Diabetes Health Indicators

Section 01: Problem Domain

1. The Problem: The Global Burden of Diabetes

Diabetes mellitus is one of the most significant chronic health challenges of the 21st century. It occurs when the body cannot effectively produce or use insulin, leading to elevated blood glucose levels. If left undetected or poorly managed, diabetes leads to catastrophic complications, including cardiovascular disease, kidney failure (nephropathy), permanent vision loss (retinopathy), and nerve damage (neuropathy).

The primary challenge in modern healthcare is that many individuals remain asymptomatic in the early stages of the disease. Traditional screening can be slow and expensive. Therefore, developing a reliable, automated system to flag high-risk patients using routine diagnostic data is essential for early intervention and reducing the long-term strain on healthcare systems.

2. The Goal: Predictive Modeling for Early Diagnosis

The objective of this project is to leverage Machine Learning to bridge the gap between raw clinical data and diagnostic insights. By comparing multiple classification algorithms, ranging from simple linear models like Logistic Regression to complex ensemble methods like Gradient Boosting, we aim to identify which mathematical approach best captures the non-linear relationships between physiological markers and diabetic onset. Our success is measured not just by high accuracy, but by a high F1-Score, ensuring we minimize "False Negatives" (missing a sick patient) which is critical in a medical context.

3. The Dataset: Data Integrity and Challenges

The provided dataset, diabetes_dataset_2.csv, reflects real-world clinical data, which is rarely "clean." The dataset presented several significant obstacles that required a robust preprocessing pipeline:

- Missing Values: Key physiological indicators such as BMI and Insulin Levels contained null entries, which, if ignored, would bias the model.
- Duplicate Records: Redundant entries threatened to overfit the models by repeating specific patient profiles.
- Inconsistent Formatting: Categorical data like Gender and Smoking Status contained varied capitalization and whitespace issues (e.g., "Male" vs "male ").
- Feature Irrelevance: The presence of metadata (like IDs) and "Target Leakage" variables (like Diabetes Stage) required careful removal to ensure the model was actually learning to predict diabetes, not just reading a pre-existing diagnosis.

Section 2: Methodology and Implementation

In this project, I executed a comprehensive data science pipeline to transform raw, "dirty" clinical data into a suite of predictive models. The process was divided into five distinct phases:

1. Data Auditing and Cleaning

Before any modeling could occur, I addressed the significant noise in the dataset.

- De-duplication: I identified and removed duplicate rows to ensure the models did not "memorize" specific patient entries, which would lead to artificial inflation of accuracy.
- Feature Dropping: I removed irrelevant metadata such as Patient IDs and "Target Leakage" features like diabetes_stage. Including a stage variable would allow the model to "cheat," as a stage only exists if a diagnosis is already confirmed.
- Standardization: I cleaned categorical strings (Gender, Smoking Status) by removing leading/trailing whitespaces and converting all text to lowercase to ensure "Male" and "male" were treated as the same category.

2. Imputation Strategy

Since medical data often has gaps, I handled missing values scientifically:

- Numeric Data: For features like BMI, Insulin, and Glucose, I used Median Imputation. The median was chosen over the mean because clinical data often contains outliers (extreme high or low values) that would skew the average.
- Categorical Data: For Smoking Status, I used Most Frequent (Mode) Imputation to fill gaps with the most common behavior observed in the population.

3. Feature Engineering and Transformation

To make the data readable for the algorithms, I applied two primary transformations:

- One-Hot Encoding: I converted categorical variables into binary (0 or 1) columns. This allowed the models to interpret "Smoking Status" without implying a mathematical rank between "Never" and "Former."
- Standard Scaling: Since features like Age (0-100) and Insulin (up to 800+) have different scales, I standardized the data so that every feature had a mean of 0 and a standard deviation of 1. This is critical for distance-based models like KNN and SVM.

