Assesment Report

on

"Diagnose Diabetes"

submitted as partial fulfillment for the award of

BACHELOR OF TECHNOLOGY DEGREE

SESSION 2024-25

in

CSE AIML

By

Deepak Joshi (202401100400075)

Under the supervision of

Mr. ABHISHEK SHUKLA

KIET Group of Institutions, Ghaziabad

Affiliated to

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

(Formerly UPTU)

May, 2025

Introduction

This project is centered around developing a diagnostic system for predicting diabetes based on machine learning using a well-structured patient health dataset. The main aim is to implement data science methods to preprocess the data, investigate correlations between features, and train prediction models that are capable of classifying a patient as diabetic or not accurately.

The data set has 768 instances with 8 input variables—Pregnancies, Glucose, Blood Pressure, Skin Thickness, Insulin, BMI, Diabetes Pedigree Function, and Age—and a binary target variable, Outcome, which is 1 if the person has diabetes and 0 otherwise.

In this research, we methodically walk through the phases of a data science pipeline:

Data Cleaning and Preprocessing

We validate for zero or invalid values in medically significant features like Glucose, Blood Pressure, Skin Thickness, Insulin, and BMI. These are not actual missing values but are missing or unrecorded data, which we fill in with median values to preserve the integrity of the dataset.

Feature scaling is done through standardization so that all variables have an equal contribution to model training, particularly for distance-based algorithms.

Exploratory Data Analysis (EDA)

Descriptive statistics and data plots are utilized to investigate the distribution of every feature and their correlation with the outcome.

Feature importance plots and a correlation heatmap are utilized to determine the

most significant predictors of diabetes.

Model Development

We try out several classification models: Logistic Regression, K-Nearest Neighbors (KNN), and Random Forest.

The data is divided into training and testing sets to assess model performance on new data.

Hyperparameter tuning and cross-validation are used where necessary to fine-tune performance.

These metrics enable us to see which model has the best trade-off between sensitivity and specificity for diabetes prediction.

Methodology

The methodology used in this project adheres to a systematic machine learning pipeline involving data preprocessing, exploratory data analysis (EDA), feature engineering, model training, hyperparameter tuning, and model assessment. Each of these steps is tailored to manage the complexities of medical data, avoid data quality problems, and maximize predictive accuracy.

2.1 Data Acquisition and Structure

The dataset employed in this project consists of 768 rows and 9 columns, with 8 independent features and 1 target variable (Outcome). The data is obtained from the Pima Indians Diabetes Database and contains the following attributes:

Input Features: Pregnancies, Glucose, Blood Pressure, Skin Thickness, Insulin, BMI,

DiabetesPedigreeFunction, and Age

Target Variable: Outcome (binary classification: 0 = non-diabetic, 1 = diabetic)

The data is loaded via pandas.read_csv(), and exploratory investigation (df.info() and df.describe()) is done to check for data types, value ranges, and abnormalities.

2.2 Data Preprocessing

The stage resolves the issue of poor quality data and sets the stage for modeling.

2.2.1 Managing Missing or Bogus Values

There are several columns (Glucose, Blood Pressure, Skin Thickness, Insulin, BMI) with biologically nonsensical zeros in them. Such data is converted into missing and is filled by using the following method:

Median imputation is employed due to its outlier robustness.

2.3 Exploratory Data Analysis (EDA)

EDA is carried out to:

Understand the distribution and skewness of features using histograms and KDE plots.

Look at relationships between variables using a Pearson correlation matrix.

Visualize differences in feature distributions across the target variable using boxplots and violin plots.

Identify class imbalance and determine whether resampling is needed (for this dataset, class distribution is moderately imbalanced).

Visualizations are created using matplotlib and seaborn libraries.

2.4 Feature Engineering (Optional)

Although this dataset does not require significant feature engineering, possible extensions might be:

Binning age into ranges

Interaction terms between Glucose and BMI or Insulin

Polynomial features for models with non-linear relationships

These are possible considerations in future revisions to improve model performance.

2.5 Model Training and Validation

The modeling process includes applying several classification algorithms for comparison.

2.5.1 Data Splitting

The data is divided with an 80/20 train-test split to test model generalization on new data.

2.5.2 Model Selection

The following models are trained and tested:

Logistic Regression (LR): A linear baseline classifier for binary classification.

