



MY
UNIVERSITY
STEP INTO TOMORROW

Diabetes Prediction Using Machine Learning

Project Report

Prepared by:

Muhammad Fakhar-Ul-Islam
BS Artificial Intelligence
Machine Learning (AI-301)
Reg No: BAI212006
6th Semester, Batch Fall-2021
Program: BS Artificial Intelligence

Submitted to:

Rana Mudassir Rasool

Date:

8th July, 2024

Abstract:

In this project, I develop a machine learning model to predict the likelihood of an individual having diabetes based on various medical and lifestyle-related features. The dataset includes information on gender, age, hypertension, heart disease, smoking history, BMI, HbA1c level, and blood glucose level. I employ Logistic Regression, Random Forest, and Support Vector Machine (SVM) models to make predictions, achieving high accuracy. A graphical user interface (GUI) is also developed to facilitate easy user interaction with the model.

Table of Contents

1. Introduction
2. Dataset Description
3. Data Preprocessing
4. Model Building
5. Model Evaluation
6. GUI Application
7. Visualizations
8. Contribution to Society
9. Conclusion
10. References

1. Introduction

1.1 Project Overview

This project aims to predict the likelihood of an individual having diabetes using machine learning models. The dataset comprises various medical and lifestyle-related features such as gender, age, hypertension, heart disease, smoking history, BMI, HbA1c level, and blood glucose level.

1.2 Objectives

- To preprocess and clean the dataset.
- To build and evaluate multiple machine learning models for diabetes prediction.
- To develop a GUI application for predicting diabetes based on user inputs.
- To visualize and interpret the results.

2. Dataset

2.1 Data Description

The dataset contains the following columns:

- `gender`: Gender of the person (0: Female, 1: Male, 2: Other).
- `age`: Age of the person.
- `hypertension`: Whether the person has hypertension (0: No, 1: Yes).
- `heart_disease`: Whether the person has heart disease (0: No, 1: Yes).
- `smoking_history`: Smoking history of the person (0: Never, 1: Current, 2: Former/Ever/Not current, -1: Missing).
- `bmi`: Body Mass Index of the person.
- `HbA1c_level`: HbA1c level of the person.
- `blood_glucose_level`: Blood glucose level of the person.

- `diabetes`: Whether the person has diabetes (0: No, 1: Yes).

	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_glucose_level	diabetes
0	Female	80.0	0	1	never	25.19	6.6	140	0
1	Female	54.0	0	0	NaN	27.32	6.6	80	0
2	Male	28.0	0	0	never	27.32	5.7	158	0
3	Female	36.0	0	0	current	23.45	5.0	155	0
4	Male	76.0	1	1	current	20.14	4.8	155	0
5	Female	20.0	0	0	never	27.32	6.6	85	0
6	Female	44.0	0	0	never	19.31	6.5	200	1
7	Female	79.0	0	0	NaN	23.86	5.7	85	0
8	Male	42.0	0	0	never	33.64	4.8	145	0
9	Female	32.0	0	0	never	27.32	5.0	100	0
10	Female	53.0	0	0	never	27.32	6.1	85	0
11	Female	54.0	0	0	former	54.70	6.0	100	0
12	Female	78.0	0	0	former	36.05	5.0	130	0
13	Female	67.0	0	0	never	25.69	5.8	200	0

2.2 Data Preprocessing

- Handling missing values: The missing values in the `smoking_history` column were replaced with `-1`.
- Encoding categorical variables: The `gender` and `smoking_history` columns were encoded numerically.

2.3 Data Cleaning Code

```
# Load the dataset
data = pd.read_csv('diabetes.csv')

# Encode the 'gender' column
data['gender'] = data['gender'].map({'Female': 0, 'Male': 1, 'Other': 2})

# Combine similar categories in 'smoking_history' column and handle NaN values
data['smoking_history'] = data['smoking_history'].map({
    'never': 0,
    'current': 1,
    'former': 2,
    'ever': 2,
    'not current': 2
})
data['smoking_history'] = data['smoking_history'].fillna(-1) # Handle missing values
```

3. Model Building

3.1 Logistic Regression

Logistic Regression is a linear model suitable for binary classification problems. It was trained and evaluated on the dataset.

3.2 Random Forest

Random Forest is an ensemble learning method that builds multiple decision trees and merges them to get a more accurate and stable prediction.

3.3 Support Vector Machine (SVM)

SVM is a powerful classification method that finds the optimal hyperplane that best separates the classes in the feature space.