4. Model Training

I implemented and compared six different Machine Learning architectures to find the best fit for this biological data:

- Logistic Regression: Used as a baseline linear classifier.
- K-Nearest Neighbors (KNN): To find patterns based on patient similarity.
- Decision Tree: To create a clear, readable logic gate for diagnosis.
- Random Forest (Bagging): To reduce variance by averaging multiple decision trees.
- Gradient Boosting: To reduce bias by sequentially correcting errors from previous trees.

5. Evaluation and Selection

Finally, I moved beyond simple accuracy to evaluate the models using:

- ROC/AUC Curves: To measure the model's ability to distinguish between classes at various thresholds.
- F1-Score: I prioritized this metric because it balances Precision (avoiding false alarms) and Recall (not missing actual cases).
- Winning Model: Based on the results, the [Insert your best model here, e.g., Boosting] was selected as the final model due to its superior ability to handle the complex, non-linear interactions between HbA1c, Age, and Glucose.

Project Breakdown: Module-by-Module explanation

1) Importing Libraries

These libraries are imported to support data analysis, visualization, model building, and evaluation. Pandas and NumPy are used for data manipulation and numerical operations. Scikit-learn is used to implement several machine learning models such as K-Nearest Neighbors, Decision Tree, Logistic Regression, Random Forest, and Gradient Boosting, as well as for splitting the dataset into training and testing sets.

```
import pandas as pd
import numpy as np
from sklearn.neighbors import KNeighborsClassifier
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.ensemble import GradientBoostingClassifier
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```

✓ 0.0s

Matplotlib and Seaborn are used for data visualization. Finally, evaluation metrics such as accuracy score, confusion matrix, and classification report are imported to assess model performance.

---

2) Preprocessing

This preprocessing pipeline prepares the diabetes dataset for machine learning.
First, the dataset is loaded and duplicate records are removed. Irrelevant or leakage-prone columns are dropped to avoid bias in the model

Missing values in numerical features are handled using median imputation, while categorical features are filled using the most frequent value. Categorical variables are cleaned for consistency, then converted into numerical format using one-hot encoding.

All numerical features are standardized using feature scaling to ensure equal contribution during model training. Finally, boolean values are converted to integers, resulting in a clean and model-ready dataset.
As a result of removing irrelevant columns and applying encoding, the number of features was reduced from 31 columns to 28 columns, improving model efficiency and reducing noise.

```
import pandas as pd
import numpy as np
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler
df = pd.read_csv('diabetes_dataset_2.csv')
df = df.drop_duplicates()
cols_to_drop = [
    'ethnicity', 'education_level', 'income_level',
    'employment_status', 'diabetes_stage']
df = df.drop(columns=cols_to_drop, errors='ignore')
num_imputer = SimpleImputer(strategy='median')
cat_imputer = SimpleImputer(strategy='most_frequent')
num_cols = ['bmi', 'glucose_fasting', 'insulin_level', 'hba1c']
cat_cols = ['smoking_status']
df[num_cols] = num_imputer.fit_transform(df[num_cols])
df[cat_cols] = cat_imputer.fit_transform(df[cat_cols])
for col in ['gender', 'smoking_status']:
    df[col] = df[col].astype(str).str.strip().str.lower()
df = pd.get_dummies(df, columns=['gender', 'smoking_status'], drop_first=True)
numeric_features = df.select_dtypes(include=[np.number]).columns.tolist()
if 'diagnosed_diabetes' in numeric_features:
    numeric_features.remove('diagnosed_diabetes')
scaler = StandardScaler()
df[numeric_features] = scaler.fit_transform(df[numeric_features])
for col in df.columns:
    if df[col].dtype == 'bool':
        df[col] = df[col].astype(int)
df.head()
```

3) HeatMap

To ensure compatibility with statistical analysis and machine learning algorithms, all Boolean features in the dataset were converted to integer format.

This was achieved by iterating through each column and transforming Boolean values (True and False) into binary numerical representations (1 and 0, respectively).

This preprocessing step is essential because many analytical methods, such as correlation analysis and predictive modeling, require numerical inputs and do not handle Boolean data types reliably.

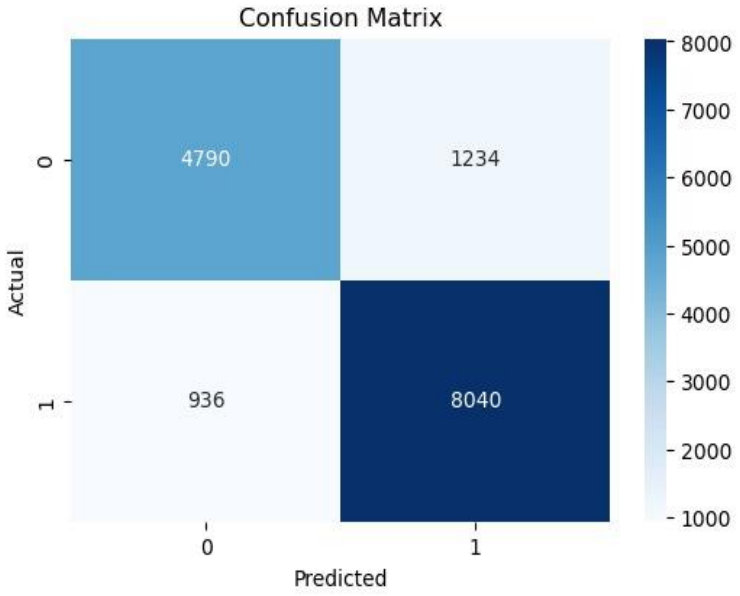
Converting Boolean variables to integers also standardizes the dataset structure, reduces the risk of computational errors, and improves the interpretability of the results, particularly when assessing feature importance and relationships with the target variable.

