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

```python
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="Blue
s")


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_data
set,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 = 'entrop
y', 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_accu
racybayes*100,'%')
print('The accuracy of test data set in random forest method   is  ',te
st_accuracy_ranfor*100,'%')

#input or replace values in this array as per the order of the data
#array=[pregnacies,glucose,bloodpressure,skinthickness,insulin,bmi,diab
etespdogreefunction,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 w
e 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,diab
etespdogreefunction,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,diab
etespdogreefunction,age]

array=[6.000000,140.250000,80.000000,32.000000,127.250000,36.600000,0.6
26250,41.000000]
#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,diab
etespdogreefunction,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")
```