Assessment Report

on

"Diagnose Diabetes"

submitted as partial fulfillment for the award of

BACHELOR OF TECHNOLOGY DEGREE

SESSION 2024-25

in

CSE(AIML)

By

Divyansh Yadav (202401100400085)

Under the supervision of

"Mr. Abhishek Shukla Sir"

KIET Group of Institutions, Ghaziabad

Affiliated to

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

(Formerly UPTU)

May, 2025

Introduction:

This project involves diagnosing whether a patient has diabetes based on certain medical parameters. Using a dataset of patient records, a classification model is built to predict the presence of diabetes. The model's performance is evaluated using metrics such as accuracy, precision, and recall, with results visualized via a confusion matrix heatmap.

Methodology:

The dataset was uploaded and loaded using Google Colab's file upload functionality. The features were separated from the target variable (Outcome). The data was split into training and test sets. A StandardScaler was applied for feature scaling. Logistic Regression was chosen as the classification model and trained on the scaled data. Predictions were made on the test set and evaluated using a confusion matrix, accuracy, precision, recall, and a classification report.

Code:

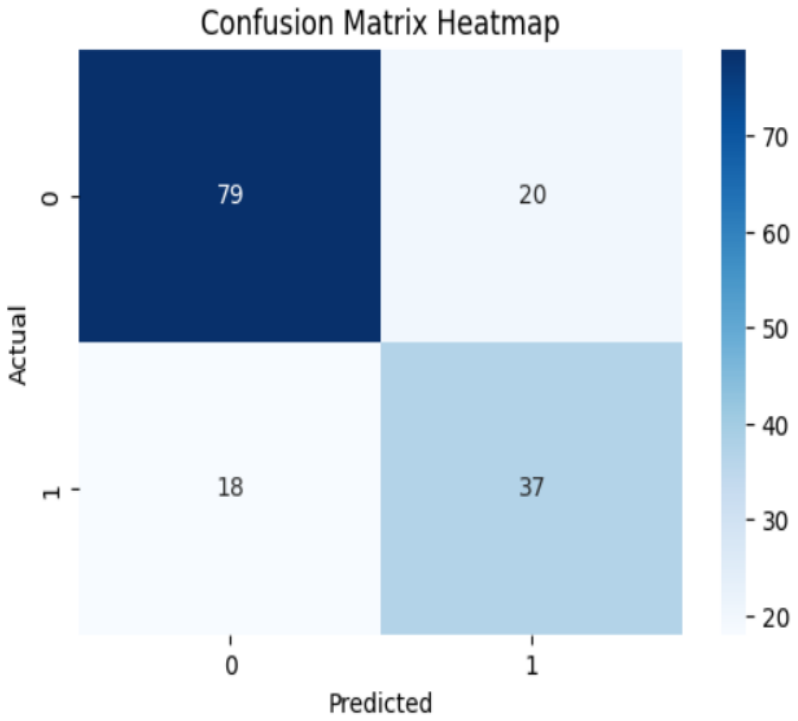
```
# Install missing dependencies if needed
# !pip install -q seaborn scikit-learn pandas

# Import libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score,
precision_score, recall_score
import seaborn as sns
import matplotlib.pyplot as plt

# For file upload in Google Colab
from google.colab import files
```

```
import io

# Upload the dataset file
uploaded = files.upload()

# Load the uploaded CSV file
for file_name in uploaded.keys():
   data = pd.read_csv(io.BytesIO(uploaded[file_name]))

# Check the first few rows
data.head()

# Define features and target
X = data.drop('Outcome', axis=1)
y = data['Outcome']

# Split the data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Scale the features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Initialize and fit a Logistic Regression model
model = LogisticRegression(random_state=42)
model.fit(X_train_scaled, y_train)

# Make predictions
y_pred = model.predict(X_test_scaled)

# Calculate confusion matrix
cm = confusion_matrix(y_test, y_pred)

# Plot heatmap of confusion matrix
plt.figure(figsize=(6,4))
```

```
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix Heatmap')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()

# Calculate evaluation metrics
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
report = classification_report(y_test, y_pred)

# Print metrics
print(f"Accuracy: {accuracy:.4f}")
print(f"Precision: {precision:.4f}")
print(f"Recall: {recall:.4f}")
print("\nClassification Report:\n", report)

# ---- New Feature: User Input for Diagnosis ----
print("\n--- Enter your medical details to check diabetes diagnosis ---")

# List of feature names
features = X.columns

# Collect user input values
user_data = []
for feature in features:
    value = float(input(f"Enter {feature}: "))
    user_data.append(value)

# Convert to dataframe and scale it
user_df = pd.DataFrame([user_data], columns=features)
user_scaled = scaler.transform(user_df)

# Make prediction
user_prediction = model.predict(user_scaled)
```

```
# Display result
if user_prediction[0] == 1:
    print("\n✅ Diagnosis: The person is likely to have Diabetes.")
else:
    print("\n✅ Diagnosis: The person is unlikely to have Diabetes.")
```

Result:




```
Accuracy: 0.7532
Precision: 0.6491
Recall: 0.6727

Classification Report:
              precision    recall  f1-score   support

           0       0.81      0.80      0.81        99
           1       0.65      0.67      0.66        55

    accuracy                           0.75       154
   macro avg       0.73      0.74      0.73       154
weighted avg       0.76      0.75      0.75       154
```

References:

Dataset Source: Pima Indians Diabetes Database
Code developed using: Python, scikit-learn, seaborn, matplotlib, Google Colab
Tools: Google Colab File Upload feature