Diabetes Analytics Project Report

Ajmal M S

August 22, 2025 (Updated)

Abstract

This project analyzes a diabetes dataset to predict the patient likelihood of developing diabetes using various machine learning models. The report covers data preprocessing, exploratory data analysis (EDA), model training and evaluation, feature importance analysis, and interactive prediction.

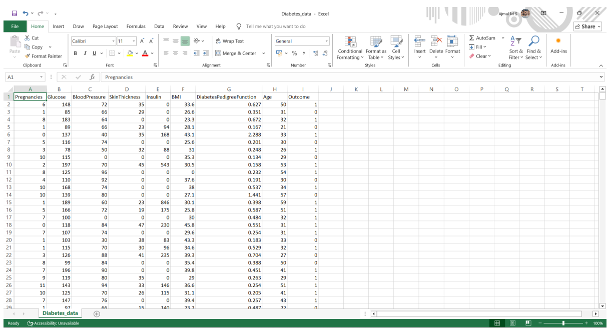
Chapter 1

Introduction

Diabetes is a critical health condition that needs early diagnosis and management. This project leverages a medical dataset containing health parameters related to diabetes to build predictive models aiming to help classify patients as diabetic or non-diabetic.

1.1 Dataset Description

Dataset is sourced from an open-source collection from the internet. The dataset comprises 768 patient records and 9 attributes, including pregnancy status, glucose level, blood pressure, BMI, and an outcome indicating diabetes diagnosis. Figure 1.1 gives a glimpse of the data set used.



Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
6	148	72	35	0	33.6	0.627	50	1
1	85	66	29	0	26.6	0.351	31	0
8	183	64	0	0	23.3	0.672	32	1
1	89	66	23	94	28.1	0.167	21	0
0	137	40	35	168	43.1	2.288	33	1
5	116	74	0	0	25.6	0.201	30	0
3	78	50	32	88	31	0.248	26	1
10	115	0	0	0	35.3	0.134	29	0
2	197	70	45	543	30.5	0.158	53	1
8	125	96	0	0	0	0.232	54	1
4	110	92	0	0	37.6	0.191	30	0
10	168	74	0	0	38	0.537	34	1
10	139	80	0	0	27.1	1.441	57	0
1	189	60	23	846	30.1	0.398	59	1
5	166	72	19	175	25.8	0.587	51	1
7	100	0	0	0	30	0.484	32	1
0	118	84	47	230	45.8	0.551	31	1
7	107	74	0	0	29.6	0.254	31	1
1	103	30	38	83	43.3	0.183	33	0
1	115	70	30	96	34.6	0.529	32	1
3	126	88	41	235	39.3	0.704	27	0
8	99	84	0	0	35.4	0.388	50	0
7	196	90	0	0	39.8	0.451	41	1
9	119	80	35	0	29	0.263	29	1
11	143	94	33	146	36.6	0.254	51	1
10	125	70	26	115	31.1	0.205	41	1
7	147	76	0	0	39.4	0.257	43	1
1	97	66	15	140	23.2	0.487	22	0

Figure 1.1: Caption

Column	**Data Type**
Pregnancies	Integer
Glucose	Integer
BloodPressure	Integer
SkinThickness	Integer
Insulin	Integer
BMI	Float
DiabetesPedigreeFunction	Float
Age	Integer
Outcome	Integer

Table 1.1: Dataset columns and their data types

Chapter 2

Exploratory Data Analysis

2.1 Null Value Analysis

Initial inspection revealed that some columns contain zeroes representing missing values. These were replaced by the mean or median values to maintain data integrity.

Column	**Data Type**
Feature	**Missing Values**
Glucose	5
BloodPressure	35
SkinThickness	227
Insulin	374
BMI	11

