at training accuracy?**

Q5. **Imagine you are recommending a KNN model for predicting diabetes in a real hospital setting. Which value of k would you choose and why? Justify your answer with your testing results.**

Step 19: Find the Best Training and Testing Score in KNN

In this step, you will carefully examine how your K-Nearest Neighbors (KNN) model performs using different values of "k" (the number of neighbors). You'll look at both:

- How well the model performs on the training data
- How well the model performs on the testing data

Then, you'll find out:

- Which value of k gave the **best training score**
- Which value of k gave the **best testing score**

Why This Is Important:

- The **training score** shows how well your model learned from the known data.
- The **testing score** shows how well your model performs on new, unseen data.

The goal is to find a **balance** — a k-value where both training and testing scores are reasonably high.

1. **Review the Accuracy Scores:**
   After training your KNN model with different values of k (from 1 to 14), you now have two lists:
   - One list showing the model's accuracy on training data
   - One list showing the accuracy on testing data
2. **Find the Best Training Score:**
   - Go through the list of training scores and find the highest value.

  - Find the value(s) of k that gave this highest training score.
  - This tells you which k made your model learn best from the training data.
3. **Find the Best Testing Score:**
  - Now check the testing scores list.
  - Find the highest score there and the value(s) of k that achieved it.
  - This shows you which k worked best on unseen data — a good sign of a strong model.
4. **Compare Training vs. Testing Results:**
  - If the training score is very high but the testing score is much lower, your model may be **overfitting** (too focused on the training data).
  - If both scores are similar and reasonably high, your model is likely **balanced and reliable**.
5. **Make a Final Recommendation:**
  - Choose the best value of k not just based on the highest testing score, but also by checking if the training score is close.
  - Your final k-value should give good results for both training and testing.

Questions for Report

Q1. Which value(s) of k gave the highest training accuracy?
What was the training accuracy percentage for that value(s)? Explain why this is important to observe.

Q2. Which value(s) of k gave the highest testing accuracy?
What was the testing accuracy percentage for that value(s)? Why is testing accuracy important?

Q3. Did the best training score and best testing score happen at the same value of k?
If not, what does the difference tell you about how the model behaves?

Q4. **Is there a big gap between training and testing accuracy?**
If yes, what could this mean? Is the model memorizing the data (overfitting) or struggling to learn (underfitting)?

Q5. Which k value did you finally choose for your model? Why?
Explain your choice using both training and testing scores. Try to give a balanced and practical reason.

Step 20: Plotting Training vs Testing Accuracy (KNN)

To help students visually compare how the model performs on training data and testing data for different values of **k** in the K-Nearest Neighbors algorithm.

1. **Understand the Purpose**
   In this step, we want to see how the model's accuracy changes when we try different values of **k** (which means how many neighbors we use to make a prediction).
2. **Prepare the Accuracy Scores**

- You should already have two sets of scores:
    - One for training accuracy (how well the model does on the data it learned from).
    - One for testing accuracy (how well the model does on new data it hasn't seen before).

3. **Set the Plot Size**
    - Decide how big the chart should be. We want it wide enough to clearly see the trend. A size like 12 inches wide and 5 inches tall works well.
4. **Draw the Line for Training Accuracy**
    - On the x-axis, show different values of **k** (for example, from 1 to 14).
    - On the y-axis, plot the training accuracy for each **k** value.
    - Use a marker (like a star) to make the points easy to see.
5. **Draw the Line for Testing Accuracy**
    - Again, use the same x-axis (values of **k**).
    - This time, plot the testing accuracy for each **k** value.
    - Use a different marker (like a circle) and a different color so you can tell the lines apart.
6. **Add a Legend**
    - This helps the viewer understand which line represents training accuracy and which one shows testing accuracy.
7. **Interpret the Chart**
    - Look for the value of **k** where the testing accuracy is high and close to training accuracy.
    - Avoid values where training accuracy is very high but testing accuracy is low. That means the model is memorizing but not generalizing.

Questions for Report

Q1. What do you observe about the training accuracy as the value of K increases?

- Is it always high, or does it decrease?
- What does that tell you about the model's ability to memorize the training data?

Q2. At which value of K does the testing accuracy seem to be the highest?

- Why might this be a good choice for the final model?
- What does a high testing accuracy mean in real-life terms (e.g., predicting diabetes correctly)?

Q3. Is there a big difference between training and testing accuracy at some K values?