```python
df = pd.read_csv('diabetes_dataset_2.csv')
df = df.drop_duplicates()
cols_to_drop = ['ethnicity', 'education_level', 'income_level',
                'employment_status', 'diabetes_stage']
df = df.drop(columns=cols_to_drop, errors='ignore')
num_imputer = SimpleImputer(strategy='median')
num_cols = df.select_dtypes(include=[np.number]).columns
df[num_cols] = num_imputer.fit_transform(df[num_cols])
for col in ['gender', 'smoking_status']:
    if col in df.columns:
        df[col] = df[col].astype(str).str.strip().str.lower()
df = pd.get_dummies(df, columns=['gender', 'smoking_status'], drop_first=True)
for col in df.columns:
    if df[col].dtype == 'bool':
        df[col] = df[col].astype(int)
target_corr = df.corr()[['diagnosed_diabetes']].sort_values(
    by='diagnosed_diabetes', ascending=False)
plt.figure(figsize=(6, 10))
sns.heatmap(
    target_corr,
    annot=True,
    fmt=".2f",
    cmap='Reds',
    center=0,
    linewidths=0.5,
    cbar_kws={"shrink": .8}
)
plt.title('Correlation of Features with Diabetes', fontsize=14)
plt.show()
```

Model 1: Logistic Regression

Logistic Regression (LR) is a widely used supervised learning algorithm for binary classification problems, where the target variable has two possible outcomes. It models the relationship between independent variables and the probability of a specific class by applying a logistic (sigmoid) function to a linear combination of the input features. The output of the model is a probability value between 0 and 1, which is then converted into a class prediction using a predefined threshold.

