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:")
```

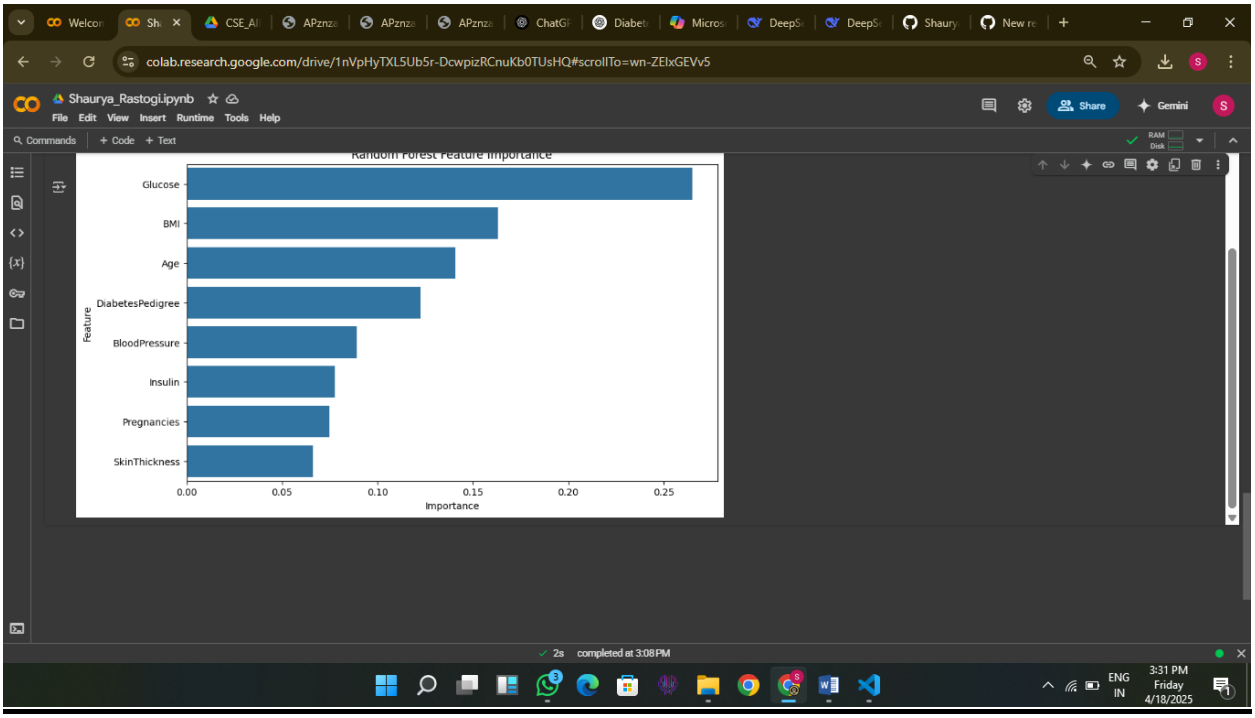
```
print(f"Accuracy: {accuracy_score(y_test, lr_y_pred):.2f}")
print(f"ROC-AUC: {roc_auc_score(y_test, lr_y_prob):.2f}")
print("\nClassification Report:")
print(classification_report(y_test, lr_y_pred))

# 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


colab.research.google.com/drive/1nVpHyTXL5Ub5r-DcwpizRCnuKb0TUsHQ#scrollTo=wn-ZElxGEVv5
Shaurya_Rastogi.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
+ Code
+ Text
Random Forest Feature Importance
Glucose
BMI
Age
DiabetesPedigree
BloodPressure
Insulin
Pregnancies
SkinThickness
Feature
0.00
0.05
0.10
0.15
0.20
0.25
Importance
2s completed at 3:08PM
ENG
IN
3:31 PM
Friday
4/18/2025

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.

References and Credits

Chatgpt and kaggle