Table 2.1: Count of missing values (represented as zeroes) before imputation

2.2 Distribution of Features

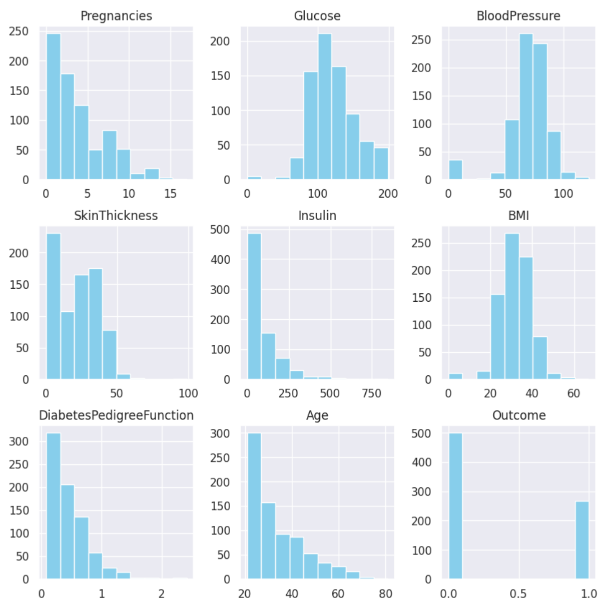


Figure 2.1: Distribution of Values Before Cleaning Null Values

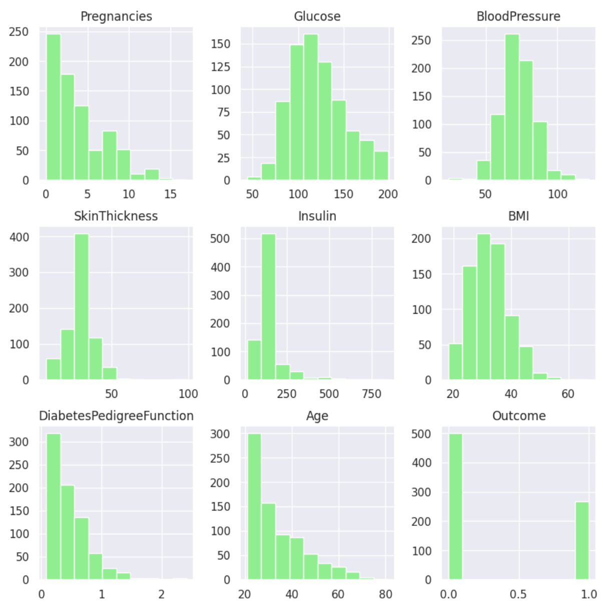


Figure 2.2: Distribution of Values After Cleaning Null Values

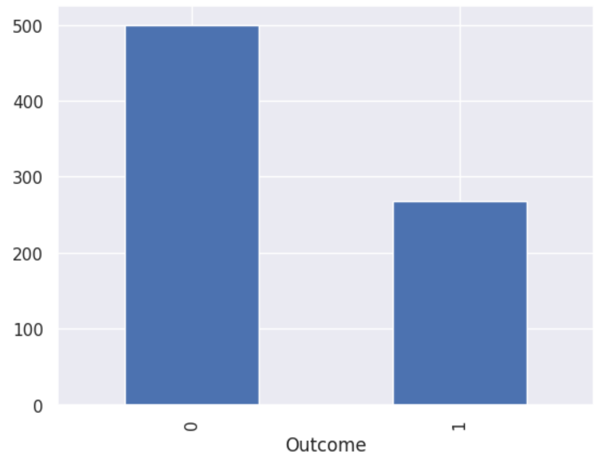


Figure 2.3: Bar plot showing distribution of diabetic (1) and non-diabetic (0) patients

2.3 Correlation Heatmap

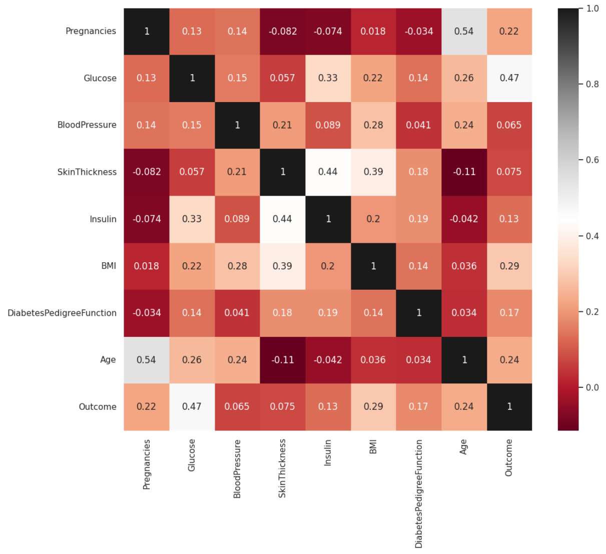


Figure 2.4: Heatmap depicting correlations between attributes

Chapter 3

Data Preprocessing

Data scaling was applied using StandardScaler to bring attributes into comparable ranges for the models.

3.1 Modeling and Evaluation

3.1.1 Train-Test Split

The dataset was split into 67% training and 33% testing sets to evaluate model generalization.

