**Coimbatore Institute of Technology**

**Pre-Assessment Test – Curnue**

**Task-01**

**Name : Indhu Sree S**

**Roll no : 1832023**

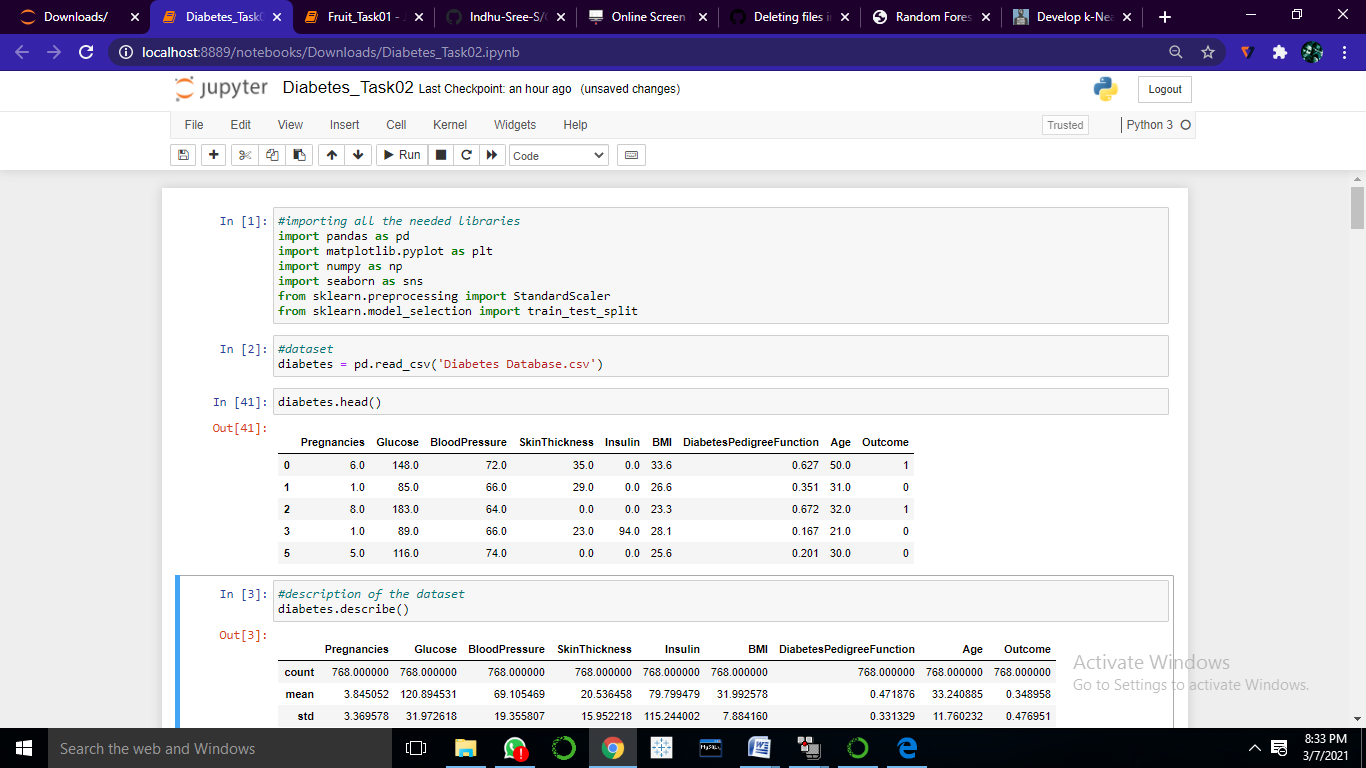
**M.Sc Data Science**

**Ques : SD03Q03**

**Task:**

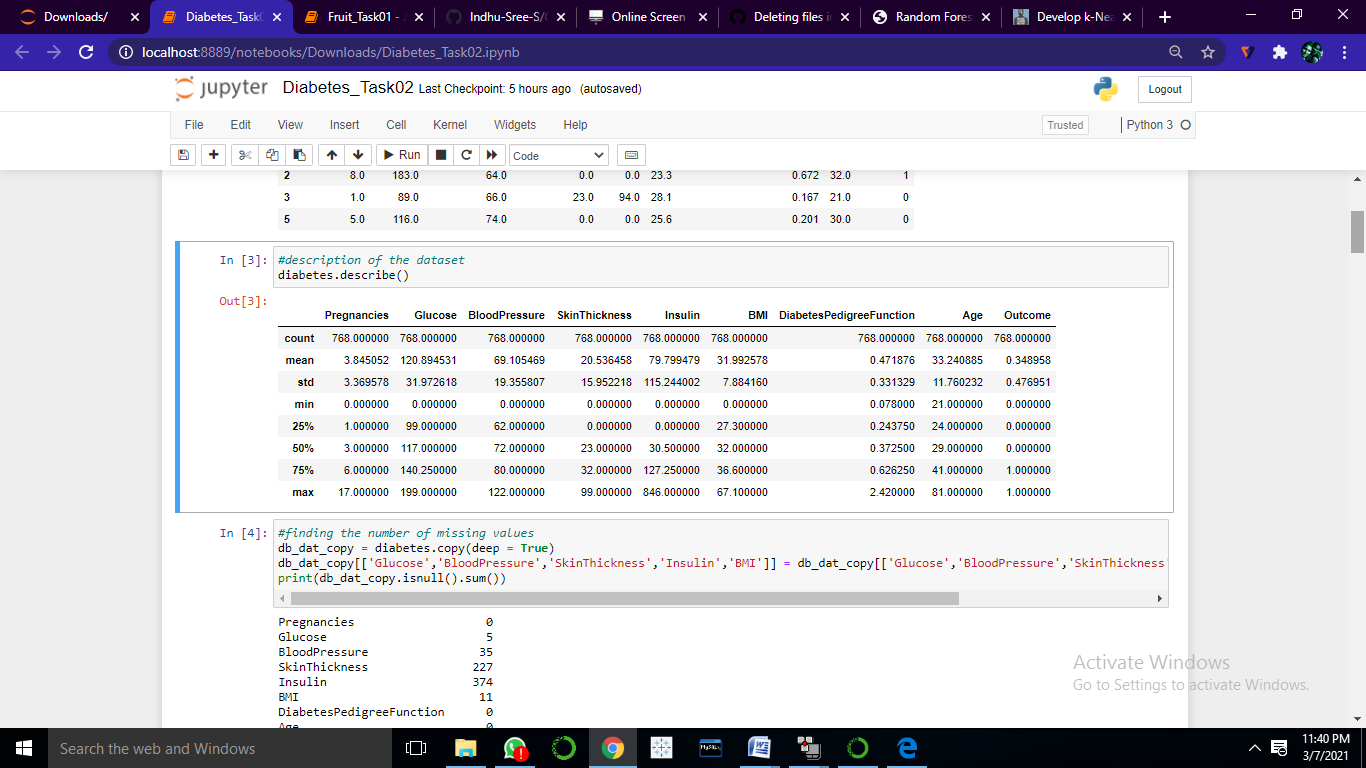
Predict which patient has diabetes from Diabetes Database.csv and try to understand the dataset attributes and try to figure out type ML model suits and build from scratch

**Dataset:**



* Pregnancies : Pregnancy rate of individuals
* Glucose: Glucose rate of individuals
* BloodPressure: BloodPressure rate of individuals
* Skin Thickness: Skin Thickness rate of individuals
* Insulin, BMI, Age : Insulin, BMI, Age of individuals
* Outcome: Depending upon the other attributes – Diabetes results is classified.
* Therefore classification of presences of diabetes based on Pregnancies, Glucose, BloodPressure, Skin Thickness, Insulin, BMI, Age

**Description:**



Above table shows that mean, minimum, maximum, quartiles, standard deviation and count of each variable that have numerical values.