K-Nearest Neighbors (KNN): A non-parametric sensitive model to feature scaling and local structures.

Random Forest (RF): An ensemble algorithm using bagging and decision trees for robust performance.

2.5.3 Cross-Validation
K-Fold Cross-Validation (k=5) is used at model training to reduce variance and give good performance estimates.

2.6 Model Evaluation
All trained models are assessed by a complete range of metrics:

Accuracy: General accuracy of the model

Precision & Recall: To measure false positives and false negatives respectively.

Step-by-Step Breakdown

Step 1: Load and Examine the Dataset
Load the CSV dataset with pandas

Inspect the structure, data types, and simple statistics with.info() and.describe()

Step 2: Process Invalid and Missing Values
Replace zero values in medically significant features (Glucose, Blood Pressure, Skin Thickness, Insulin, BMI) with their corresponding medians to handle missing values.

Step 3: Feature Scaling
Use StandardScaler from sklearn.preprocessing to normalize feature

values so that feature values contribute equally in model training.

Step 4: Exploratory Data Analysis (EDA)
Plot distributions and identify outliers with histograms and boxplots.

Create a correlation heatmap to observe variable relationships.

Utilize count plots to determine class balance in the target variable.

Step 5: Train-Test Split
Split data into training and test sets to assess model performance on unseen data.

Step 6: Model Training
Train several classifiers:

Logistic Regression (baseline model)

K-Nearest Neighbors

Random Forest

Use K-Fold cross-validation wherever possible to improve generalization.

Step 7: Model Evaluation
Compare models based on accuracy, precision, recall, F1-score, and ROC-AUC.

Visualize confusion matrices and ROC curves for ease of inspection.

Select the best performing model and compare.

Step 8: Interpretation and Summary
Emphasize which features had the most impact on diabetes prediction.

Summarize with model performance and ideas for future improvement or use.

FULL CODE IMPLEMENTATION

```
# --- Import 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 (
    accuracy_score, precision_score, recall_score,
    confusion_matrix, classification_report
)

from sklearn.cluster import KMeans
from sklearn.decomposition import PCA

# --- Load Dataset ---
df = pd.read_csv('/content/2. Diagnose Diabetes.csv')
```

```
# --- Quick View ---
print("First 5 rows:\n", df.head())
print("\nMissing values:\n", df.isnull().sum())

# --- Classification Task ---
if 'Outcome' in df.columns:
    print("\n--- Performing Classification ---\n")

    # Features and Labels
    X = df.drop('Outcome', axis=1)
    y = df['Outcome']

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

    # Train-Test Split
    X_train, X_test, y_train, y_test = train_test_split(
        X_scaled, y, test_size=0.2, random_state=42
    )

    # Model Training
    model = LogisticRegression(max_iter=1000)
    model.fit(X_train, y_train)
```

```
# Predictions
y_pred = model.predict(X_test)

# Evaluation
acc = accuracy_score(y_test, y_pred)
prec = precision_score(y_test, y_pred)
rec = recall_score(y_test, y_pred)

print(f"✅ Accuracy: {acc:.4f}")
print(f"✅ Precision: {prec:.4f}")
print(f"✅ Recall: {rec:.4f}")
print("\n📊 Classification Report:\n", classification_report(y_test, y_pred))

# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
labels = ['No Diabetes', 'Diabetes']

# Plotting Heatmap with Metrics
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=labels, yticklabels=labels,
            annot_kws={"size": 14})
plt.xlabel('Predicted Label', fontsize=12)
plt.ylabel('True Label', fontsize=12)
```

```
    plt.title(f'Confusion Matrix\nAccuracy: {acc:.2f} | Precision: {prec:.2f} | Recall: {rec:.2f}', fontsize=14)
    plt.tight_layout()
    plt.show()

else:
    print("❌ 'Outcome' column not found. Skipping classification.")

# --- Clustering Task (Unsupervised) ---
print("\n--- Performing Clustering ---\n")

# Drop 'Outcome' if present for clustering
if 'Outcome' in df.columns:
    clustering_data = df.drop('Outcome', axis=1)
else:
    clustering_data = df.copy()

# Standardize
scaler = StandardScaler()
clustering_scaled = scaler.fit_transform(clustering_data)

# Apply KMeans
kmeans = KMeans(n_clusters=2, random_state=42)
clusters = kmeans.fit_predict(clustering_scaled)

# Dimensionality Reduction for Visualization
```