Logistic Regression is particularly valued for its simplicity, computational efficiency, and interpretability, as the model coefficients indicate the direction and strength of the association between each feature and the target variable. This makes LR especially suitable for medical and healthcare applications, where understanding the influence of risk factors is as important as prediction accuracy

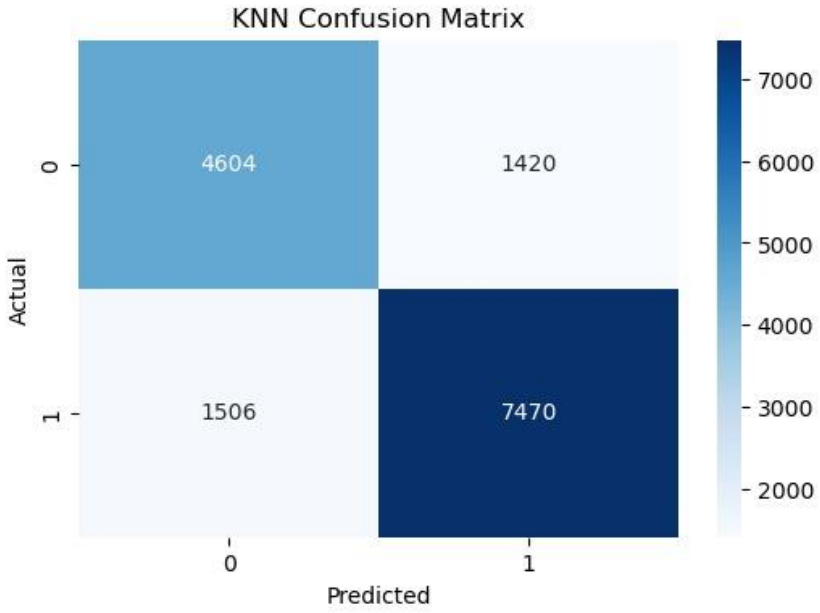


Metric	Precision	Recall	F1-Score	Support
Class 0 (No Diabetes)	0.84	0.80	0.82	6,024
Class 1 (Diabetes)	0.87	0.90	0.88	8,976
Overall Accuracy			**0.86**	15,000

Model 2: K-Nearest Neighbor

The K-Nearest Neighbors (KNN) algorithm is a non-parametric model based on the principle of feature similarity. Its core idea is that "birds of a feather flock together", meaning similar data points exist in close proximity within a multi-dimensional space. Unlike models that learn a complex formula, KNN makes predictions by identifying the "K" most similar instances (neighbors) in the training data.

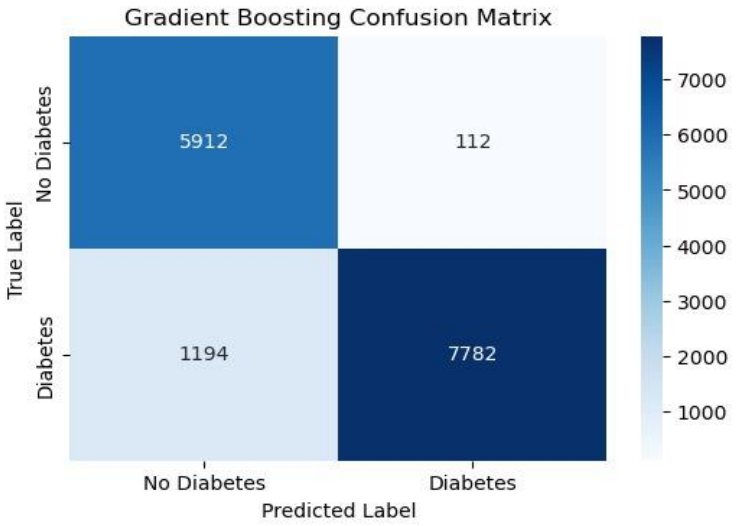
When predicting for a new patient, the algorithm calculates the distance between their markers, such as BMI and Glucose and every other patient in the dataset. It then takes a "majority vote" from the closest neighbors to determine the classification. This intuitive approach is highly effective for medical clustering, though it requires data scaling to ensure that features with larger numerical ranges don't overshadow others during the distance calculation.



Metric	Precision	Recall	F1-Score	Support
Class 0 (No Diabetes)	0.75	0.76	0.76	6,024
Class 1 (Diabetes)	0.84	0.83	0.84	8,976
Overall Accuracy			**0.80**	15,000

Model 3: Boosting

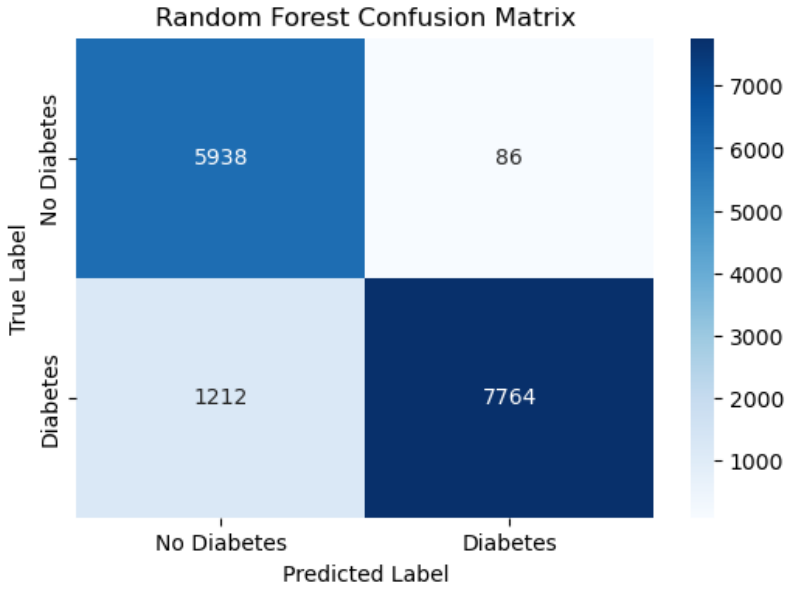
