Data

The dataset is from National Institute of Diabetes and Digestive and Kidney Diseases.

Link for the data:

<https://www.kaggle.com/uciml/pima-indians-diabetes-database?select=diabetes.csv>

Description on the data

The data has 9 columns:

* Pregnancies- The number of pregnancies till date(in case of women)
* Glucose- The level of glucose after an overnight fast measured in mg/dL
* Blood Pressure-The blood pressure levels measured in mm Hg
* Skin Thickness- The triceps skin thickness.
* Insulin-Serum insulin measured in mu/mL
* Body Mass Index-Body Mass Index counted by height and weight of the body( height in meters and weight in kilograms)
* Diabetes Pedigree Function
* Age- Age in years
* Outcome- 0 or 1

0-Non-diabetic

1-Diabetic

The data has 768 rows meaning 768 different values of all the 9 columns

Code: (Please go to colab link for complete code with all comments,headers and graphs)

This only contains code

import numpy as np

import pandas as pd

from sklearn.preprocessing import StandardScaler

from sklearn import svm

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

from sklearn.metrics import accuracy\_score

import seaborn as sns

import matplotlib.pyplot as plt

import warnings

warnings.filterwarnings('ignore')

# loading the diabetes dataset to a pandas DataFrame

diabetes\_dataset = pd.read\_csv('/content/diabetes.csv')

diabetes\_dataset.head()

sns.countplot(x = 'Outcome',data = diabetes\_dataset)

sns.barplot(x='Outcome',y='Glucose',data=diabetes\_dataset,palette="Blues")

sns.barplot(x='Outcome',y='Pregnancies',data=diabetes\_dataset,palette="Accent\_r")

sns.barplot(x='Outcome',y='BloodPressure',data=diabetes\_dataset,palette="BrBG")

sns.barplot(x='Outcome',y='Insulin',data=diabetes\_dataset,palette="BuGn")

sns.barplot(x='Outcome',y='BMI',data=diabetes\_dataset,palette="BuPu")

sns.barplot(x='Outcome',y='DiabetesPedigreeFunction',data=diabetes\_dataset,palette="CMRmap")

sns.barplot(x='Outcome',y='Age',data=diabetes\_dataset,palette="PRGn")

cor=diabetes\_dataset.corr()

sns.heatmap(cor, annot = True)

plt.show()

diabetes\_dataset.describe()

P=diabetes\_dataset.drop(columns='Outcome',axis=1)

Q=diabetes\_dataset['Outcome']

print(P)

print(Q)

standardscaler= StandardScaler()

standardscaler.fit(P)

s\_data=standardscaler.transform(P)

print(s\_data)

P=s\_data

print(P)

P\_train, P\_test, Q\_train, Q\_test = train\_test\_split(P,Q, test\_size = 0.2, stratify=Q, random\_state=4)

print(P.shape, P\_train.shape, P\_test.shape)

  train = svm.SVC(kernel='linear')

train.fit(P\_train, Q\_train)

P\_prediction = train.predict(P\_train)

training\_accuracy = accuracy\_score(P\_prediction, Q\_train)

print('The accuracy of test data set is ',test\_accuracy\*100,'%')

testing\_prediction = train.predict(P\_test)

test\_accuracy = accuracy\_score(testing\_prediction, Q\_test)

print('The accuracy of test data set is ',test\_accuracy\*100,'%')

from sklearn.naive\_bayes import GaussianNB

bayes = GaussianNB()

bayes.fit(P\_train, Q\_train)

testing\_prediciton\_nb=bayes.predict(P\_test)

test\_accuracybayes = accuracy\_score(testing\_prediciton\_nb, Q\_test)

from sklearn.ensemble import RandomForestClassifier

randomf = RandomForestClassifier(n\_estimators = 12, criterion = 'entropy', random\_state = 32)

randomf.fit(P\_train, Q\_train)

testing\_prediction\_ranfor = randomf.predict(P\_test)

test\_accuracy\_ranfor = accuracy\_score(testing\_prediction\_ranfor,Q\_test)

print('The accuracy of test data set in SVM ',test\_accuracy\*100,'%')

print('The accuracy of test data set in Bayes algorithm is  ',test\_accuracybayes\*100,'%')

print('The accuracy of test data set in random forest method   is  ',test\_accuracy\_ranfor\*100,'%')

#input or replace values in this array as per the order of the data

#array=[pregnacies,glucose,bloodpressure,skinthickness,insulin,bmi,diabetespdogreefunction,age]

array=[2,120,60,20,120,30,0.2,30]

check=np.asarray(array)

check1=check.reshape(1,-1)

input=standardscaler.transform(check1)

print(input)

#prints the value of the data which is tranformed and standardized as we did earler for higher accuracy

predict\_valuesgiven=train.predict(input)

print(predict\_valuesgiven)

if(predict\_valuesgiven[0]==1):

    print("The person is diabetic")

else:

    print("NO,The person is not diabetic")

diabetes\_dataset.describe()

#input or replace values in this array as per the order of the data

#array=[pregnacies,glucose,bloodpressure,skinthickness,insulin,bmi,diabetespdogreefunction,age]

#array=[3,120,69,20,79,31,0.471,33]

check=np.asarray(array)

check1=check.reshape(1,-1)

input=standardscaler.transform(check1)

print(input)

predict\_valuesgiven=train.predict(input)

print(predict\_valuesgiven)

if(predict\_valuesgiven[0]==1):

    print("The person is diabetic")

else:

    print("NO,the person is not diabetic")

#input or replace values in this array as per the order of the data

#array=[pregnacies,glucose,bloodpressure,skinthickness,insulin,bmi,diabetespdogreefunction,age]

array=[6.000000,140.250000,80.000000,32.000000,127.250000,36.600000,0.626250,41.000000]

#array=[preg,gluc,bp,skin,ins,bmi,dpf,age]

check=np.asarray(array)

check1=check.reshape(1,-1)

input=standardscaler.transform(check1)

print(input)

predict\_valuesgiven=train.predict(input)

print(predict\_valuesgiven)

if(predict\_valuesgiven[0]==1):

    print("The person is diabetic")

else:

    print("NO,the person is not diabetic")

#input or replace values in this array as per the order of the data

#array=[pregnacies,glucose,bloodpressure,skinthickness,insulin,bmi,diabetespdogreefunction,age]

array=[3,140.25000,80.0000,20.5,127.250000,36.6,0.47,41]

check=np.asarray(array)

check1=check.reshape(1,-1)

input=standardscaler.transform(check1)

print(input)

predict\_valuesgiven=train.predict(input)

print(predict\_valuesgiven)

if(predict\_valuesgiven[0]==1):

    print("The person is diabetic")

else:

    print("NO,the person is not diabetic")