# Diabetes Data Analysis Project Documentation

## Project Overview

This project analyzes the Pima Indians Diabetes Dataset to predict the onset of diabetes based on diagnostic measurements. The analysis includes data cleaning, exploratory data analysis (EDA), and predictive modeling.

## Dataset Description

* **Source:** Kaggle (mathchi/diabetes-data-set)
* **Rows:** 768 entries
* **Columns:** 9 features
* **Target Variable:** Outcome (0 = no diabetes, 1 = diabetes)

**Features:**

1. Pregnancies: Number of times pregnant
2. Glucose: Plasma glucose concentration
3. BloodPressure: Diastolic blood pressure (mm Hg)
4. SkinThickness: Triceps skin fold thickness (mm)
5. Insulin: 2-Hour serum insulin (mu U/ml)
6. BMI: Body mass index
7. DiabetesPedigreeFunction: Diabetes pedigree function
8. Age: Age in years
9. Outcome: Target variable (0 or 1)

## Understanding the Dataset and Cleaning

Several medical features cannot realistically be zero. These were identified and handled:

**Critical features with zero values:**

* Glucose (5 zeros)
* BloodPressure (35 zeros)
* SkinThickness (227 zeros)
* Insulin (374 zeros)
* BMI (11 zeros)

**Observations:**

* No null values in the original dataset, but many biologically impossible zero values
* Outcome distribution: 500 non-diabetic (0), 268 diabetic (1) cases
* Basic statistics show some features have wide ranges and potential outliers

**Distribution Analysis:**

* Glucose: Fairly symmetric (bell-like) distribution
* BloodPressure: Roughly symmetric distribution
* BMI: Fairly symmetric distribution
* These distributions supported using mean imputation for zero values

**Handling Missing Values:**

* Zero values in these critical features were replaced with np.nan
* Glucose, BloodPressure, and BMI zeros were replaced with their mean values
* SkinThickness and Insulin required further handling strategy using **KNN imputer**

**Why KNN Imputer?**  
We use KNN imputer because it considers the similarity between data points when filling missing values. Unlike simple methods like mean or median imputation, KNN preserves the natural relationships between features. For Insulin and SkinThickness, which depend on other factors like BMI, Glucose, and Age.

## Outlier Handling

1. **Insulin**
   * Step: Log transformation was applied (Insulin\_log) to reduce skewness and lessen the impact of extreme values
   * Result: Boxplots showed a more normalized distribution after log transformation, making extreme high values less influential
2. **SkinThickness**
   * Step 1: Outliers were detected using the IQR (Interquartile Range) method. Values beyond Q1 - 1.5IQR or Q3 + 1.5IQR were considered potential outliers
   * Step 2: Based on medical knowledge, any SkinThickness above 65 mm was deemed unrealistic and removed
   * Result: The dataset was filtered to exclude these extreme values, and boxplots after removal confirmed a cleaner distribution
3. **DiabetesPedigreeFunction (PDF)**
   * Observation: Most values naturally fell between 0.1 and 2.5, which is considered realistic
   * Action: No outlier removal was required because the data already lay within an acceptable range
   * Result: PDF distribution was reasonable without modifications

## Visualization

1. Correlation analysis between features
2. Distribution comparison between diabetic and non-diabetic groups
3. Pairplots and feature relationships visualization

## Modeling Approach

1. Data splitting into training and test sets
2. Feature selection based on EDA insights
3. Model selection (likely logistic regression, random forest, or gradient boosting)
4. Hyperparameter tuning
5. Model evaluation using appropriate metrics (accuracy, precision, recall, F1-score, ROC-AUC)
6. Cross-validation for robust performance estimation

## Model Development Strategy

After tuning a Random Forest with GridSearchCV and seeing minimal improvement, we shifted to training multiple models, including Logistic Regression, Decision Tree, Random Forest, and XGBoost, to compare performance. **AdaBoost** emerged as the best-performing model with **72.7% Accuracy** and an **F1-score of 0.604**, making it the final choice for diabetes prediction.

## AdaBoost Model Development

* **Data Preparation:** Features and target were separated, numeric features were scaled using StandardScaler, and class imbalance was addressed with **SMOTE**. Data was split 80:20 for training and testing with stratification
* **Initial Evaluation:** An AdaBoost model with a DecisionTreeClassifier (max\_depth=1) as base estimator was trained. 5-fold stratified CV evaluated baseline F1-score for stability
* **Hyperparameter Tuning:** Grid search optimized n\_estimators, learning\_rate, and max\_depth using 5-fold CV, targeting F1-score improvement for the minority class
* **Final Evaluation:** The tuned model was tested on unseen data, reporting Accuracy, ROC-AUC, and Classification Report to assess generalization and class-specific performance

**Summary:** Combining scaling, SMOTE, cross-validation, and hyperparameter tuning, this workflow produces a robust AdaBoost model for diabetes prediction

## Model Deployment and User Interaction

The final AdaBoost model has been deployed using **Streamlit**, providing an interactive interface for users. Users can input their personal health values (e.g., Glucose, BMI, Insulin, SkinThickness) to receive a prediction indicating the likelihood of diabetes.

To enhance the user experience and provide practical guidance, the application includes a **personalized checklist of recommended “Dos and Don’ts”**, helping users understand lifestyle and health measures based on their prediction results. This approach not only delivers the model’s output but also offers actionable insights for managing or preventing diabetes, adding a personalized and user-friendly dimension to the application.

## Integration of Gemini LLM API for Personalized Recommendations

The application uses the **Gemini LLM API** to provide a personalized checklist of “Dos and Don’ts” based on the diabetes prediction result. After a user inputs their health data and receives a prediction (diabetic or non-diabetic), the prediction outcome is sent to the API.

The **Gemini LLM** then generates a tailored list of recommendations corresponding to the user’s result—different advice is provided for diabetic and non-diabetic cases. This approach ensures that users receive relevant, context-specific guidance, enhancing the practicality and personalized experience of the application.



