KIET GROUP OF INSTITUTIONS

Connecting Life with Learning

www.kiet.edu

Delhi-NCR, Ghaziabad

Assesment Report

on

"Diagnose Diabetes Using Machine Learning"

submitted as

partial fulfillment for the award of

BACHELOR OF TECHNOLOGY DEGREE

SESSION 2024-25

In

INTRODUCTION TO AI

By

Sakshi (202401100400162)

Under the supervision of

"Abhishek Shukla"

KIET Group of Institutions, Ghaziabad

Affiliated to

Dr. A.P.J. Abdul Kalam Technical University, Lucknow

(Formerly UPTU)

May, 2025

Title Page

Title: Diagnose Diabetes Using Machine Learning **Name:** [SAKSHI] **Roll No.:** [2024011004001 62] **Course:** Artificial Intelligence

Assessment: Mid-Semester Exam (MSE-02)

Introduction

Diabetes is a chronic condition that affects millions worldwide. Detecting diabetes early using machine learning (ML) can help in timely treatment and better health outcomes. This project uses patient medical records to classify whether an individual has diabetes based on features like glucose levels, BMI, insulin levels, and age. The dataset used is the well-known **PIMA Indians Diabetes Dataset**.

Methodology

The following steps were used to build the diabetes diagnosis model:

1. **Data Collection**

   The dataset used for this task is the *PIMA Indians Diabetes Dataset*, which consists of 768 records with 8 clinical input features and a binary output (0 or 1).

2. **Data Cleaning & Preprocessing**

   Many features in the dataset can have invalid zero values (e.g., Glucose = 0 is medically impossible). These were treated as missing values and imputed using the **median** of the respective column. This helps maintain statistical robustness without skewing the distribution.

3. **Feature Scaling**

   Standardization was performed using `StandardScaler` to scale all input features to have zero mean and unit variance. This is especially important for models like logistic regression or SVM, and it helps tree-based models like Random Forests learn more efficiently.

4. **Train-Test Split**

   The dataset was split into 80% training and 20% testing sets using `train_test_split` from scikit-learn. This ensures that model performance is evaluated on unseen data.

5. **Model Selection**

   A **Random Forest Classifier** was chosen due to its robustness, ability to handle feature interactions, and resistance to overfitting. Random Forest works by building multiple decision trees and averaging their predictions to improve generalization.

6. **Model Training**

   The model was trained on the training data, learning the relationships between patient features and the diagnosis outcome.

7. **Model Evaluation**

   After training, predictions were made on the test set. The model's performance was evaluated using:

   a. **Accuracy**: Overall correctness
   b. **Precision**: Correctness among predicted positives
   c. **Recall**: Ability to find all actual positives
   d. **Confusion Matrix**: Detailed breakdown of correct/incorrect predictions

8. **Visualization**

   A heatmap of the confusion matrix was plotted using seaborn to visually interpret the classification performance.