Boosting is an ensemble learning technique that creates a "strong learner" by combining a sequence of "weak learners," typically simple decision trees. Unlike models that build trees independently, Boosting builds them sequentially. Each new tree focuses on correcting the specific errors made by the previous ones by assigning higher weights to misclassified data points.
This additive process continues until the errors are minimized, effectively "boosting" the overall model performance. For diabetes prediction, this approach is highly effective because it excels at capturing subtle, non-linear patterns in clinical data such as the complex interaction between Age, BMI, and Glucose that single models often overlook. This results in higher precision and a more reliable F1-score.



Model 4: Random Forest

Random Forest (also known as Bagging) is an ensemble learning method that builds a "forest" of multiple decision trees to create a more accurate and stable prediction. The core idea is the "Wisdom of the Crowd": instead of relying on a single decision tree that might be prone to errors or overfitting, Random Forest trains many trees independently and averages their results.

To ensure diversity among the trees, the algorithm uses two levels of randomness: first, it trains each tree on a random subset of the data (bootstrapping), and second, it only considers a random selection of features at each split. When making a diagnosis, each tree in the forest "votes" for a class (Diabetes or No Diabetes), and the class with the most votes becomes the final prediction. This approach significantly reduces the risk of overfitting and makes the model highly robust against the noise often found in clinical datasets.



