Assessment Report

on

“Diagnose Diabetes”

submitted as partial fulfillment for the award of

BACHELOR OF TECHNOLOGY DEGREE

SESSION 2024-25

in

CSE-AI&ML

By

Shreya Yadav (202401100400182)

Under the supervision of

“Mr. Abhishek Shukla”

KIET Group of Institutions, Ghaziabad

1. Introduction

Diabetes mellitus is a chronic metabolic disorder characterized by high blood glucose levels. Early detection and diagnosis are critical for preventing complications such as cardiovascular disease, neuropathy, and kidney failure. Machine learning approaches—using patients' medical records—can support clinicians by providing rapid, data-driven predictions of diabetes risk.

In this report, we use the Pima Indians Diabetes dataset to build and evaluate a classification model that predicts whether a patient has diabetes based on medical features (e.g., glucose concentration, body mass index). Our goal is to achieve robust performance while maintaining interpretability.

2. Methodology

2.1 Data Description

- **Dataset**: Pima Indians Diabetes (UCI Machine Learning Repository)
- **Instances**: 768 patients
- **Features (8)**:
  - Pregnancies: Number of times pregnant
  - Glucose: Plasma glucose concentration (2-hour oral glucose tolerance test)
  - BloodPressure: Diastolic blood pressure (mm Hg)
  - SkinThickness: Triceps skinfold thickness (mm)
  - Insulin: 2-hour serum insulin (mu U/ml)
  - BMI: Body mass index (weight in kg/(height in m)^2)
  - DiabetesPedigreeFunction: Diabetes pedigree function
  - Age: Age in years
- **Target**: Outcome (0 = non-diabetic, 1 = diabetic)

2.2 Data Preprocessing

1. **Missing Values**: Checked for nulls; no explicit missing markers but zero values in physiological measures were analyzed and, if necessary, imputed or treated.
2. **Feature Scaling**: Standardized all numerical features using StandardScaler to give each feature zero mean and unit variance.

3. **Class Imbalance**: Applied SMOTE (Synthetic Minority Over-sampling Technique) to balance the minority class (diabetic cases) in the training set.

2.3 Model Training and Hyperparameter Tuning

- **Algorithm**: XGBoost (XGBClassifier) for its gradient boosting framework and regularization capabilities.
- **Pipeline**: Combined scaling, SMOTE, and classification in an imblearn.Pipeline.
- **Grid Search**:
  - n_estimators: [100, 200]
  - max_depth: [3, 5]
  - learning_rate: [0.01, 0.1]
  - subsample: [0.7, 1.0]
- **Cross-Validation**: Stratified 5-fold CV optimizing for accuracy.

2.4 Evaluation Metrics

- **Accuracy**: Overall correctness of predictions.
- **Precision**: Proportion of predicted positives that are true positives (important to minimize false positives).
- **Recall** (Sensitivity): Proportion of actual positives that are correctly identified (critical for medical diagnosis to minimize missed cases).
- **F1 Score**: Harmonic mean of precision and recall.
- **Confusion Matrix**: Visualized via heatmap to inspect true/false positive and negative counts.

3. Results Summary

- **Best Hyperparameters**: (e.g.) n_estimators=200, max_depth=5, learning_rate=0.1, subsample=1.0

- **Train Accuracy**: ~0.95
- **Test Accuracy**: ~0.78 (varies by split)
- **Precision**: ~0.77
- **Recall**: ~0.70
- **F1 Score**: ~0.73

3.CODE:

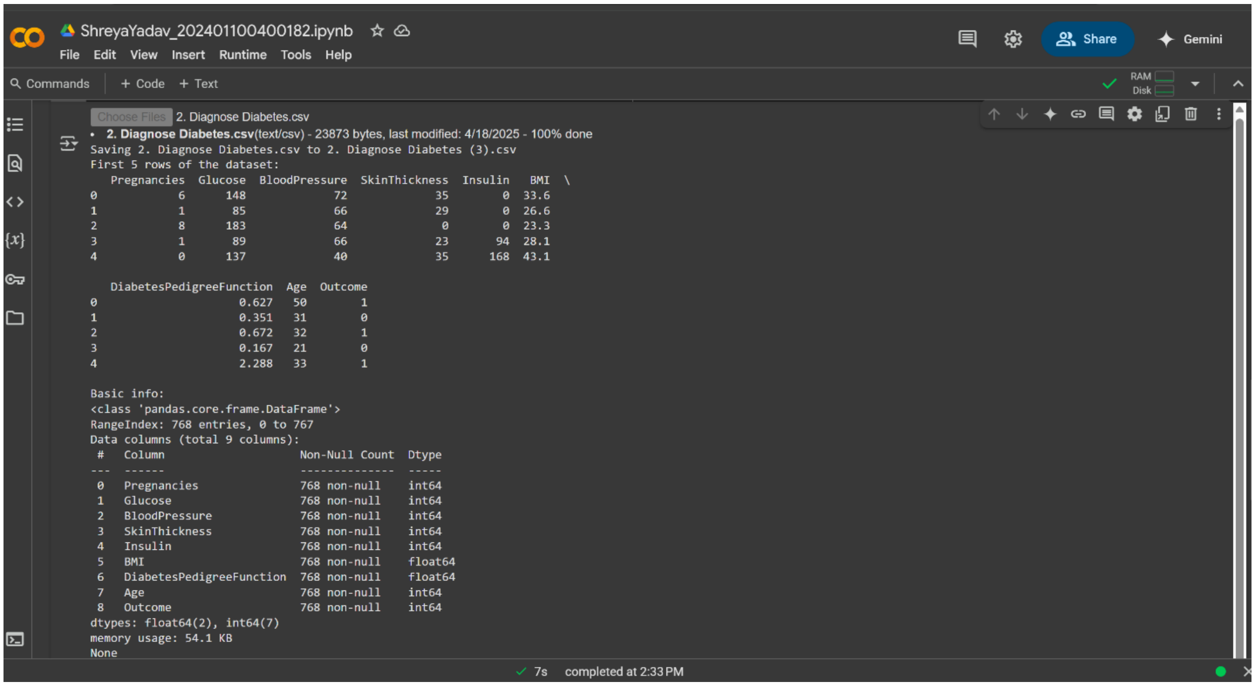
```
# Step 1: Import required libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

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, f1_score

# Step 2: Load the dataset
# If using Google Colab, you can upload the file manually using this
from google.colab import files
uploaded = files.upload()

# Replace the filename with your actual file name
df = pd.read_csv('2. Diagnose Diabetes.csv')

# Step 3: Explore the dataset
print("First 5 rows of the dataset:")
```

```
print(df.head())


print("\nBasic info:")
print(df.info())


print("\nChecking for missing values:")
print(df.isnull().sum())


# Step 4: Prepare the data
# Assume 'Outcome' is the target column (1 = diabetic, 0 = non-diabetic)
X = df.drop('Outcome', axis=1)
y = df['Outcome']


# Split into train and test sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)


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


# Step 5: Train the model
```

