Input:

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import warnings

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn import svm

from sklearn.metrics import classification\_report

from sklearn.metrics import confusion\_matrix

from sklearn.metrics import ConfusionMatrixDisplay

warnings.filterwarnings("ignore", category=UserWarning)

RED = "\033[91m"

GREEN = "\033[92m"

YELLOW = "\033[93m"

BLUE = "\033[94m"

RESET = "\033[0m"

df = pd.read\_csv("/kaggle/input/diabetes-data-set/diabetes.csv")

# DATA CLEANING

print(BLUE + "\nDATA CLEANING" + RESET)

# --- Check for missing values

missing\_values = df.isnull().sum()

print(GREEN + "Missing Values : " + RESET)

print(missing\_values)

# --- Handle missing values

mean\_fill = df.fillna(df.mean())

df.fillna(mean\_fill, inplace=True)

# --- Check for duplicate values

duplicate\_values = df.duplicated().sum()

print(GREEN + "Duplicate Values : " + RESET)

print(duplicate\_values)

# --- Drop duplicate values

df.drop\_duplicates(inplace=True)

# DATA ANALYSIS

print(BLUE + "\nDATA ANALYSIS" + RESET)

# --- Summary Statistics

summary\_stats = df.describe()

print(GREEN + "Summary Statistics : " + RESET)

print(summary\_stats)

# --- Class Distribution

class\_distribution = df["Outcome"].value\_counts()

print(GREEN + "Class Distribution : " + RESET)

print(class\_distribution)

# Support Vector Machine Modelling

# --- Separate features and target variable

print(BLUE + "\nMODELLING" + RESET)

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

y = df["Outcome"]

# --- Splitting the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

X, y, test\_size=0.2, random\_state=42

)

# --- Standardize Features

scaler = StandardScaler()

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

X\_test = scaler.transform(X\_test)

# --- init and train SVM model

model = svm.SVC(kernel="linear")

model.fit(X\_train, y\_train)

# --- Predict on test data

y\_pred = model.predict(X\_test)

# --- Evaluate model performance

accuracy = model.score(X\_test, y\_test)

print(GREEN + "Model Accuracy : " + RESET)

print(accuracy)

# --- Classification Report and Confusion Matrix

print(GREEN + "Classification Report : " + RESET)

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

print(GREEN + "Confusion Matrix : " + RESET)

cm = ConfusionMatrixDisplay.from\_predictions(y\_test, y\_pred)

sns.heatmap(cm.confusion\_matrix, annot=True, cmap="Blues")

plt.show()

print("Displayed")

# DATA VISUALIZATION

print(BLUE + "\nDATA VISUALIZATION" + RESET)

# --- Pair Plot

print(GREEN + "PairPlot : " + RESET)

sns.pairplot(df, hue='Outcome',diag\_kind='kde',palette = "Blues")

plt.title("Pairwise Relationships")

plt.show()

# --- Histogram for age distribution

print(GREEN + "Histogram : " + RESET)

sns.histplot(df["Age"], bins=10, kde=True,palette = "Blues")

plt.xlabel("Age")

plt.ylabel("Count")

plt.title("Age Distribution")

plt.show()

# --- Box plot to visualize glucose levels by outcome

print(GREEN + "BoxPlot : " + RESET)

sns.boxplot(x="Outcome", y="Glucose", data=df,palette = "Blues")

plt.xlabel("Diabetes Outcome (0: No, 1: Yes)")

plt.ylabel("Glucose Level")

plt.title("Glucose Levels by Diabetes Outcome")

plt.show()

# --- Correlation heatmap

print(GREEN + "Correlation Heatmap : " + RESET)

correlation\_matrix = df.corr()

sns.heatmap(correlation\_matrix, annot=True, cmap="Blues")

plt.title("Correlation Heatmap")

plt.show()

# SAVING THE FILE

df.to\_csv("/kaggle/working/cleaned\_diabetes.csv", index=False)

print(BLUE + "\nDATA SAVING" + RESET)

print(GREEN + "Data Cleaned and Saved !" + RESET)

print("\n")

output:

DATA CLEANING

Missing Values :

