Assessment Report

on

“Diagnose Diabetes”

SESSION 2024-25

By

Yuvraj Singh Taniya (202401100300292, CSE(AI)-D)

Under the supervision of

“Abhishek Shukla Sir”

Problem Statement: Diagnose Diabetes

Objective:

The goal is to **predict whether a person has diabetes** using a classification model trained on medical records. The model should analyze various health-related features and determine the likelihood of the individual having diabetes. This is a supervised machine learning task.

Background

Diabetes is a chronic condition that affects how the body processes glucose (blood sugar). Detecting diabetes early is crucial in preventing severe health complications. Traditionally, this diagnosis involves lab tests and clinical judgment. By applying machine learning techniques, we can assist healthcare professionals in making quicker, data-driven decisions using historical health data.

Dataset Description

The dataset used is the **Pima Indians Diabetes Dataset**, collected by the National Institute of Diabetes and Digestive and Kidney Diseases. It includes data on **female patients aged 21 years and older** of Pima Indian heritage.

Features (Input Variables):

1. **Pregnancies** – Number of times the patient has been pregnant
2. **Glucose** – Plasma glucose concentration (mg/dL)
3. **BloodPressure** – Diastolic blood pressure (mm Hg)
4. **SkinThickness** – Triceps skinfold thickness (mm)
5. **Insulin** – 2-Hour serum insulin (mu U/ml)
6. **BMI** – Body Mass Index (weight in kg / height in m^2)

7. **DiabetesPedigreeFunction** – A score based on family history of diabetes
8. **Age** – Age in years

Target Variable:

- **Outcome**:
  - 0: The individual does **not** have diabetes
  - 1: The individual **has** diabetes

Methodology

To solve the diabetes diagnosis problem, a structured machine learning approach was followed:

1. **Data Acquisition**
   The Pima Indians Diabetes Dataset was used, containing patient medical records and a binary outcome indicating diabetes presence.

2. **Data Preprocessing**
   - Features and target were separated.
   - Data was split into training and testing sets (80/20).
   - Feature scaling was applied using StandardScaler to normalize the data.

3. **Model Training**
   A Logistic Regression model was chosen for its simplicity and effectiveness in binary classification. The model was trained on the standardized training data.

4. **Prediction and Evaluation**
   - The model was tested on unseen data.
   - Performance was evaluated using accuracy, precision, recall, F1-score, and confusion matrix.
   - A heatmap of the confusion matrix was generated for visual analysis.

5. **Conclusion**
   The model's performance was interpreted through the evaluation metrics, highlighting its effectiveness in identifying diabetic cases.

Code:

```
# Import necessary libraries for data handling, visualization, and machine learning

import pandas as pd              # For data manipulation

import seaborn as sns            # For heatmap visualization

import matplotlib.pyplot as plt    # For plotting graphs

from sklearn.model_selection import train_test_split  # For splitting data into train and test sets

from sklearn.preprocessing import StandardScaler      # For standardizing (normalizing) data

from sklearn.linear_model import LogisticRegression   # For building a logistic regression model

from sklearn.metrics import (                    # For evaluating the model

    confusion_matrix,

    accuracy_score,

    precision_score,

    recall_score,

    f1_score,

    classification_report

)


# Load the dataset from the file

df = pd.read_csv("/content/2. Diagnose Diabetes.csv")


# Show basic information about the dataset (columns, data types, nulls, etc.)

print("Dataset Summary:")

print(df.info())


# Display the first 5 rows of the dataset for a quick look
```