3.1.2 Models Used

Four machine learning models were trained and evaluated:

- Random Forest Classifier
- Decision Tree Classifier
- XGBoost Classifier
- Support Vector Machine (SVM)

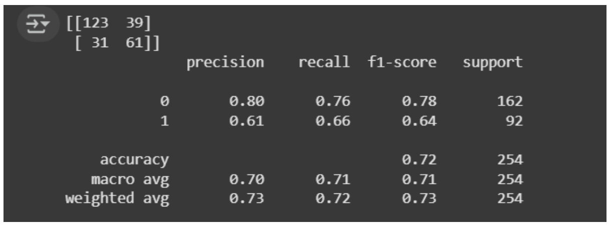
3.1.3 Model Performance

The table below summarizes the accuracy scores for each model:

Model	**Accuracy Score**
Random Forest	0.76
Decision Tree	0.71
XGBoost	0.72
SVM	0.75

Table 3.1: Model accuracy on test data

3.1.4 Confusion Matrices and Classification Reports



```
[[123  39]
 [ 31  61]]
              precision    recall  f1-score   support

           0       0.80      0.76      0.78       162
           1       0.61      0.66      0.64        92

    accuracy                           0.72       254
   macro avg       0.70      0.71      0.71       254
weighted avg       0.73      0.72      0.73       254
```

Figure 3.1: Caption

3.2 Feature Importance

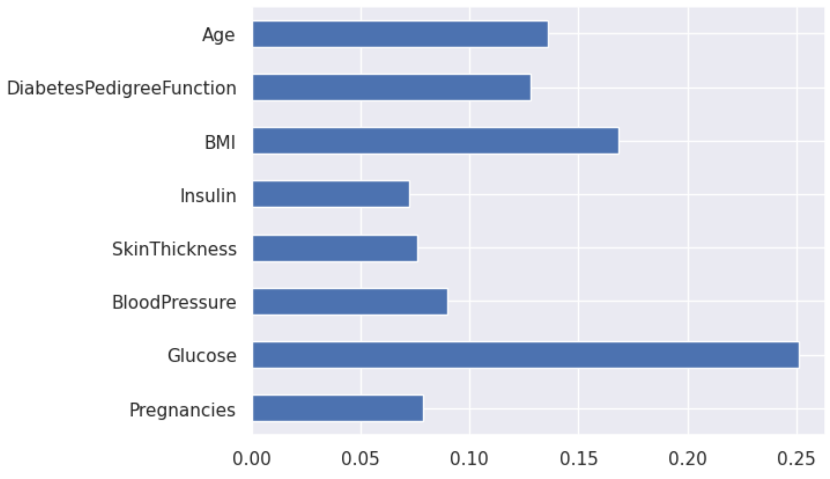
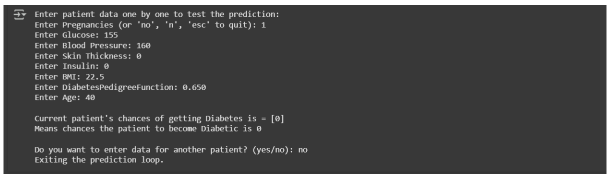


Figure 3.2: Feature importance from the Random Forest Classifier

3.3 Interactive Prediction

The trained Random Forest model was saved and implemented in a command-line interface to input patient data for real-time diabetes prediction.

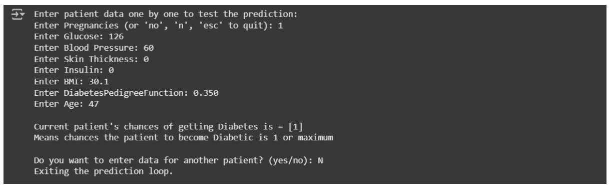


```
Enter patient data one by one to test the prediction:
Enter Pregnancies (or 'no', 'n', 'esc' to quit): 1
Enter Glucose: 155
Enter Blood Pressure: 160
Enter Skin Thickness: 0
Enter Insulin: 0
Enter BMI: 22.5
Enter DiabetesPedigreeFunction: 0.650
Enter Age: 40

Current patient's chances of getting Diabetes is = [0]
Means chances the patient to become Diabetic is 0

Do you want to enter data for another patient? (yes/no): no
Exiting the prediction loop.
```

Figure 3.3: Prediction of a Patient’s Diabetes as No



```
Enter patient data one by one to test the prediction:
Enter Pregnancies (or 'no', 'n', 'esc' to quit): 1
Enter Glucose: 126
Enter Blood Pressure: 60
Enter Skin Thickness: 0
Enter Insulin: 0
Enter BMI: 30.1
Enter DiabetesPedigreeFunction: 0.350
Enter Age: 47

Current patient's chances of getting Diabetes is = [1]
Means chances the patient to become Diabetic is 1 or maximum

Do you want to enter data for another patient? (yes/no): N
Exiting the prediction loop.
```

Figure 3.4: Prediction of a Patient’s Diabetes as Yes

Sample output after entering patient features:

```
Current patient’s chances of getting Diabetes is = [1]
Means chances the patient to become Diabetic is 1 or maximum
```

Chapter 4

Conclusion

Random Forest achieved the highest accuracy of 76%. Data preprocessing and feature importance analysis highlighted critical indicators such as glucose and BMI. Future improvements may explore deeper hyperparameter tuning, ensemble methods, and additional features.