**Presence of NULL:**

Pregnancies 0

Glucose 5

BloodPressure 35

SkinThickness 227

Insulin 374

BMI 11

DiabetesPedigreeFunction 0

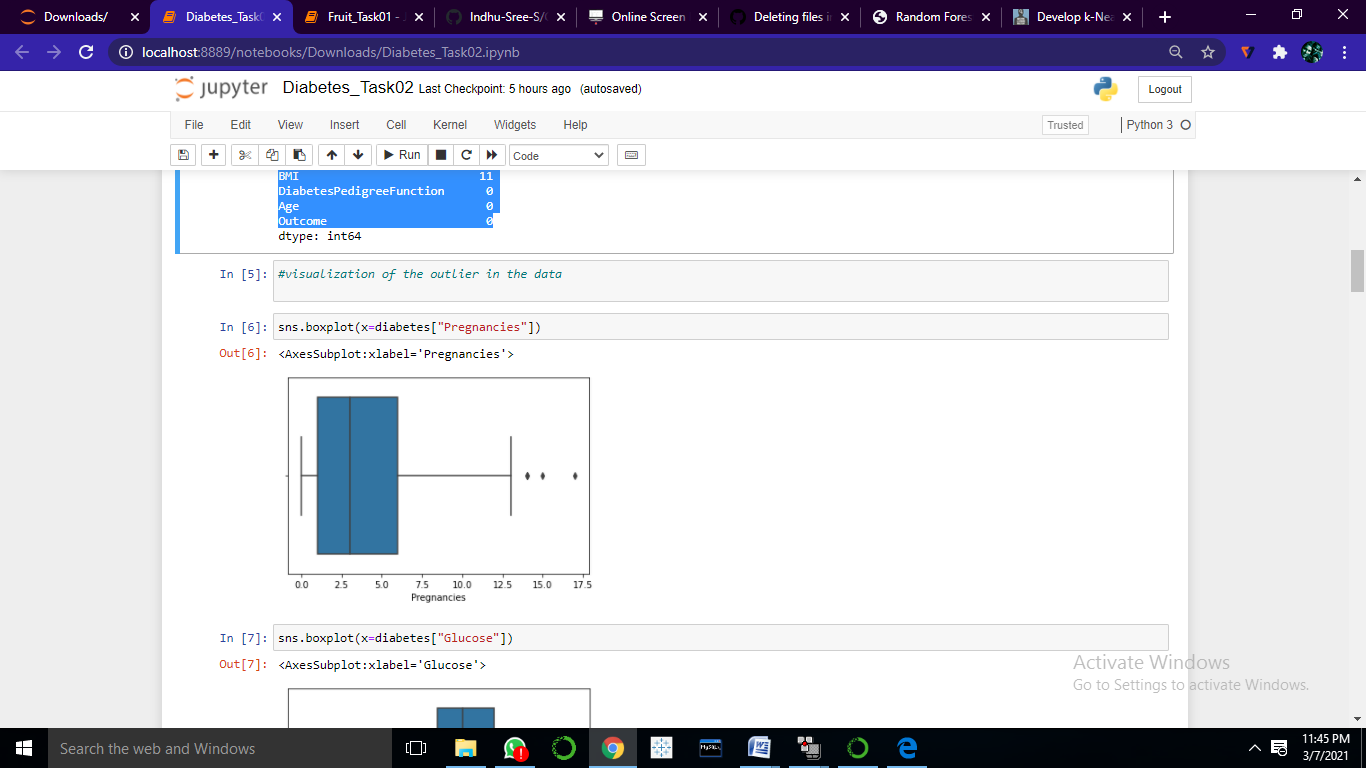
Age 0

Outcome 0

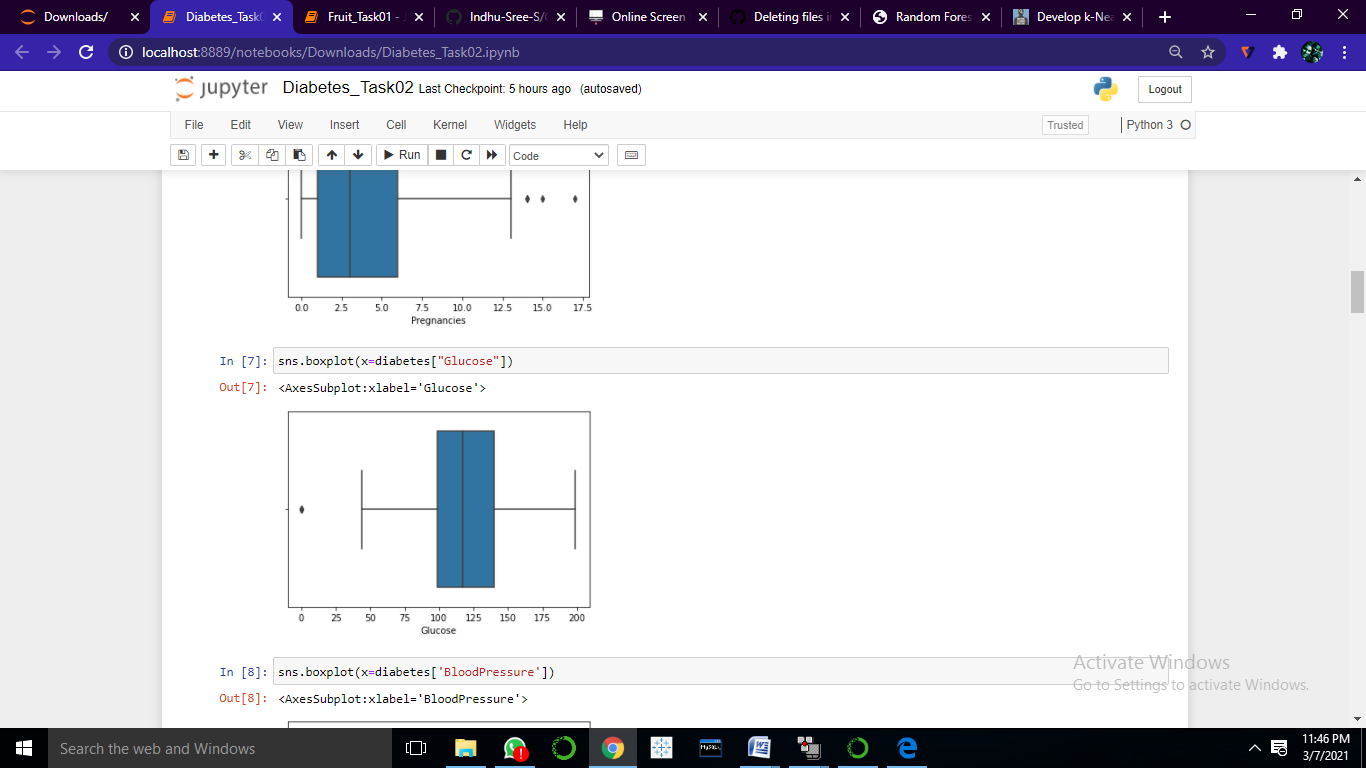
Bloodpressure, Skinthickness, Insulin, BMI have null values