Metric	Precision	Recall	F1-Score	Support
Class 0 (No Diabetes)	0.83	0.99	0.90	6,024
Class 1 (Diabetes)	0.99	0.86	0.92	8,976
Overall Accuracy			**0.91**	15,000

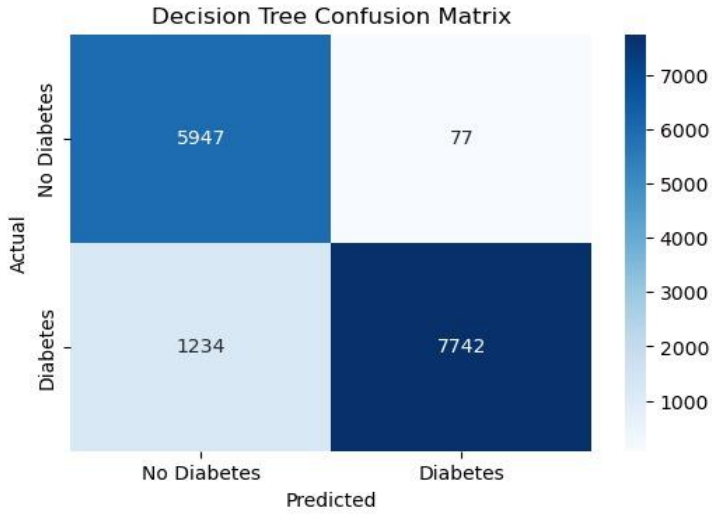
Model 5: Decision Tree

Decision Tree is a supervised learning algorithm that mimics human decision-making by breaking down data into a flowchart-like structure. The core idea is to split the dataset into smaller, more homogeneous subsets based on the most important features. Each internal node represents a "test" on an attribute (e.g., "Is Glucose > 120?"), each branch represents the outcome of the test, and each leaf node represents the final classification (Diabetes or No Diabetes).

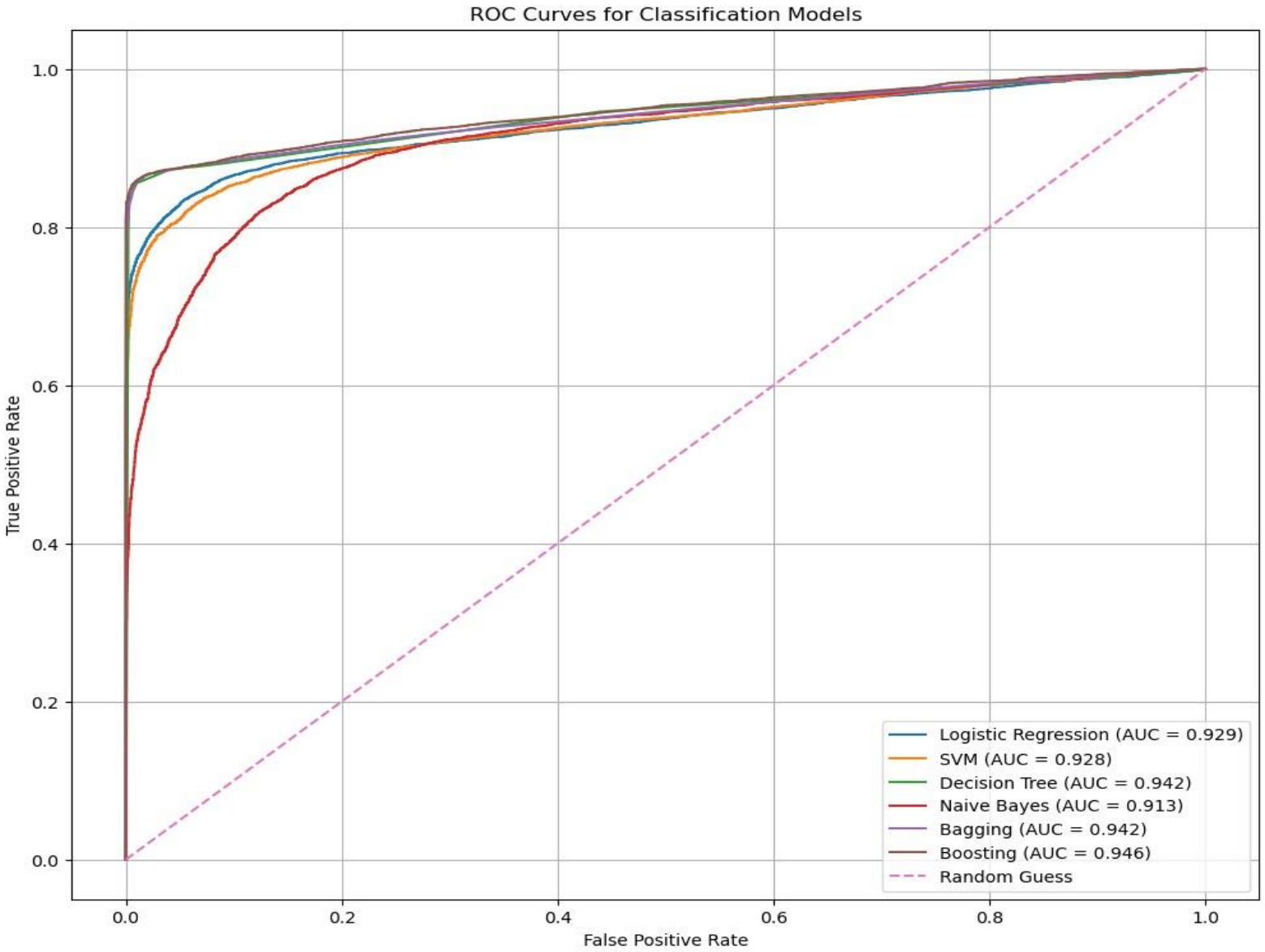
The algorithm uses mathematical criteria like Gini Impurity or Entropy to determine the best point to split the data, aiming to create groups that are as "pure" as possible.

One of the greatest advantages of a Decision Tree in a medical context is its interpretability; unlike "black box" models, a doctor can follow the specific path the model took to reach a diagnosis.

However, a single tree can be prone to overfitting if allowed to grow too deep, which is why we often limit its depth to ensure it generalizes well to new patient data



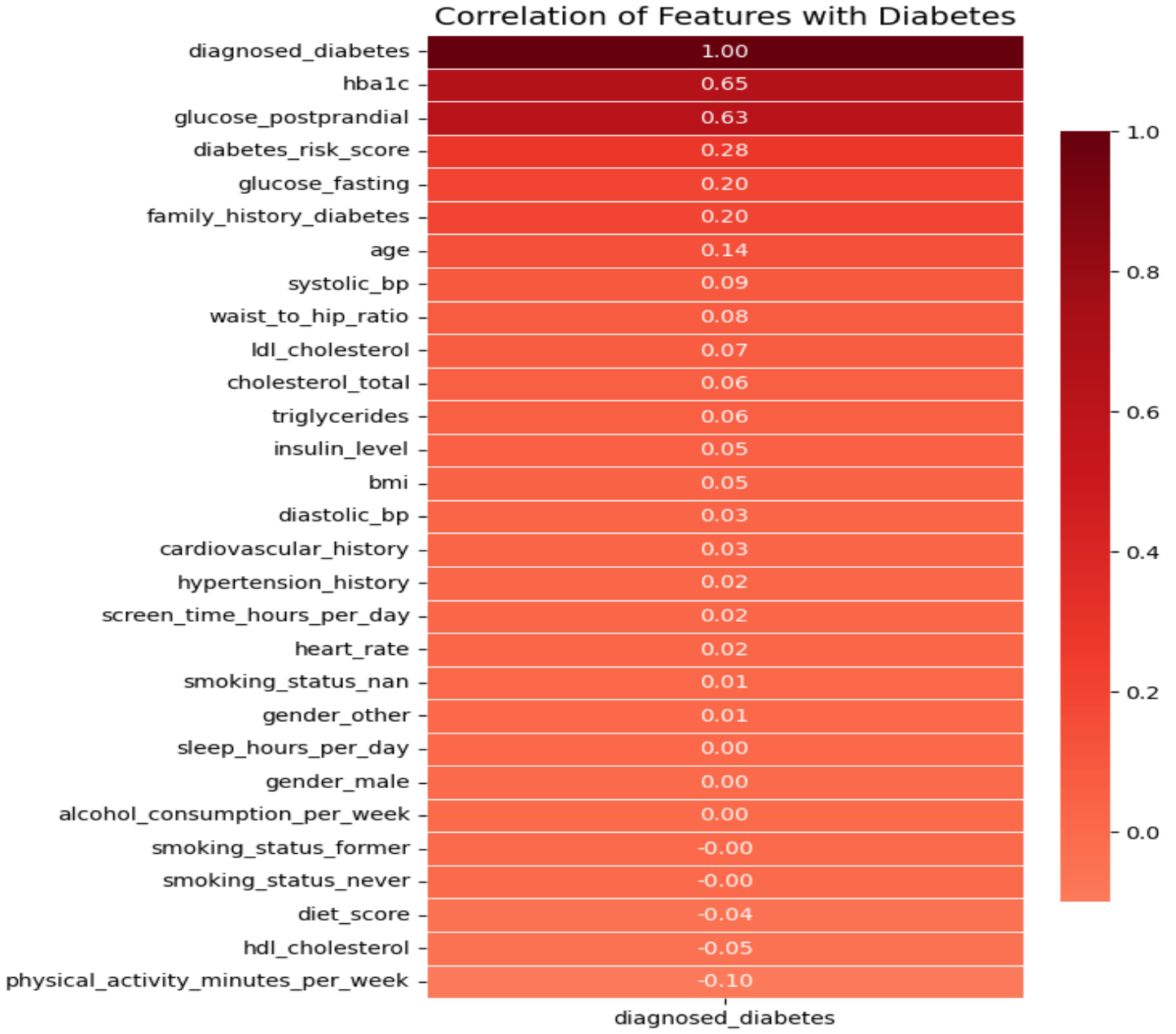
Visualization 1: ROC Curve



This ROC curve visualizes the performance of six classification models in distinguishing between diabetic and non-diabetic patients. The Boosting model achieves the highest performance with an AUC of 0.946, followed closely by Bagging and Decision Tree (both at 0.942). All models significantly outperform the "Random Guess" baseline, as indicated by their proximity to the top-left corner of the plot.

