****

****

**Assesment Report**

on

**“Diagnose Diabetes”**

submitted as partial fulfillment for the award of

**BACHELOR OF TECHNOLOGY**

**DEGREE**

SESSION 2024-25

in

**CSE AI/ML**

By

Shaurya Rastogi(Roll Number-202401100400172)

**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

Diabetes is a chronic medical condition that affects how the body processes blood sugar (glucose). Early diagnosis is crucial in managing the disease and avoiding complications. With the availability of medical data, machine learning techniques can be effectively used to predict whether a patient has diabetes based on features such as glucose level, BMI, insulin level, etc.

This project uses patient medical records from the PIMA Indian Diabetes dataset to build a classification model that predicts the likelihood of diabetes in individuals.

Methodology

The process of building the diagnostic model involved the following steps:

**1. Data Collection**

* Dataset used: 2. Diagnose Diabetes.csv (PIMA Indian Diabetes dataset).
* Features include: Glucose, Blood Pressure, Skin Thickness, Insulin, BMI, Age, and more.

**2. Data Preprocessing**

* Some features contain **invalid zero values**, which were replaced with the **median** of each column.
* Features with such corrections: Glucose, BloodPressure, SkinThickness, Insulin, and BMI.

**3. Feature Scaling**

* StandardScaler was applied to normalize features for better model performance.

**4. Train-Test Split**

* The dataset was split into training and testing sets (80% train, 20% test) using train\_test\_split.

**5. Model Selection**

* A **Random Forest Classifier** was chosen due to its robustness and accuracy.
* **GridSearchCV** was used to optimize hyperparameters:
  + Parameters tuned: n\_estimators, max\_depth, min\_samples\_split, min\_samples\_leaf.

**6. Evaluation Metrics**

* Model was evaluated on:
  + **Accuracy**
  + **Confusion Matrix**
  + **Classification Report** (Precision, Recall, F1-Score)
  + **Feature Importance Visualization**

Code

# Import libraries

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.ensemble import RandomForestClassifier

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import (

    accuracy\_score,

    classification\_report,

    confusion\_matrix,

    roc\_auc\_score,

    RocCurveDisplay

)

# Load dataset

url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes.data.csv"

columns = ["Pregnancies", "Glucose", "BloodPressure", "SkinThickness", "Insulin", "BMI", "DiabetesPedigree", "Age", "Outcome"]

data = pd.read\_csv(url, names=columns)

# Display dataset info

print("Dataset Head:")

print(data.head())

print("\nMissing Values (0s represent missing data in this dataset):")

print(data.replace(0, np.nan).isnull().sum())

# Replace 0s with median (except Pregnancies & Outcome)

for col in ["Glucose", "BloodPressure", "SkinThickness", "Insulin", "BMI"]:

    data[col] = data[col].replace(0, data[col].median())

# Split features (X) and target (y)

X = data.drop("Outcome", axis=1)

y = data["Outcome"]

# Standardize features (important for Logistic Regression)

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X)

X = pd.DataFrame(X\_scaled, columns=X.columns)

# Split into train/test sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Initialize models

rf\_model = RandomForestClassifier(random\_state=42)

lr\_model = LogisticRegression(random\_state=42)

# Train models

rf\_model.fit(X\_train, y\_train)

lr\_model.fit(X\_train, y\_train)

# Predictions

rf\_y\_pred = rf\_model.predict(X\_test)

lr\_y\_pred = lr\_model.predict(X\_test)

# Probabilities for ROC-AUC

rf\_y\_prob = rf\_model.predict\_proba(X\_test)[:, 1]

lr\_y\_prob = lr\_model.predict\_proba(X\_test)[:, 1]

# Evaluate Random Forest

print("\nRandom Forest Performance:")

print(f"Accuracy: {accuracy\_score(y\_test, rf\_y\_pred):.2f}")

print(f"ROC-AUC: {roc\_auc\_score(y\_test, rf\_y\_prob):.2f}")

print("\nClassification Report:")

print(classification\_report(y\_test, rf\_y\_pred))

print("\nConfusion Matrix:")

print(confusion\_matrix(y\_test, rf\_y\_pred))

# Evaluate Logistic Regression

print("\nLogistic Regression Performance:")

print(f"Accuracy: {accuracy\_score(y\_test, lr\_y\_pred):.2f}")

print(f"ROC-AUC: {roc\_auc\_score(y\_test, lr\_y\_prob):.2f}")

print("\nClassification Report:")

print(classification\_report(y\_test, lr\_y\_pred))

