COLLEEGE OF COMPUTING DEPARTMENT OF SOFTWARE ENGINEERING

MACHINE LEARNING

INDIVIDUAL ASSIGNMENT

FEBRUARY 9, 2025

YAWUKAL MELAK
ID: 1403104

SUBMITTED TO: Mr Derbew

Diabetes Prediction System using Machine Learning

1. Introduction

Project Overview: The Diabetes Prediction System is a machine learning project aimed at predicting whether an individual has diabetes based on a set of medical parameters. This project leverages a classification model to solve a real-world problem and provides an easy-to-use interface through which users can input their health data and get a prediction.

Objective: The primary objective of this project is to demonstrate the full machine learning pipeline, including:

- Data acquisition
- Exploratory Data Analysis (EDA)
- Data preprocessing
- Model implementation
- Model evaluation
- Deployment of the model via an API using FastAPI and a user interface built with Streamlit

Problem Statement: Diabetes is a chronic medical condition that affects millions of people worldwide. Early prediction of diabetes is crucial for timely intervention and management. This project seeks to predict the likelihood of diabetes based on user inputs such as blood pressure, glucose levels, insulin levels, age, etc.

2. Data Acquisition & Source

Dataset Source: The dataset used for this project is sourced from the UCI Machine Learning Repository. It contains 768 rows and 9 columns representing various medical features related to diabetes. The dataset can be accessed from the following link: UCI Diabetes Dataset

Dataset Properties:

- **Rows:** 768
- **Columns:** 9 (Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI, DiabetesPedigreeFunction, Age, Outcome)
- **Target Variable:** Outcome (0 - Non-Diabetic, 1 - Diabetic)

License and Terms of Use: The dataset is available under the UCI Machine Learning Repository's open-access license, which allows free usage for educational and research purposes.

3. Problem Definition

The goal of this project is to classify individuals as diabetic or non-diabetic based on a set of medical features. The prediction task is a **classification problem**, as the target variable (Outcome) is categorical with two classes:

- 0: Non-Diabetic
- 1: Diabetic

The model is designed to help healthcare professionals in making early predictions about diabetes, enabling better healthcare management and intervention.

4. Data Understanding and Exploration

Loading the Dataset: The dataset is loaded into a Pandas DataFrame using the following code:

```
import pandas as pd
df = pd.read_csv("diabetes.csv")
```

Exploratory Data Analysis (EDA): During the EDA phase, we analyze the dataset to understand the distribution of features, check for missing values, identify outliers, and visualize relationships between features and the target variable.

- **Feature Distributions:**
    - Summary statistics and histograms are generated for each feature to understand their distributions.
    - For example, Glucose levels show a normal distribution, while BMI values have a skewed distribution.
- **Missing Values:**
    - We check for missing values using df.isnull().sum() and handle them accordingly.
- **Outliers:**
    - Boxplots are used to identify potential outliers in features like BMI and Insulin.
- **Visualizing Relationships:**
    - We use correlation matrices and scatter plots to visualize relationships between features and the target variable.

5. Data Preprocessing

Handling Missing Values: Any missing values in the dataset are handled by either removing the rows or replacing them with the mean or median values, depending on the feature.

Feature Scaling: Since the features are on different scales, we apply **StandardScaler** to normalize the numerical values so that they are all on the same scale (zero mean and unit variance).

Encoding Categorical Features: If any categorical features are present (in this case, none), they would be encoded using techniques like one-hot encoding.

Feature Selection: We ensure the dataset contains only the relevant features (Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI, DiabetesPedigreeFunction, and Age) for the prediction task.

6. Model Implementation and Training

Model Choice: For this classification task, a **Logistic Regression** model is chosen. Logistic Regression is suitable for binary classification tasks and works well with datasets like this one, where the relationship between the input variables and the output is assumed to be linear.

Data Splitting: The dataset is split into a training set (80%) and a testing set (20%) using:

```
from sklearn.model_selection import train_test_split

X = df.drop("Outcome", axis=1)

y = df["Outcome"]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Model Training: The Logistic Regression model is trained using the training set:

```
from sklearn.linear_model import LogisticRegression

model = LogisticRegression()

model.fit(X_train, y_train)
```

Hyperparameter Tuning: Hyperparameters such as C and solver are tuned using GridSearchCV to find the optimal configuration.

7. Model Evaluation and Analysis

Performance Metrics: The model is evaluated using the following metrics:

- **Accuracy**
- **Precision**
- **Recall**
- **F1-Score**
- **AUC-ROC Curve**

Model Evaluation:

- The model's accuracy on the test set is calculated and displayed.
- A confusion matrix is also generated to understand the number of true positives, false positives, true negatives, and false negatives.

Comparison Against Baseline: The Logistic Regression model is compared against a simple baseline model (e.g., DummyClassifier) to assess if the model provides a better performance than random guessing.

8. Model Deployment

Deployment Overview: The trained model is deployed as an API using **FastAPI** and **Streamlit**.

FastAPI API:

- The FastAPI app provides an endpoint /predict, which accepts input data (medical features) and returns the model's prediction (Diabetic or Non-Diabetic) along with the probability.

```
@app.post("/predict")
def predict_diabetes(data: DiabetesInput):
    # Preprocess input data and predict
    processed_data = preprocess(data)
    prediction = model.predict(processed_data)[0]
    probability = model.predict_proba(processed_data)[0][1]
    return {"prediction": int(prediction), "probability": float(probability)}
```

Streamlit UI:

- The Streamlit interface allows users to enter their health data through an interactive form and receive predictions.

```
if st.button("Predict"):
    response = requests.post(API_URL, json=input_data)
    st.write(f"Prediction: {'Diabetic' if prediction == 1 else 'Non-Diabetic'}")
    st.write(f"Probability: {probability * 100:.2f}%")
```

Deployment Details:

- The API is deployed on **Render** for public access. Users can interact with the Streamlit UI, input their medical data, and get predictions from the FastAPI server.

---

9. Future Improvements

- **Model Improvement:**
    - Investigate more complex machine learning models like Random Forests or Neural Networks to improve prediction accuracy.
- **Feature Engineering:**
    - Create new features based on domain knowledge (e.g., BMI-to-age ratio) to potentially improve model performance.
- **Real-Time Data:**
    - Integrate with real-time health data APIs to allow dynamic input for prediction.

10. Conclusion

This project demonstrates a full machine learning pipeline, from data acquisition and preprocessing to model deployment and evaluation. The system provides an easy-to-use interface for predicting diabetes, which could have practical applications in healthcare.

11. Code and Repository

The full code and project files can be accessed on the GitHub repository:
[git@github.com:YAWUKAL21/Diabetes-Prediction-Api.git]