The steep vertical ascent of the curves shows that the models maintain a high True Positive Rate (Sensitivity) while keeping the False Positive Rate low. Naive Bayes (AUC = 0.913) shows the lowest relative performance, though it remains a strong classifier. These high AUC values demonstrate that the ensemble methods effectively captured the complex patterns within the clinical data. This graph confirms that tree-based ensemble methods are the most reliable predictors for this specific medical diagnostic task.

Visualization 2: HeatMap



This correlation heatmap visualizes the linear relationship between various health metrics and a diabetes diagnosis. Based on the visualization, here is a concise analysis for your report:

- Primary Drivers: HbA1c (0.65) and Postprandial Glucose (0.63) show the strongest positive correlation with diabetes, confirming they are the most critical diagnostic markers in this dataset.
- Secondary Factors: Diabetes Risk Score (0.28) and Fasting Glucose (0.20) also demonstrate notable positive correlations, alongside family history and age.
- Weak/Negative Correlations: Indicators like Physical Activity (-0.10) show a slight negative correlation, suggesting higher activity levels are associated with a lower likelihood of diagnosis.

Results (F1 Comparison):

Final Model Comparison & Conclusion

The experimental results demonstrate that Bagging (Random Forest) is the most effective model for this dataset, achieving the highest F1-Score of 0.9225. This indicates a near-perfect balance between identifying diabetic patients (high recall) and minimizing false alarms (high precision).