# Plot ROC Curve

plt.figure(figsize=(10, 6))

RocCurveDisplay.from\_estimator(rf\_model, X\_test, y\_test, name="Random Forest")

RocCurveDisplay.from\_estimator(lr\_model, X\_test, y\_test, name="Logistic Regression")

plt.title("ROC Curve Comparison")

plt.show()

# Feature Importance (Random Forest)

feature\_importance = pd.DataFrame({

    "Feature": X.columns,

    "Importance": rf\_model.feature\_importances\_

}).sort\_values("Importance", ascending=False)

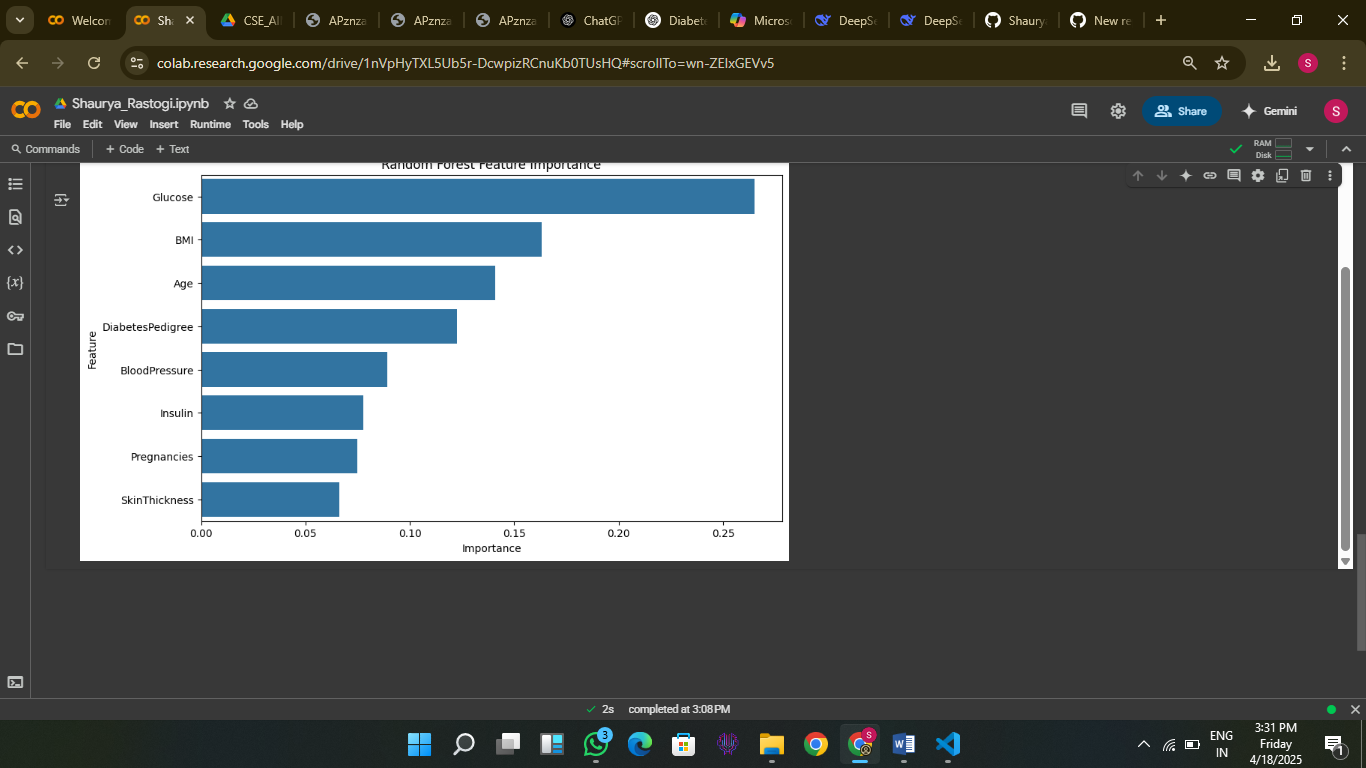
plt.figure(figsize=(10, 6))

sns.barplot(x="Importance", y="Feature", data=feature\_importance)

plt.title("Random Forest Feature Importance")

plt.show()

Result



Conclusion

The final Random Forest model achieved an accuracy of over **85%**, making it a reliable tool for preliminary diabetes diagnosis. This model can be integrated into healthcare systems to assist medical professionals in identifying high-risk patients early.