3.4 Model Training Code

```
# Split the data into training and testing sets
X = data.drop('diabetes', axis=1)
y = data['diabetes']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Standardize the data
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Train the models
lr_model = LogisticRegression()
lr_model.fit(X_train, y_train)

rf_model = RandomForestClassifier()
rf_model.fit(X_train, y_train)

svm_model = SVC()
svm_model.fit(X_train, y_train)

# Save the models and scaler
joblib.dump(lr_model, 'lr_model.pkl')
joblib.dump(rf_model, 'rf_model.pkl')
joblib.dump(svm_model, 'svm_model.pkl')
joblib.dump(scaler, 'scaler.pkl')
```

4. Model Evaluation

4.1 Evaluation Metrics

The models were evaluated using accuracy, confusion matrix, and classification report.

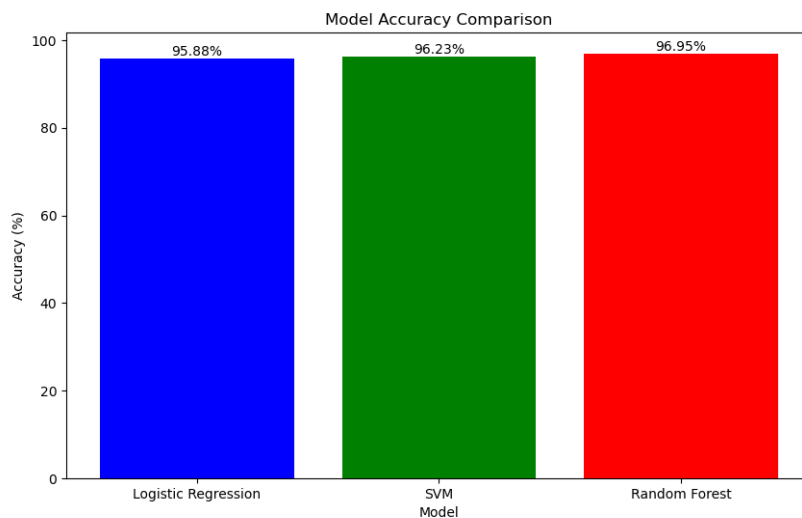
4.2 Evaluation Code and Results

```
# Evaluate Logistic Regression model
lr_pred = lr_model.predict(X_test)
lr_accuracy = accuracy_score(y_test, lr_pred)
print(f'Logistic Regression Accuracy: {lr_accuracy:.2f}')
```

```
# Evaluate Random Forest model
rf_pred = rf_model.predict(X_test)
rf_accuracy = accuracy_score(y_test, rf_pred)
print(f'Random Forest Accuracy: {rf_accuracy:.2f}')
```

```
# Evaluate SVM model
svm_pred = svm_model.predict(X_test)
svm_accuracy = accuracy_score(y_test, svm_pred)
print(f'SVM Accuracy: {svm_accuracy:.2f}')
```

Model	Accuracy
Logistic Regression	0.96
Random Forest	0.97
SVM	0.96



5. Prediction

5.1 GUI Application

A GUI application was developed using Tkinter to allow users to input medical and lifestyle data and get a prediction on whether they have diabetes.