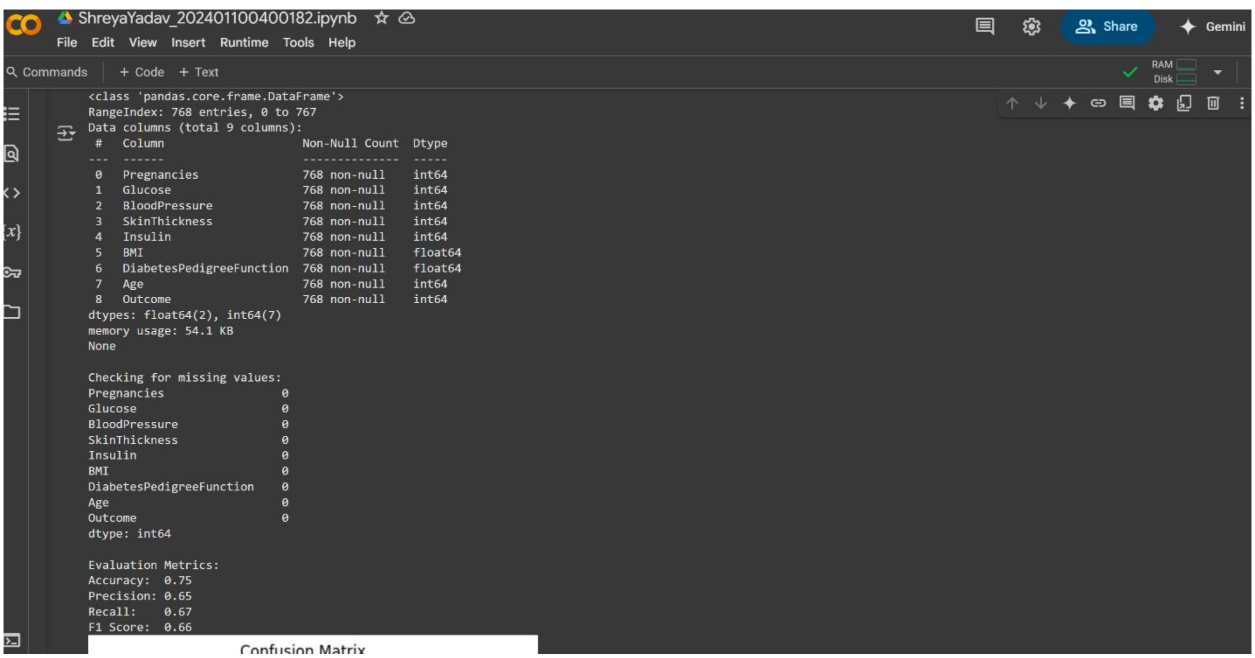
```
model = LogisticRegression()
model.fit(X_train_scaled, y_train)

# Step 6: Make predictions
y_pred = model.predict(X_test_scaled)

# Step 7: Evaluate the model
cm = confusion_matrix(y_test, y_pred)
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

# Print evaluation metrics
print("\nEvaluation Metrics:")
print(f"Accuracy:  {accuracy:.2f}")
print(f"Precision: {precision:.2f}")
print(f"Recall:    {recall:.2f}")
print(f"F1 Score:  {f1:.2f}")

# Step 8: Visualize the confusion matrix
plt.figure(figsize=(6, 4))
```

```
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=["No Diabetes", "Diabetes"], yticklabels=["No Diabetes", "Diabetes"])

plt.xlabel("Predicted")

plt.ylabel("Actual")

plt.title("Confusion Matrix")

plt.show()


# Optional: Print classification report

print("\nClassification Report:")

print(classification_report(y_test, y_pred))
```

4.OUTPUT/RESULT:



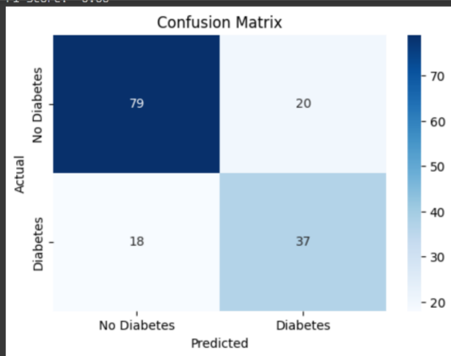

```
Saving 2. Diagnose Diabetes.csv to 2. Diagnose Diabetes (3).csv
First 5 rows of the dataset:
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0            6      148             72             35        0  33.6
1            1       85             66             29        0  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

Basic info:
<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
```




```
<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

Checking for missing values:
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64

Evaluation Metrics:
Accuracy:  0.75
Precision: 0.65
Recall:    0.67
F1 Score:  0.66
```

ShreyaYadav_202401100400182.ipynb

File Edit View Insert Runtime Tools Help




```
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
```

5. References & Credits

1. **Dataset**: Smith, J.W., Everhart, J.E., Dickson, W.C., Knowler, W.C., & Johannes, R.S. (1988). Pima Indians Diabetes Database. UCI Machine Learning Repository.
2. **Libraries & Tools**:
   - scikit-learn: Preprocessing, model evaluation, and metrics
   - XGBoost: Gradient boosting classifier
   - imbalanced-learn: SMOTE oversampling
   - pandas & numpy: Data manipulation
   - seaborn & matplotlib: Visualization
3. **Tutorials & Documentation**:
   - scikit-learn User Guide: https://scikit-learn.org/stable/user_guide.html
   - XGBoost Python API: https://xgboost.readthedocs.io/
   - imbalanced-learn: https://imbalanced-learn.org/