- What could that indicate about the model? (Hint: overfitting or underfitting)

Q4. Why is it important to compare both training and testing accuracy when selecting a machine learning model?

- What could go wrong if we only looked at one of them?

Step 21: Evaluating the Final Model with K-Nearest Neighbors (KNN)

You will now train a machine learning model using the K-Nearest Neighbors (KNN) method and evaluate how well it performs in predicting diabetes.

1: Set the number of neighbors (K)
Decide how many nearby data points (neighbors) the model should look at when making a prediction. Based on your previous steps, choose the value of K that gave the best results (for example, K = 11). This number comes from the earlier chart where you compared different K values.

2: Build your final model
Now that you've chosen your best K, you will build the KNN model using your training dataset. Make sure this dataset was balanced using SMOTE (you did this earlier to handle class imbalance).

3: Train the model
Train the model using the training dataset. This is where the model learns how health-related values (like glucose, insulin, BMI, etc.) relate to the outcome (diabetes or not).

4: Test the model on unseen data
After training, you must test the model using your separate test dataset (this was not used during training). This step checks how well the model performs on new patients.

5: Note the accuracy score
The model will give you an accuracy score — this is a number between 0 and 1 (or 0% to 100%) showing how many patients were predicted correctly.

Questions for Report

Q1. **What value of K (number of neighbors) did you choose for the KNN model, and why?**
Explain how you selected this number and what the training and testing scores looked like for this value.

Q2. **What is the final accuracy of your model on the test data?**
Do you think this accuracy is high enough for making real predictions? Why or why not?

Q3. **Is your model performing better on training data or testing data? What does this tell you about the model?**
Reflect on whether your model might be overfitting, underfitting, or balanced.

Q4. **How did applying SMOTE (oversampling) affect your model's performance?**
Did balancing the data improve the results? Why do you think that happened?

Q5. **If you had more time or resources, what would you try next to improve the model's accuracy or usefulness?**

Think creatively: Would you try a different algorithm, collect more data, or change how the features are selected?

Step 22: Calculating Precision, Recall, and F1 Score

To evaluate how well our machine learning model (KNN) is performing in predicting diabetes cases using three key metrics: Precision, Recall, and F1 Score.

1. **Start with your trained model**
   - Make sure you have already trained your KNN model using the training data that was balanced with SMOTE.
   - Also, confirm that you have used the test data to make predictions using this model.
2. **Predict outcomes on the test data**
   - Use your trained model to predict whether each person in the test set is diabetic or not.
   - This gives you the predicted labels (also called predicted values).
3. **Compare predictions with actual outcomes**
   - Check how many predictions were correct and how many were wrong by comparing the predicted values with the actual values from the test set.
4. **Understand what you are calculating**
   - **Precision** tells you: Out of all the people predicted as diabetic, how many were actually diabetic?
   - **Recall** tells you: Out of all the actual diabetic people, how many were correctly identified by the model?
   - **F1 Score** is the balance between Precision and Recall. It helps when we want a model that is both accurate and reliable.
5. **Use the right function or tool to calculate these scores**
   - There are ready-made tools in your environment to help you compute Precision, Recall, and F1 Score easily once you give it the actual and predicted values.
   - These tools will give you the three numbers that you will interpret in the next step.
6. **Interpret the results**
   - If **Precision is low**, your model is making too many false alarms.
   - If **Recall is low**, your model is missing too many actual diabetic cases.
   - If **F1 Score is low**, your model is unbalanced and needs improvement.
7. **Note your findings and add them to your project report**
   - Write down what each score means for your model.
   - Mention whether the model is suitable for use, and if not, what could be improved.

Questions for Report

Q1. **What does your model's precision score tell you?**

- Was your model mostly correct when it predicted a person has diabetes?
- Did it raise too many false alarms?

Samatrix.io

Q2. **What does the recall score reveal about your model's performance?**

- Was your model able to catch most of the actual diabetes cases?
- How many cases might it have missed?

Q3. **Why is the F1 score useful in this project?**

- How does it help you understand the overall balance between catching cases and avoiding false alarms?

Q4. **How did SMOTE affect the balance of your training data, and do you think it improved your model's performance?**

- Compare what might have happened if you didn't use SMOTE.

Q5. **Based on your evaluation (precision, recall, F1), what would you suggest to improve this model's performance further?**

- Would you try a different model, tune settings, or use more features? Why?