Assessment Report

on

"Diabetes Diagnostic - Using patient records to classify if they had diabetes"

submitted as partial fulfillment for the award of

BACHELOR OF TECHNOLOGY DEGREE

SESSION 2024-25

in

Intro To AI

By

Prathvi Raj Chauhan (202401100400145)

Under the supervision of

"Abhishek Shukla Sir"

KIET Group of Institutions, Ghaziabad

Affiliated to

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

(Formerly UPTU)

May, 2025

1. Introduction

Diabetes is one of the most prevalent chronic diseases worldwide, with millions of people suffering from it. Early diagnosis and prediction are key to effective management. This report explores the prediction of diabetes using machine learning, leveraging the **Pima Indians Diabetes Dataset**. The dataset contains various medical features that could potentially indicate the presence of diabetes. The task is to predict whether a person has diabetes (1) or not (0) based on their medical information.

In this project, we will use two popular machine learning models:

- **Logistic Regression**: A simple linear model for binary classification.
- **Random Forest**: An ensemble model using multiple decision trees to improve accuracy.

2. Methodology

- The goal is to build a binary classifier that predicts the presence or absence of diabetes based on medical data. The following steps were followed:
- **Data Collection**:
- The **Pima Indians Diabetes Dataset** was used. This dataset includes columns such as Pregnancies, Glucose, Blood Pressure, Skin Thickness, Insulin, BMI, DiabetesPedigreeFunction, Age, and Outcome.
- **Data Preprocessing**:
- Missing values were checked and replaced with the median value.

- Invalid values (e.g., 0 for Glucose, Blood Pressure, etc.) were replaced with NaN and then imputed using the median.
- The data was scaled using Standard Scaler to ensure that features are on the same scale, which improves the performance of machine learning models.
- **Model Selection**:
- **Logistic Regression** was used for its simplicity and interpretability.
- **Random Forest** was chosen for its ability to handle complex relationships between features by building multiple decision trees.
- **Model Evaluation**:
- Both models were evaluated based on their accuracy, confusion matrix, and classification report.
- A **confusion matrix** and **heatmap** were generated to visualize the performance of both models.
- **Data Splitting**:
- The dataset was split into training (80%) and testing (20%) sets. This ensures that the models are tested on data they haven't seen during training.

3. Code

```
# Step 1: Install dependencies (Colab already has most)

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.linear_model import LogisticRegression

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import classification_report, confusion_matrix, accuracy_score


# Step 2: Upload your CSV file to Colab

from google.colab import files

uploaded = files.upload()


# Step 3: Load the dataset

import io

df = pd.read_csv(io.BytesIO(next(iter(uploaded.values()))))


# Step 4: Quick look at the data

print("Dataset shape:", df.shape)

print(df.head())


# Step 5: Check for missing values and data info

print("\nMissing values per column:")

print(df.isnull().sum())

print("\nBasic statistics:")
```