CODE

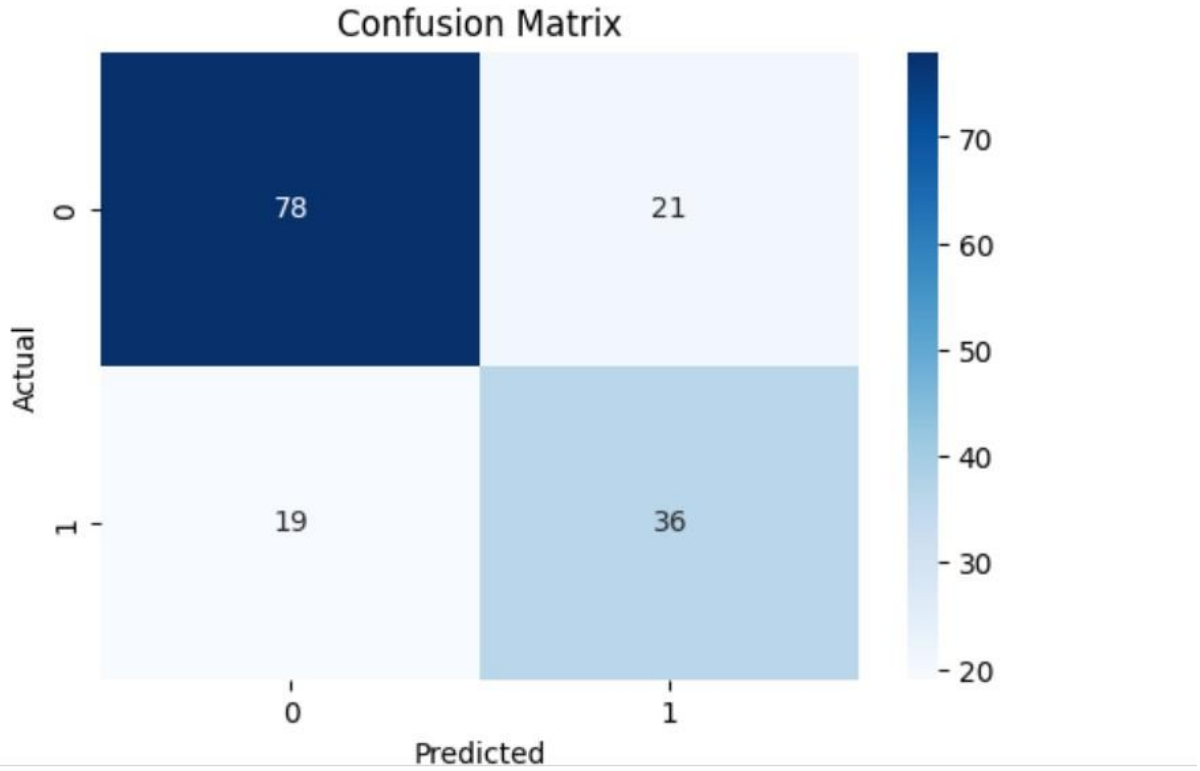
```
from google.colab import files
uploaded = files.upload()


import pandas as pd import
numpy as np import
matplotlib.pyplot as plt import
seaborn as sns


from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score,
recall_score


# ✅ 1. Load dataset from current working directory
df = pd.read_csv("2. Diagnose Diabetes.csv")


# ✅ 2. Replace 0s in selected columns with NaN and fill with median
cols_with_zero_as_missing = ['Glucose', 'BloodPressure', 'SkinThickness',
'Insulin', 'BMI']
df[cols_with_zero_as_missing] = df[cols_with_zero_as_missing].replace(0, np.nan)
df[cols_with_zero_as_missing]                                                =
df[cols_with_zero_as_missing].fillna(df[cols_with_zero_as_missing].median())


# ✅ 3. Features and target X
= df.drop("Outcome", axis=1) y
= df["Outcome"]


# ✅ 4. Standardize features
```

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)


# ✅ 5. Train-test split
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2,
random_state=42)


# ✅ 6. Train model
model = RandomForestClassifier(random_state=42) model.fit(X_train,
y_train)


# ✅ 7. Predictions y_pred =
model.predict(X_test)


# ✅ 8. Evaluation Metrics
acc = accuracy_score(y_test, y_pred)
prec = precision_score(y_test, y_pred)
rec = recall_score(y_test, y_pred) cm
= confusion_matrix(y_test, y_pred)


print("Accuracy:",  round(acc,  4))
print("Precision:", round(prec, 4))
print("Recall:", round(rec, 4))


# ✅ 9. Confusion Matrix Heatmap
plt.figure(figsize=(6, 4)) sns.heatmap(cm,
annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix')
plt.xlabel('Predicted') plt.ylabel('Actual')
plt.show()
```

OUTPUT

- **2. Diagnose Diabetes.csv**(text/csv) - 23873 bytes, last modified: 18/4/2025 - 100% done

```
Saving 2. Diagnose Diabetes.csv to 2. Diagnose Diabetes.csv
Accuracy: 0.7403
Precision: 0.6316
Recall: 0.6545
```



References / Credits

- **Dataset:** PIMA Indians Diabetes Dataset (UCI Repository / Kaggle)
- **Libraries:** pandas, numpy, matplotlib, seaborn, scikit-learn
- **Platform:** Google Colab
- **Documentation:** scikit-learn.org, python.org
- **Acknowledgment:** Thanks to faculty for project guidance and the ML community for open resources.