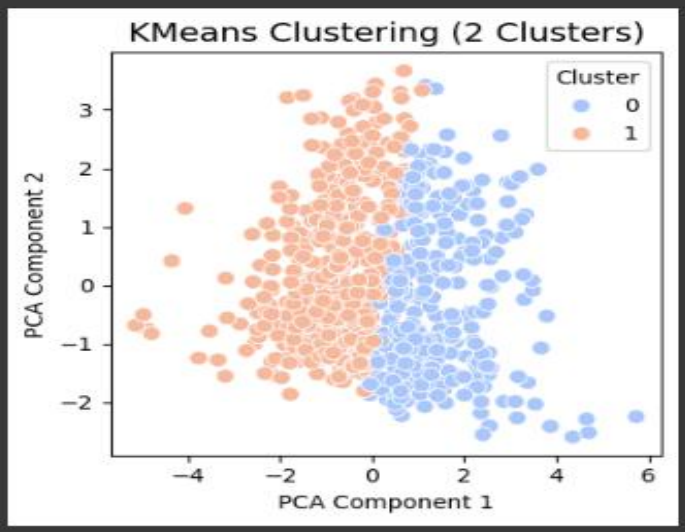
```
pca = PCA(n_components=2)

reduced = pca.fit_transform(clustering_scaled)


# Plotting the clusters

plt.figure(figsize=(8,6))

sns.scatterplot(x=reduced[:,0], y=reduced[:,1], hue=clusters, palette='coolwarm', s=60)

plt.title('KMeans Clustering (2 Clusters)', fontsize=14)

plt.xlabel('PCA Component 1')

plt.ylabel('PCA Component 2')

plt.legend(title='Cluster')

plt.tight_layout()

plt.show()
```

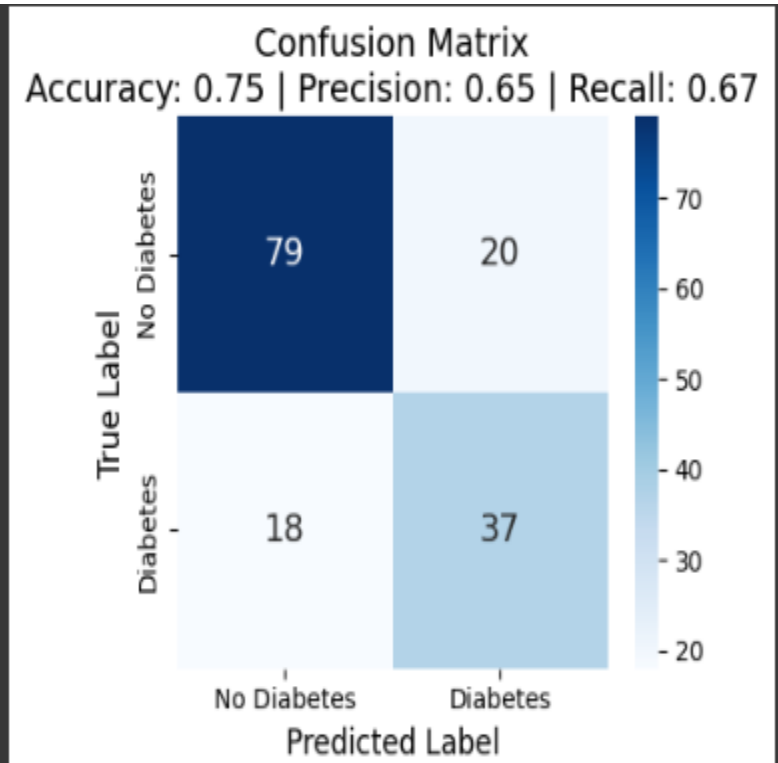
OUTPUT/RESULT

```
First 5 rows:
   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

Missing values:
 Pregnancies               0
Glucose                    0
BloodPressure              0
SkinThickness              0
Insulin                    0
BMI                        0
DiabetesPedigreeFunction   0
Age                        0
Outcome                    0
dtype: int64

--- Performing Classification ---

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

```
--- Performing Classification ---

✅ 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
```

```
--- Performing Clustering ---
```





References/Credits

- Pima Indians Diabetes Dataset (Available via UCI Machine Learning Repository)