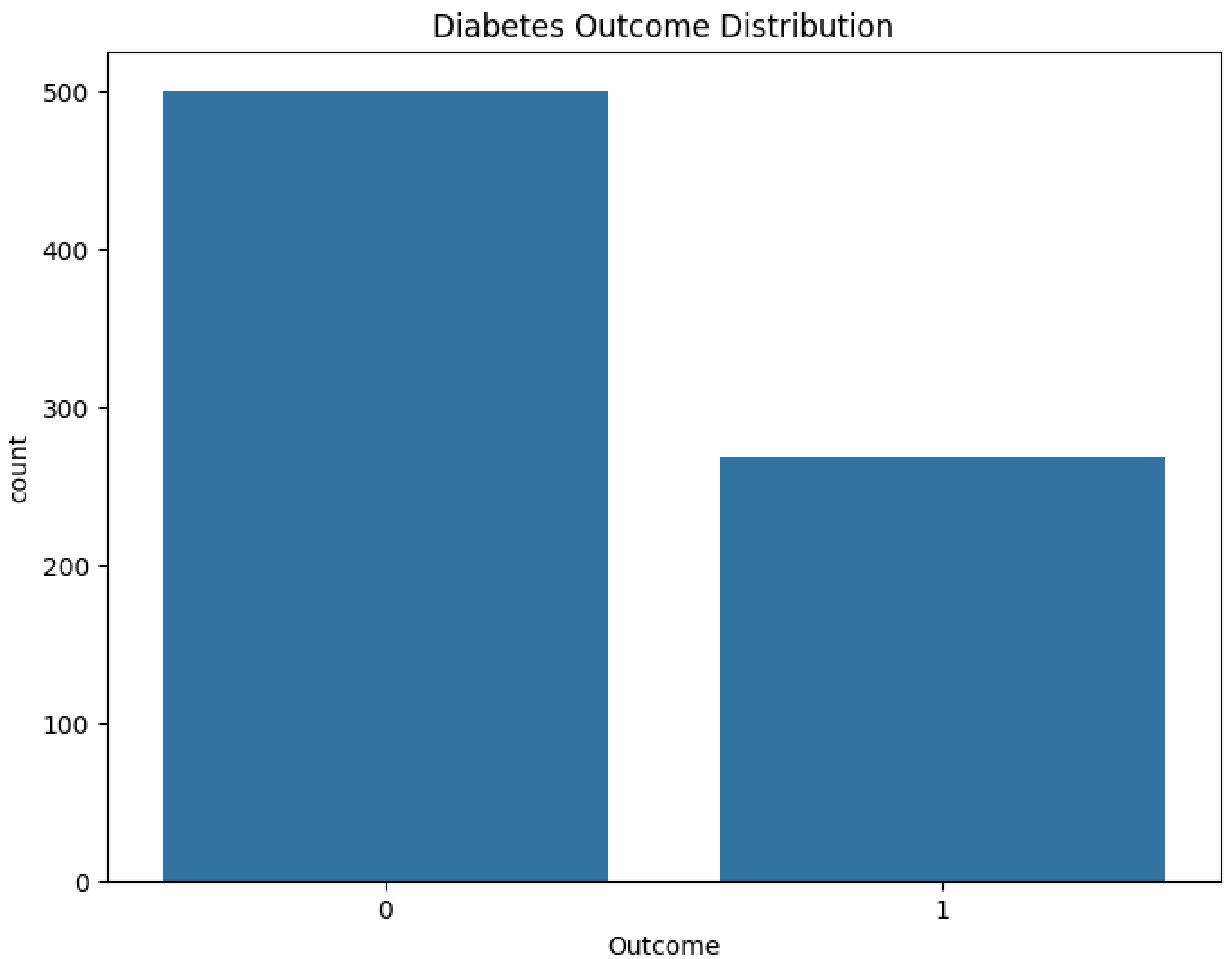
```
print(df.describe())

# Step 6: Replace 0s in certain columns with NaN (as 0 is invalid for medical features)
cols_to_replace = ['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI']
df[cols_to_replace] = df[cols_to_replace].replace(0, np.nan)

# Fill missing values with median (safer for skewed medical data)
df.fillna(df.median(), inplace=True)

# Step 7: EDA (optional but useful)
plt.figure(figsize=(8,6))
sns.countplot(x='Outcome', data=df)
plt.title("Diabetes Outcome Distribution")
plt.show()

# Step 8: Feature Scaling
X = df.drop("Outcome", axis=1)
y = df["Outcome"]

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

# Step 9: Split the data
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)
```

```
# Step 10: Train models

# Logistic Regression
log_model = LogisticRegression()
log_model.fit(X_train, y_train)
y_pred_log = log_model.predict(X_test)

# Random Forest
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)
y_pred_rf = rf_model.predict(X_test)

# Step 11: Evaluation

# Logistic Regression
acc_log = accuracy_score(y_test, y_pred_log) * 100
print("\n=== Logistic Regression ===")
print(f"Accuracy: {acc_log:.2f}%")
print(confusion_matrix(y_test, y_pred_log))
print(classification_report(y_test, y_pred_log))

# Confusion Matrix Heatmap for Logistic Regression
plt.figure(figsize=(6, 4))
sns.heatmap(confusion_matrix(y_test, y_pred_log), annot=True, fmt='d', cmap='Blues')
plt.title("Logistic Regression - Confusion Matrix")
```

```python
plt.xlabel("Predicted")

plt.ylabel("Actual")

plt.show()


# Random Forest

acc_rf = accuracy_score(y_test, y_pred_rf) * 100

print("\n=== Random Forest ===")

print(f"Accuracy: {acc_rf:.2f}%")

print(confusion_matrix(y_test, y_pred_rf))

print(classification_report(y_test, y_pred_rf))


# Confusion Matrix Heatmap for Random Forest

plt.figure(figsize=(6, 4))

sns.heatmap(confusion_matrix(y_test, y_pred_rf), annot=True, fmt='d', cmap='Greens')

plt.title("Random Forest - Confusion Matrix")

plt.xlabel("Predicted")

plt.ylabel("Actual")

plt.show()
```