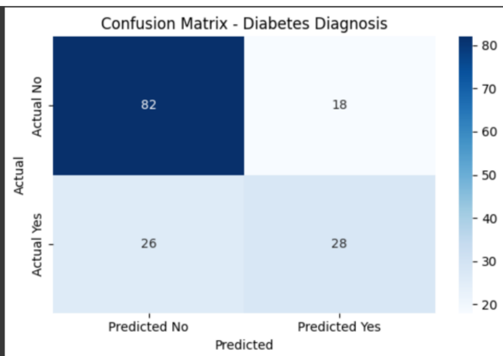
```
print("\nFirst 5 rows of data:")
print(df.head())

# Separate features (independent variables) and target (dependent variable)
X = df.drop("Outcome", axis=1)  # All columns except 'Outcome' are features
y = df["Outcome"]              # 'Outcome' column is the target (0 or 1)

# Split the data into training and testing sets (80% train, 20% test)
# Stratify=y keeps the ratio of classes same in train and test
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)

# Standardize the feature values to have mean = 0 and std = 1
# This helps many machine learning models perform better
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)  # Fit to training data and transform
X_test_scaled = scaler.transform(X_test)        # Use the same transformation on test data

# Create a Logistic Regression model and train it on the training data
model = LogisticRegression(max_iter=200)  # max_iter ensures it runs enough iterations
model.fit(X_train_scaled, y_train)        # Fit the model to the training data

# Predict the outcome for the test set
y_pred = model.predict(X_test_scaled)
```

```
# Generate the confusion matrix to evaluate classification performance

cm = confusion_matrix(y_test, y_pred)


# Labels for interpreting the confusion matrix values

cm_labels = ["True Neg", "False Pos", "False Neg", "True Pos"]

cm_reshaped = cm.reshape(-1)


# Visualize the confusion matrix using a heatmap

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

sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',

            xticklabels=["Predicted No", "Predicted Yes"],

            yticklabels=["Actual No", "Actual Yes"])

plt.title("Confusion Matrix - Diabetes Diagnosis")

plt.xlabel("Predicted")

plt.ylabel("Actual")

plt.tight_layout()

plt.show()


# Print detailed interpretation of the confusion matrix values

print("\nConfusion Matrix Details:")

for label, count in zip(cm_labels, cm_reshaped):

    print(f"{label}: {count}")


# Calculate various evaluation metrics
```

```
accuracy = accuracy_score(y_test, y_pred)       # Overall correctness

precision = precision_score(y_test, y_pred)     # True Positives / (True Positives + False Positives)

recall = recall_score(y_test, y_pred)           # True Positives / (True Positives + False Negatives)

f1 = f1_score(y_test, y_pred)                   # Harmonic mean of precision and recall


# Print evaluation metrics

print("\nModel Evaluation Metrics:")

print(f"Accuracy : {accuracy:.4f}")

print(f"Precision: {precision:.4f}")

print(f"Recall   : {recall:.4f}")

print(f"F1 Score : {f1:.4f}")


# Print a full classification report showing precision, recall, F1-score for each class

print("\nDetailed Classification Report:")

print(classification_report(y_test, y_pred, target_names=["No Diabetes", "Diabetes"]))
```

Some screenshot of the result/output:

```
Dataset Summary:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    int64
 1   Glucose                   768 non-null    int64
 2   BloodPressure             768 non-null    int64
 3   SkinThickness             768 non-null    int64
 4   Insulin                   768 non-null    int64
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64
 8   Outcome                   768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
None

First 5 rows of data:
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0            6      148             72             35        0  33.6
1            1       85             66             29        0  26.6
2            8      183             64              0        0  23.3
3            1       89             66             23       94  28.1
4            0      137             40             35      168  43.1

   DiabetesPedigreeFunction  Age  Outcome
0                     0.627   50        1
1                     0.351   31        0
2                     0.672   32        1
3                     0.167   21        0
4                     2.288   33        1
```




```
Confusion Matrix Details:
True Neg: 82
False Pos: 18
False Neg: 26
True Pos: 28

Model Evaluation Metrics:
Accuracy : 0.7143
Precision: 0.6087
Recall   : 0.5185
F1 Score : 0.5600
```

✓ 1s completed at 2:46 PM

```
Model Evaluation Metrics:
Accuracy : 0.7143
Precision: 0.6087
Recall   : 0.5185
F1 Score : 0.5600

Detailed Classification Report:
              precision    recall  f1-score   support

 No Diabetes       0.76      0.82      0.79       100
    Diabetes       0.61      0.52      0.56        54

    accuracy                           0.71       154
   macro avg       0.68      0.67      0.67       154
weighted avg       0.71      0.71      0.71       154
```

✓ 1s completed at 2:46 PM

References and Credits:

Dataset

- **Pima Indians Diabetes Dataset**
  - Source: [UCI Machine Learning Repository]
  - Originally donated by: National Institute of Diabetes and Digestive and Kidney Diseases
  - Description: This dataset contains diagnostic measurements of female Pima Indian patients aged 21 and above, used to predict the onset of diabetes based on medical attributes.

Libraries and Tools Used

- **Pandas** – Data loading, manipulation, and analysis
  URL://pandas.pydata.org/
- **NumPy** – Numerical operations (indirectly used through other libraries)
  URL: https://numpy.org/
- **Matplotlib** – For basic data visualization
  URL: https://matplotlib.org/
- **Seaborn** – Enhanced data visualization, particularly for the confusion matrix heatmap
  URL: https://seaborn.pydata.org/
- **Scikit-learn (sklearn)** – Machine learning library used for:
  - Model building (LogisticRegression)
  - Data splitting (train_test_split)
  - Feature scaling (StandardScaler)

   - Evaluation metrics (accuracy, precision, recall, F1-score)
     URL: https://scikit-learn.org/

Machine Learning Algorithms

- **Logistic Regression**
  A supervised learning algorithm used for binary classification problems. In this case, it helps predict the presence or absence of diabetes based on multiple medical parameters.

Visualizations

- **Confusion Matrix Heatmap**
  Generated using seaborn.heatmap and matplotlib.pyplot, based on the predictions of the trained model on test data. No external images were used.

Code Implementation/ Data Analysis & Modeling/Visualization & Evaluation:
Yuvraj Singh Taniya

KIET Group of Institutions, Ghaziabad

April, 2025