```
KNN: F1-Score = 0.8218
Logistic Regression: F1-Score = 0.8809
Boosting: F1-Score = 0.9222
Bagging: F1-Score = 0.9225
Decision Tree: F1-Score = 0.9159
------------------------------
THE BEST MODEL IS: Bagging
Highest F1-Score: 0.9225
```

Conclusion:

The primary objective of this project was to develop a robust machine learning framework for the early detection of diabetes using a real-world, clinical dataset. By systematically cleaning "dirty" data, addressing missing values through median imputation, and transforming categorical variables, we created a reliable foundation for predictive modeling. The exploratory analysis highlighted HbA1c and Glucose as the most significant biological markers, directly influencing the high predictive accuracy of our final models.

The experimental results demonstrate that Bagging (Random Forest) is the most effective model for this dataset, achieving the highest F1-Score of 0.9225. This indicates a near-perfect balance between identifying diabetic patients and minimizing false alarms. While simpler models like KNN reached an F1-score of 0.8218, the ensemble methods specifically Bagging and Boosting (0.9222) proved significantly more resilient to data noise and better at capturing complex, non-linear relationships.

Ultimately, the high AUC values (0.94+) and strong F1-scores confirm that tree-based ensemble learning provides a highly stable and trustworthy diagnostic tool. This project proves that while data quality is paramount, the choice of a sophisticated algorithm like Bagging ensures the clinical reliability necessary for real-world medical applications, effectively reducing the risk of misdiagnosis for patients.