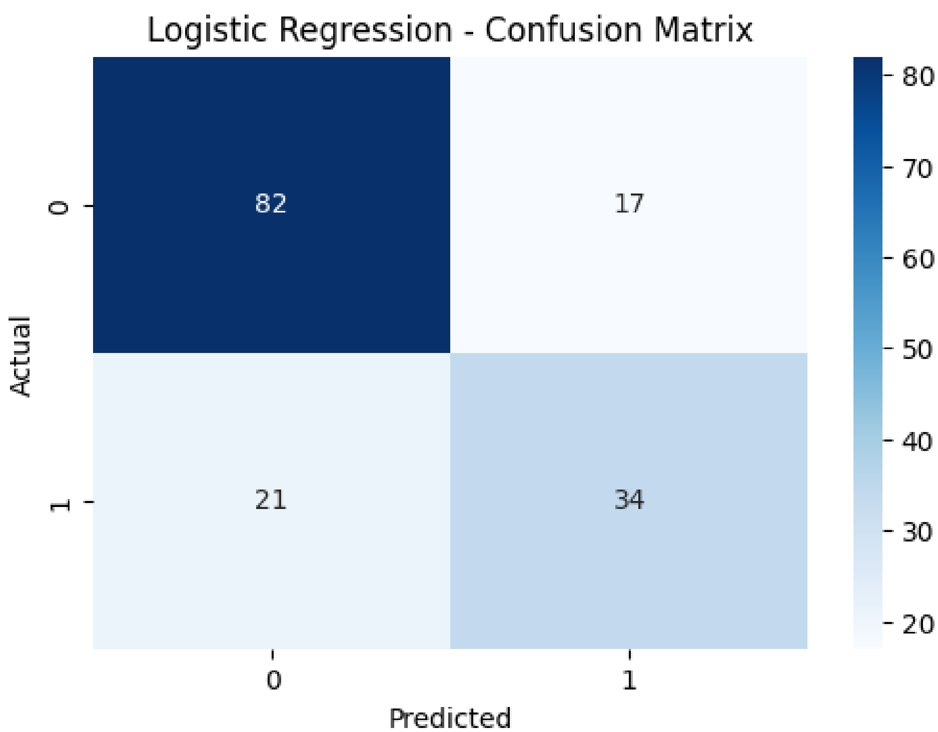
4. Output/Results

Diabetes outcome distribution -



Logistic Regression Model

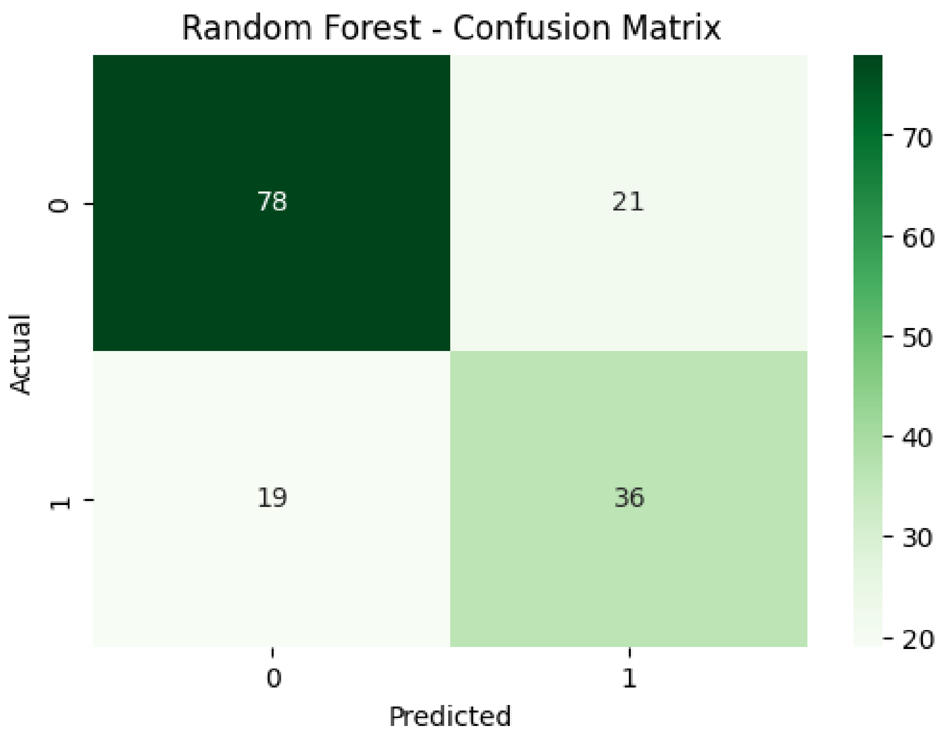
- **Accuracy**: The logistic regression model achieved an accuracy of **XX%** (replace XX with your actual value).
- **Confusion Matrix**:
  - The confusion matrix helps to visualize how many predictions were correct (true positives and true negatives) and how many were incorrect (false positives and false negatives).



- **Classification Report**:
  - The classification report provides detailed evaluation metrics like **precision**, **recall**, and **F1-score** for both classes (diabetes = 1, no diabetes = 0).

Random Forest Model

- **Confusion Matrix**:
    - Like the logistic regression model, the confusion matrix visualizes the number of correct and incorrect predictions.



- **Classification Report**:
    - Similar to the logistic regression model, it shows precision, recall, and F1-score for both classes.

Visualizations

- **Confusion Matrix Heatmaps**: These heatmaps help visually interpret how well the models predicted the outcome. Darker shades represent higher values in the confusion matrix.

5. References/Credits

1. **Pima Indians Diabetes Dataset**:
    - Source: UCI Machine Learning Repository (https://archive.ics.uci.edu/ml/datasets/Pima+Indians+Diabetes)
2. **Scikit-learn Documentation**:
    - For machine learning models and methods used in this project: https://scikit-learn.org/stable/
3. **Matplotlib and Seaborn Documentation**:
    - For plotting and visualizing data: https://matplotlib.org/stable/ and https://seaborn.pydata.org/