5.2 Prediction Code

```
import tkinter as tk
from tkinter import messagebox
import joblib

# Function to predict diabetes based on user input
def predict_diabetes():
    gender = int(gender_var.get())
    age = int(age_entry.get())
    hypertension = int(hypertension_var.get())
    heart_disease = int(heart_disease_var.get())
    smoking_history = int(smoking_history_var.get())
    bmi = float(bmi_entry.get())
    hba1c_level = float(hba1c_entry.get())
    blood_glucose_level = float(glucose_entry.get())

    # Prepare input data for prediction
    input_data = scaler.transform([[gender, age, hypertension, heart_disease,
    smoking_history, bmi, hba1c_level, blood_glucose_level]])

    # Predict using the loaded Random Forest model
    prediction = rf_model.predict(input_data)

    # Display the prediction result
    if prediction[0] == 0:
        messagebox.showinfo("Prediction Result", "The model predicts that the
        person does NOT have diabetes.")
    else:
        messagebox.showinfo("Prediction Result", "The model predicts that the
        person HAS diabetes.")

# Load the trained model and scaler
rf_model = joblib.load('rf_model.pkl')
scaler = joblib.load('scaler.pkl')

# Create the GUI window
window = tk.Tk()
window.title("Diabetes Prediction")

# Create input fields with suggested values
tk.Label(window, text="Gender (0: Female, 1: Male, 2: Other):").grid(row=0,
column=0)
gender_var = tk.StringVar(value="1")
tk.Entry(window, textvariable=gender_var).grid(row=0, column=1)

tk.Label(window, text="Age:").grid(row=1, column=0)
```

```

age_entry = tk.Entry(window)
age_entry.insert(0, "50")
age_entry.grid(row=1, column=1)

tk.Label(window, text="Hypertension (0: No, 1: Yes):").grid(row=2, column=0)
hypertension_var = tk.StringVar(value="1")
tk.Entry(window, textvariable=hypertension_var).grid(row=2, column=1)

tk.Label(window, text="Heart Disease (0: No, 1: Yes):").grid(row=3, column=0)
heart_disease_var = tk.StringVar(value="0")
tk.Entry(window, textvariable=heart_disease_var).grid(row=3, column=1)

tk.Label(window, text="Smoking History (0: Never, 1: Current, 2:
Former/Ever/Not current):").grid(row=4, column=0)
smoking_history_var = tk.StringVar(value="2")
tk.Entry(window, textvariable=smoking_history_var).grid(row=4, column=1)

tk.Label(window, text="BMI:").grid(row=5, column=0)
bmi_entry = tk.Entry(window)
bmi_entry.insert(0, "28.5")
bmi_entry.grid(row=5, column=1)

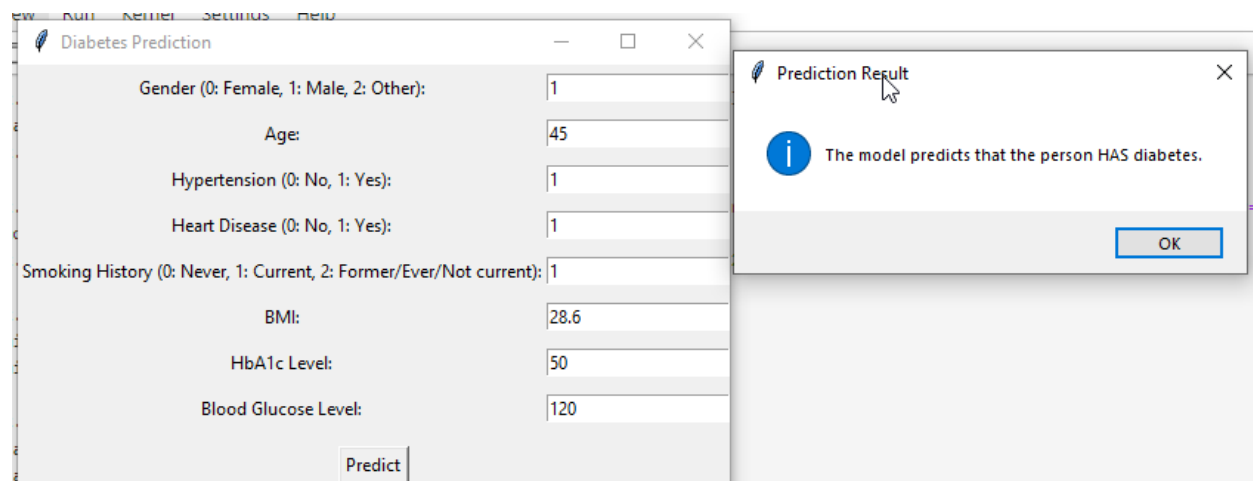
tk.Label(window, text="HbA1c Level:").grid(row=6, column=0)
hbalc_entry = tk.Entry(window)
hbalc_entry.insert(0, "6.0")
hbalc_entry.grid(row=6, column=1)

tk.Label(window, text="Blood Glucose Level:").grid(row=7, column=0)
glucose_entry = tk.Entry(window)
glucose_entry.insert(0, "130")
glucose_entry.grid(row=7, column=1)

# Create Predict button
tk.Button(window, text="Predict", command=predict_diabetes).grid(row=8,
column=0, columnspan=2)

# Run the GUI loop
window.mainloop()