Appendix A

Full Code Listing

Python Version: 3.13.7, Execution environment - Google Colab

```
from google.colab import drive
drive.mount('/content/drive')

# Importing Diabetes data set and necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
# %matplotlib inline

# Load dataset
df = pd.read_csv('/content/drive/My Drive/Diabanalytics/
   Diabetes_data.csv')

sns.set()

#showing first 5 attributes of data
df.head()

# Exploring Data and Attribute Association
df.info()

# Check the column names
df.columns

# Check null values
```

```
df.isnull()

# Null values not means0, but value inputs like NIL, NAN are also
   counted. To displat those;
df_copy = df.copy(deep = True)
df_copy[['Glucose','BloodPressure','SkinThickness','Insulin','BMI']]
    = df_copy[['Glucose','BloodPressure','SkinThickness','Insulin','
   BMI']].replace(0,np.nan)

print(df_copy.isnull().sum())

#plotting a histogram to infer details about value pattern
h = df.hist(figsize=(10,10), color='skyblue')

# Replacing null value with mean values of each column respectively
df_copy['Glucose'].fillna(df_copy['Glucose'].mean(), inplace = True)
df_copy['BloodPressure'].fillna(df_copy['BloodPressure'].mean(),
   inplace = True)
df_copy['SkinThickness'].fillna(df_copy['SkinThickness'].median(),
   inplace = True)
df_copy['Insulin'].fillna(df_copy['Insulin'].median(), inplace =
   True)
df_copy['BMI'].fillna(df_copy['BMI'].median(), inplace = True)

# Plotting a histogram after removing null values
h = df_copy.hist(figsize=(10,10),color="lightgreen")

# Representing the outcome column balance (0 is non diabetic and 1
   is diabetic patients)
color_wheel = {1: " ", 2: " "}
colors = df["Outcome"].map(lambda x: color_wheel.get(x + 1))
print(df.Outcome.value_counts())
p=df.Outcome.value_counts().plot(kind="bar")

# As per above diagram, the diabetic patients are half as count of
   total patients. Next step is checking the correlation.
plt.figure(figsize=(12,10))
# calling seaborn showcase heatmap
p = sns.heatmap(df.corr(), annot=True,cmap ='RdGy')
```

```
#From the heatmap, we can associate values, for example, insulin in
   correlated with glucose with value of 0.33, meaning glucose can
   effect insulin in body.

#Next step is scaling the data scaling, done to generalize the
   differences in data points.

#Importing Scaling and pre-models required for ML process.

from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix
from sklearn import metrics
from sklearn.metrics import classification_report

sc_X = StandardScaler()
X =  pd.DataFrame(sc_X.fit_transform(df_copy.drop(["Outcome"],axis =
    1),), columns=['Pregnancies',
'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', '
   DiabetesPedigreeFunction', 'Age'])

X.head()

# Model Building
X = df.drop('Outcome', axis=1)
y = df['Outcome']

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X,y, test_size
   =0.33,
                                                    random_state=7)

# Model 1 - Random Forest
from sklearn.ensemble import RandomForestClassifier

rfc = RandomForestClassifier(n_estimators=200)
rfc.fit(X_train, y_train)

#Checking the accuracy of model
```

```python
rfc_train = rfc.predict(X_train)
from sklearn import metrics

print("Accuracy_Score =", format(metrics.accuracy_score(y_train,
   rfc_train)))

#Getting Accuracy score of random forest
from sklearn import metrics

predictions = rfc.predict(X_test)
print("Accuracy_Score =", format(metrics.accuracy_score(y_test,
   predictions)))

#Classification report and Confusion matrix of Random Forest
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test,predictions))
print(classification_report(y_test,predictions))

# Model 2- DecisionTree
from sklearn.tree import DecisionTreeClassifier

dtree = DecisionTreeClassifier()
dtree.fit(X_train, y_train)

# Accuracy score of Decision Tree
from sklearn import metrics

predictions = dtree.predict(X_test)
print("Accuracy Score =", format(metrics.accuracy_score(y_test,
   predictions)))

#Classification report and Confusion matrix of Decision Tree
from sklearn.metrics import classification_report, confusion_matrix

print(confusion_matrix(y_test, predictions))
print(classification_report(y_test,predictions))

# Model 3- XgBoost classifier
!pip install xgboost
from xgboost import XGBClassifier
