Diabetes Prediction

This repository contains code for analyzing a diabetes dataset and developing a machine learning model to predict the likelihood of diabetes in individuals. The code includes data exploration, visualization, preprocessing, model training, and deployment using Gradio for an interactive prediction interface.

Table of Contents

Introduction

This project aims to predict the likelihood of diabetes in individuals using a variety of health metrics. The analysis involves:

- Visualizing the data to understand patterns and distributions.
- Preprocessing the data to handle imbalances and encode categorical variables.
- Training machine learning models to predict diabetes.
- Deploying the model using Gradio for an interactive user interface.

Data Exploration and Visualization

The code begins with loading the dataset and performing basic exploratory data analysis (EDA) to understand the structure and characteristics of the data. Key steps include:

- Checking for missing values.
- Examining the distribution of diabetes across different genders.
- Visualizing the relationship between various health metrics (like age, HbA1c level, and blood glucose level) and diabetes.

Key Visualizations:

- Diabetes Count by Gender: A bar plot showing the count of diabetic and non-diabetic individuals across different genders.
- Age Distribution of Diabetic Patients: A histogram displaying the age distribution of diabetic patients.

- Diabetes Prevalence by Age and Gender: Line plots showing diabetes prevalence across different ages for each gender.
- Feature Importances: A bar plot showing the importance of each feature in predicting diabetes using a Random Forest model.
- HbA1c Level vs. Blood Glucose Level: A scatter plot illustrating the relationship between HbA1c levels, blood glucose levels, and diabetes status.

Data Preprocessing and Model Training

Steps:

1. Handling Class Imbalance: Using RandomUnderSampler to balance the dataset.
2. Encoding Categorical Variables: Converting categorical variables (gender and smoking history) into numerical values using one-hot encoding.
3. Train-Test Split: Splitting the data into training and testing sets.
4. Model Training: Training a RandomForestRegressor and an XGBClassifier with hyperparameter tuning.
5. Model Evaluation: Calculating and displaying metrics like Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared value for the model.

Interactive Prediction Interface

An interactive user interface is created using Gradio

To allow users to input their health metrics and get a prediction of their diabetes risk. The Gradio interface takes the following inputs:

- Gender
- Age
- Hypertension status
- Heart disease status
- Smoking history
- BMI
- HbA1c level
- Blood glucose level

The model then provides a probability of diabetes based on the input values.

How to Run

Prerequisites

- Python 3.6 or higher
- Required libraries listed in the Dependencies section

Steps

1. ***Clone the repository:***

   git clone https://github.com/yourusername/diabetes-prediction.git

   cd diabetes-prediction

2. ***Install the dependencies:***

   pip install -r requirements.txt

3. ***Run the data exploration and model training script:***

   python diabetes_analysis_and_model.py

4. ***Run the Gradio interface for interactive predictions:***

   python gradio_interface.py

Dependencies

The required Python libraries are listed below. You can install them using pip:

pip install pandas matplotlib seaborn scikit-learn xgboost imbalanced-learn gradio joblib

Files in the Repository

- diabetes_analysis_and_model.py: The main script for data exploration, visualization, and model training.
- gradio_interface.py: The script to run the Gradio interface for interactive predictions.
- diabetes_prediction_dataset.csv: The dataset used for analysis and model training.
- requirements.txt: A file listing all the dependencies.
- README.md: This readme file.

Contributing

Contributions are welcome! Please fork the repository and submit a pull request with your changes. Ensure that your code follows the existing style and structure.