Pregnancies 0

Glucose 0

BloodPressure 0

SkinThickness 0

Insulin 0

BMI 0

DiabetesPedigreeFunction 0

Age 0

Outcome 0

dtype: int64

Duplicate Values :

0

DATA ANALYSIS

Summary Statistics :

Pregnancies Glucose BloodPressure SkinThickness Insulin \

count 768.000000 768.000000 768.000000 768.000000 768.000000

mean 3.845052 120.894531 69.105469 20.536458 79.799479

std 3.369578 31.972618 19.355807 15.952218 115.244002

min 0.000000 0.000000 0.000000 0.000000 0.000000

25% 1.000000 99.000000 62.000000 0.000000 0.000000

50% 3.000000 117.000000 72.000000 23.000000 30.500000

75% 6.000000 140.250000 80.000000 32.000000 127.250000

max 17.000000 199.000000 122.000000 99.000000 846.000000

BMI DiabetesPedigreeFunction Age Outcome

count 768.000000 768.000000 768.000000 768.000000

mean 31.992578 0.471876 33.240885 0.348958

std 7.884160 0.331329 11.760232 0.476951

min 0.000000 0.078000 21.000000 0.000000

25% 27.300000 0.243750 24.000000 0.000000

50% 32.000000 0.372500 29.000000 0.000000

75% 36.600000 0.626250 41.000000 1.000000

max 67.100000 2.420000 81.000000 1.000000

Class Distribution :

Outcome

0 500

1 268

Name: count, dtype: int64

MODELLING

Model Accuracy :

0.7597402597402597

Classification Report :

precision recall f1-score support

0 0.81 0.82 0.81 99

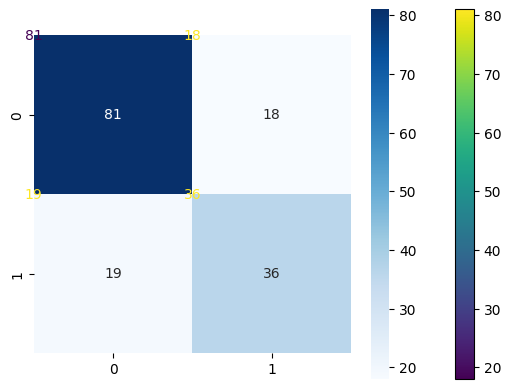
1 0.67 0.65 0.66 55

accuracy 0.76 154

macro avg 0.74 0.74 0.74 154

weighted avg 0.76 0.76 0.76 154

confusion matrix:



Pair plot:

A screenshot of a graph

Description automatically generated

Histogram:

A graph of age distribution

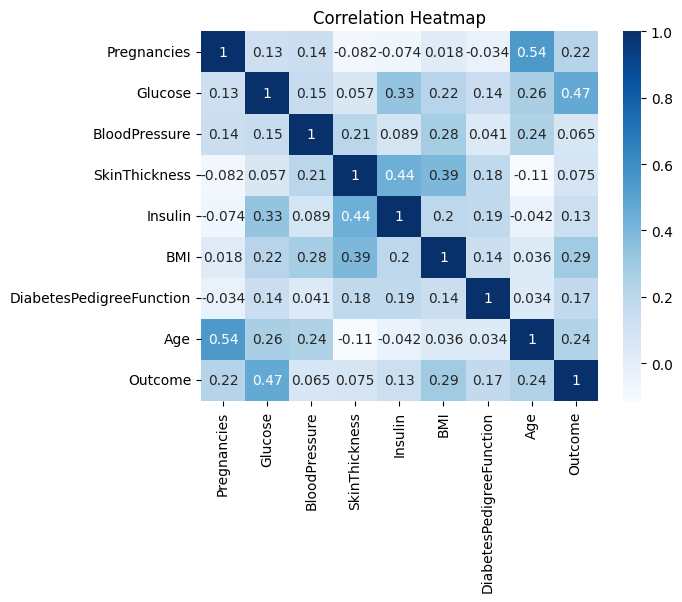
Description automatically generated

Boxplot:

A chart of diabetes and diabetes outcome

Description automatically generated with medium confidence

Correlation heatmap:



Data saved and cleaned