```

```
xgb_model = XGBClassifier(gamma=0)
xgb_model.fit(X_train, y_train)

# Accuracy score of XgBoost classifier
from sklearn import metrics

xgb_pred = xgb_model.predict(X_test)
print("Accuracy Score =", format(metrics.accuracy_score(y_test,
   xgb_pred)))

#Classification report and Confusion matrix of XgBoost classifier
from sklearn.metrics import classification_report, confusion_matrix

print(confusion_matrix(y_test, xgb_pred))
print(classification_report(y_test,xgb_pred))

# Model 4- support Vector Machine (SVM)
from sklearn.svm import SVC

svc_model = SVC()
svc_model.fit(X_train, y_train)

svc_pred = svc_model.predict(X_test)

# Accuracy score of SVM
from sklearn import metrics

print("Accuracy Score =", format(metrics.accuracy_score(y_test,
   svc_pred)))

from sklearn.metrics import classification_report, confusion_matrix

print(confusion_matrix(y_test, svc_pred))
print(classification_report(y_test,svc_pred))

#finding feature importance to analyze which column feature holds
   most weightage in making decisions.
(pd.Series(rfc.feature_importances_, index=X.columns).plot(kind='
   barh'))

# 4) Saving Model
```

```
import pickle
saved_model = pickle.dumps(rfc)
#loading the saved model
rfc_from_pickle = pickle.loads(saved_model)

# use this to make predictions
rfc_from_pickle.predict(X_test)

#Begin prediction

df.head()

df.tail()

print("Enter patient data one by one to test the prediction:")

while True:
    quit_flag = False

    # Pregnancies input loop
    while True:
        Pregnancies_input = input("Enter Pregnancies (or 'no', 'n',
           'esc' to quit): ").strip().lower()
        if Pregnancies_input in ['no', 'n', 'esc']:
            quit_flag = True
            break
        elif Pregnancies_input == '':
            print("Input cannot be blank. Please enter a value.")
            continue
        else:
            try:
                Pregnancies = int(Pregnancies_input)
                break
            except ValueError:
                print("Invalid input. Please enter a numeric value."
                    )

    if quit_flag:
        print("Exiting the prediction loop.")
        break
```

```
def get_numeric_input(prompt, val_type=float):
    while True:
        user_input = input(prompt).strip().lower()
        if user_input in ['no', 'n', 'esc']:
            return None
        elif user_input == '':
            print("Input cannot be blank. Please enter a value."
                )
            continue
        try:
            return val_type(user_input)
        except ValueError:
            print(f"Invalid input. Please enter a {'number' if
                val_type == float else 'whole number'}.")

glucose = get_numeric_input("Enter Glucose: ", int)
if glucose is None:
    print("Exiting the prediction loop.")
    break
bloodpressure = get_numeric_input("Enter Blood Pressure: ", int)
if bloodpressure is None:
    print("Exiting the prediction loop.")
    break
skinthickness = get_numeric_input("Enter Skin Thickness: ", int)
if skinthickness is None:
    print("Exiting the prediction loop.")
    break
insulin = get_numeric_input("Enter Insulin: ", int)
if insulin is None:
    print("Exiting the prediction loop.")
    break
bmi = get_numeric_input("Enter BMI: ", float)
if bmi is None:
    print("Exiting the prediction loop.")
    break
DiabetesPedigreeFunction = get_numeric_input("Enter
   DiabetesPedigreeFunction: ", float)
if DiabetesPedigreeFunction is None:
    print("Exiting the prediction loop.")
    break
age = get_numeric_input("Enter Age: ", int)
```

```
if age is None:
    print("Exiting the prediction loop.")
    break

print()  # Blank line after inputs

result = rfc.predict([[Pregnancies, glucose, bloodpressure,
   skinthickness, insulin, bmi, DiabetesPedigreeFunction, age]])
print(f"Current patient's chances of getting Diabetes is = {
   result}")

if result[0] == 0:
    print("Means chances the patient to become Diabetic is 0")
elif result == 1:
    print("Means chances the patient to become Diabetic is 1 or
       maximum")

print()  # Blank line before continue prompt

cont = input("Do you want to enter data for another patient? (
   yes/no): ").strip().lower()
if cont in ['no', 'n', 'esc']:
    print("Exiting the prediction loop.")
    break
```