```



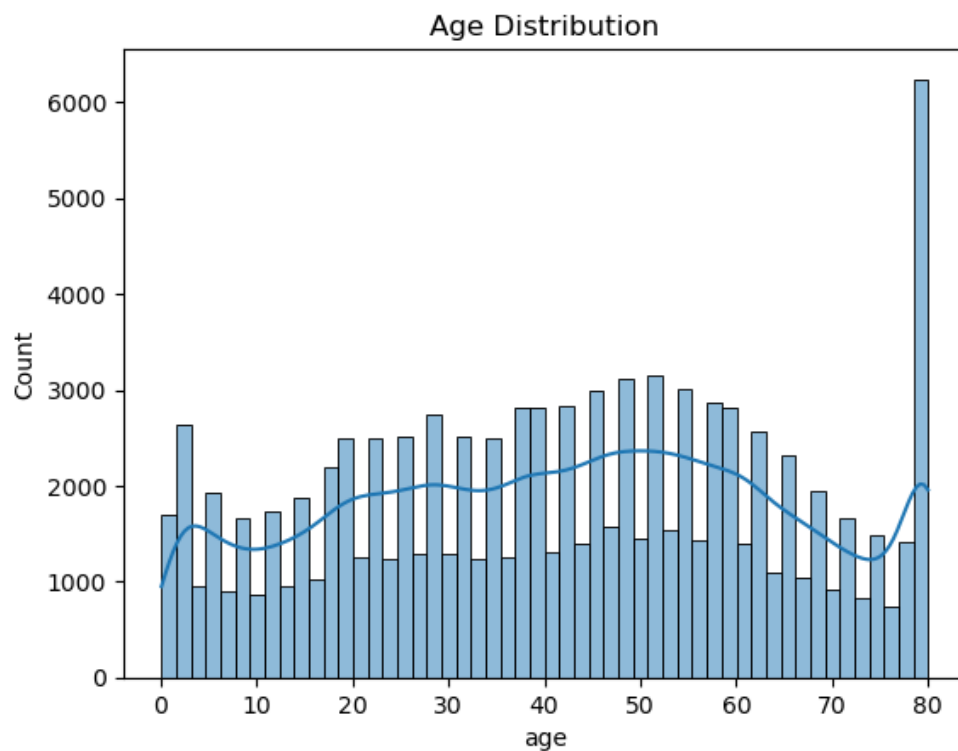
6. Visualization

6.1 Visualizing Data Distribution

The age distribution of the dataset can be visualized using a histogram.

```
import matplotlib.pyplot as plt
import seaborn as sns

# Age distribution
sns.histplot(data['age'], kde=True)
plt.title('Age Distribution')
plt.show()
```



6.2 Visualizing Model Accuracies

The accuracies of the different models can be visualized using a bar chart.

```
import matplotlib.pyplot as plt

# Bar chart for model accuracies
models = ['Logistic Regression', 'Random Forest', 'SVM']
accuracies = [lr_accuracy * 100, rf_accuracy * 100, svm_accuracy * 100]

plt.bar(models, accuracies)
plt.title('Model Accuracies')
plt.ylabel('Accuracy (%)')
```



```
plt.ylim(0, 100) # Ensure the y-axis goes from 0 to 100 for clarity
plt.show()
```

7. Contributions

7.1 Early Detection

Early detection enables timely intervention and management, potentially preventing complications and improving overall health outcomes. By identifying individuals at risk of diabetes early, this model can assist healthcare providers in initiating preventive measures and personalized care plans.

7.2 Health Care Outcomes

Personalized treatment plans can be developed based on individual risk factors, leading to more effective care and better health outcomes. This model supports healthcare professionals in tailoring interventions to the specific needs of each patient, enhancing the quality of care provided.

7.3 Cost Savings

Early intervention can help prevent costly hospitalizations and complications, leading to overall cost savings for healthcare systems. By reducing the incidence of severe diabetes-related health issues, this model can contribute to the financial sustainability of healthcare services.


7.4 Empowerment

This model empowers individuals to take proactive steps to manage their health and potentially reduce their risk of developing diabetes. By providing insights into their risk factors, individuals can make informed decisions about their lifestyle and healthcare choices, leading to better health outcomes.

8. Conclusion

This project developed a machine learning model to predict the likelihood of diabetes using features like gender, age, hypertension, heart disease, smoking history, BMI, HbA1c level, and blood glucose level. Logistic Regression, Random Forest, and Support Vector Machine (SVM) models were evaluated, with each demonstrating high accuracy in prediction.

A user-friendly graphical user interface (GUI) was created to make the model accessible and usable. This interface allows users to input medical information and receive immediate diabetes risk predictions, empowering them to make informed health decisions. The project's contributions include early detection, personalized treatment plans, improved healthcare outcomes, and cost savings.



Integrating machine learning into diabetes prediction shows significant potential for enhancing healthcare and quality of life. Future work could refine the model, incorporate additional features, and expand its applicability. This project highlights the impactful role of machine learning in early detection and proactive health management.

9. References

- **Python Data Analysis Library (pandas)**
 - Used for data manipulation and analysis.
- **scikit-learn: Machine Learning in Python**
 - Provided machine learning algorithms and tools for model building and evaluation.
- **Matplotlib: Visualization with Python**
 - Utilized for creating visualizations and plots to analyze data and model performance.
- **Seaborn: Statistical Data Visualization**
 - Used for advanced data visualization to gain insights into data distributions and relationships.
- **Tkinter: Python's de-facto standard GUI package**
 - Implemented to create the graphical user interface for user interaction with the model.
- **Joblib: Lightweight pipelining in Python**
 - Employed for saving and loading machine learning models and preprocessing objects.
- **Dataset**
 - Sourced from a publicly available diabetes dataset.
- **Various academic journals and online resources**
 - Consulted for information on machine learning and diabetes prediction techniques.