**Outliers:**

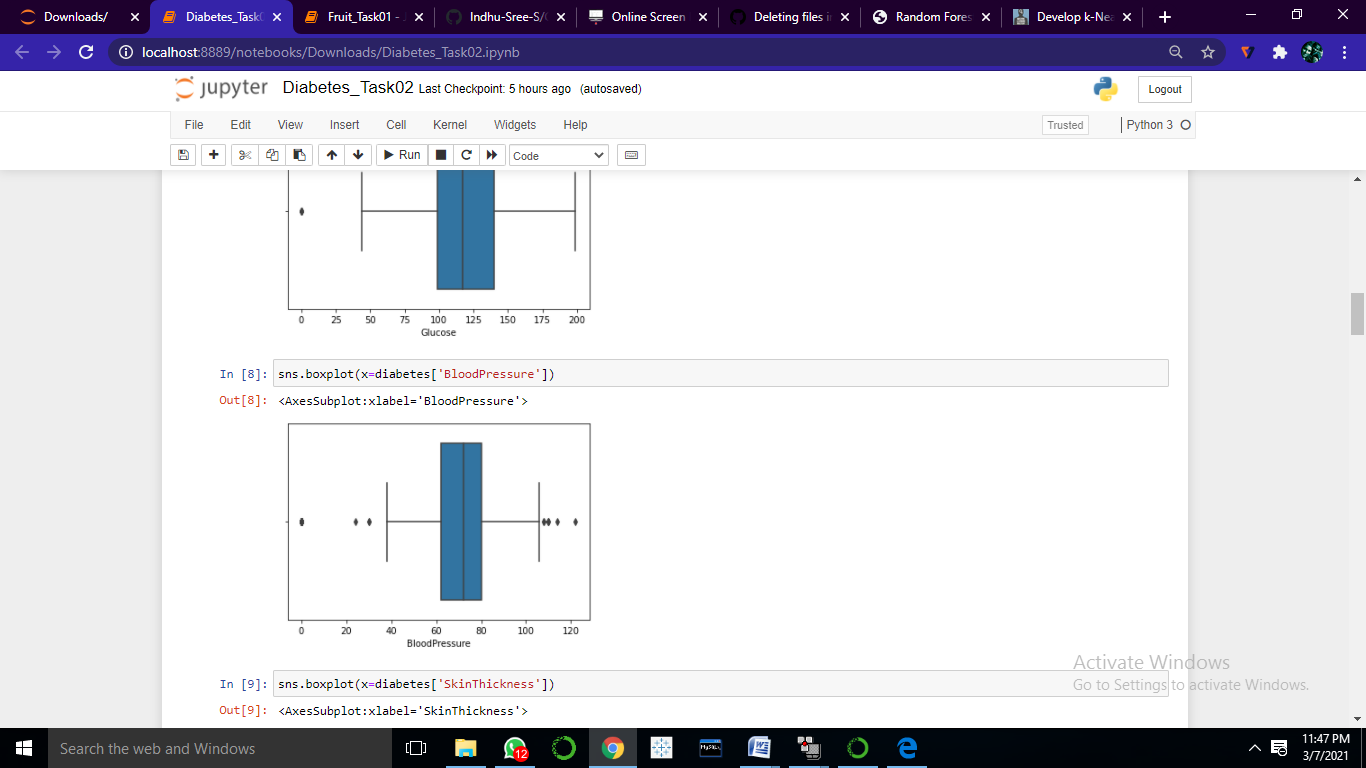
* Pregnancies



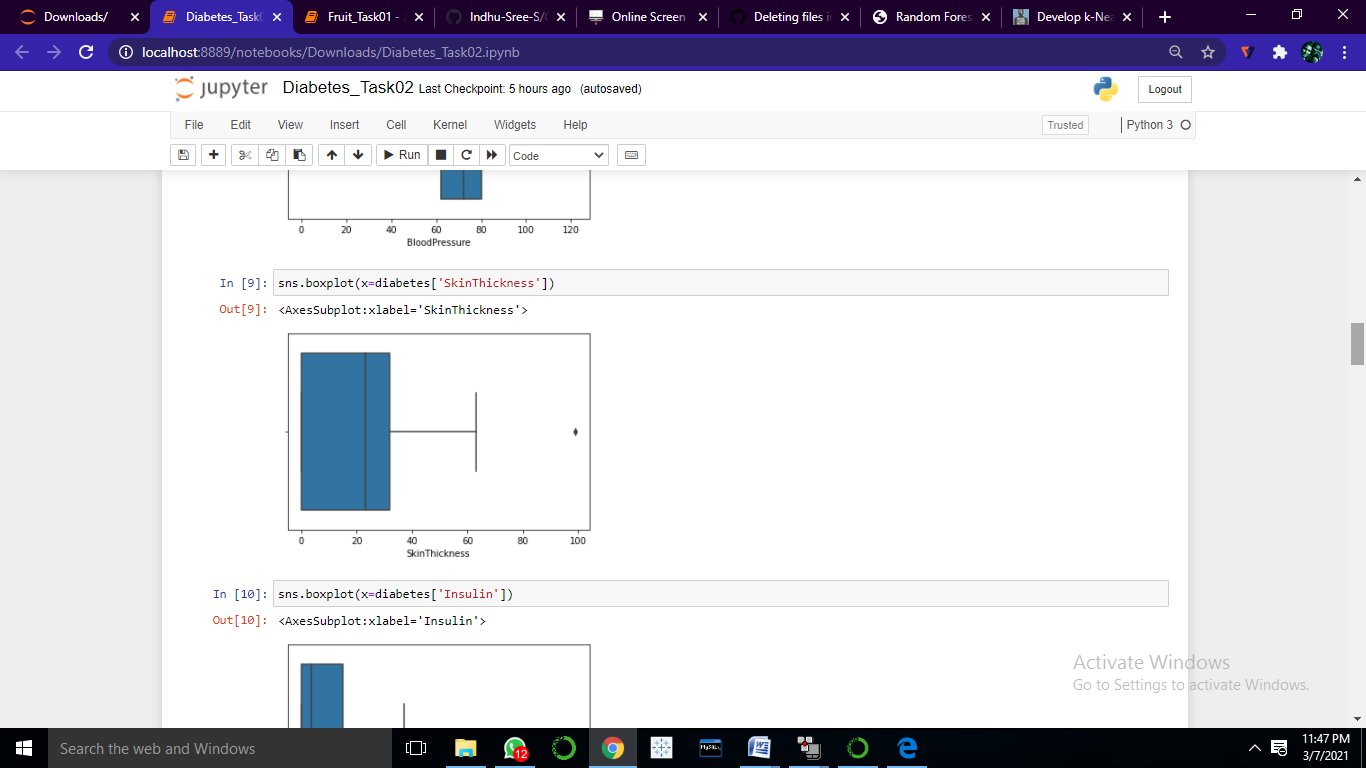
* Glucose



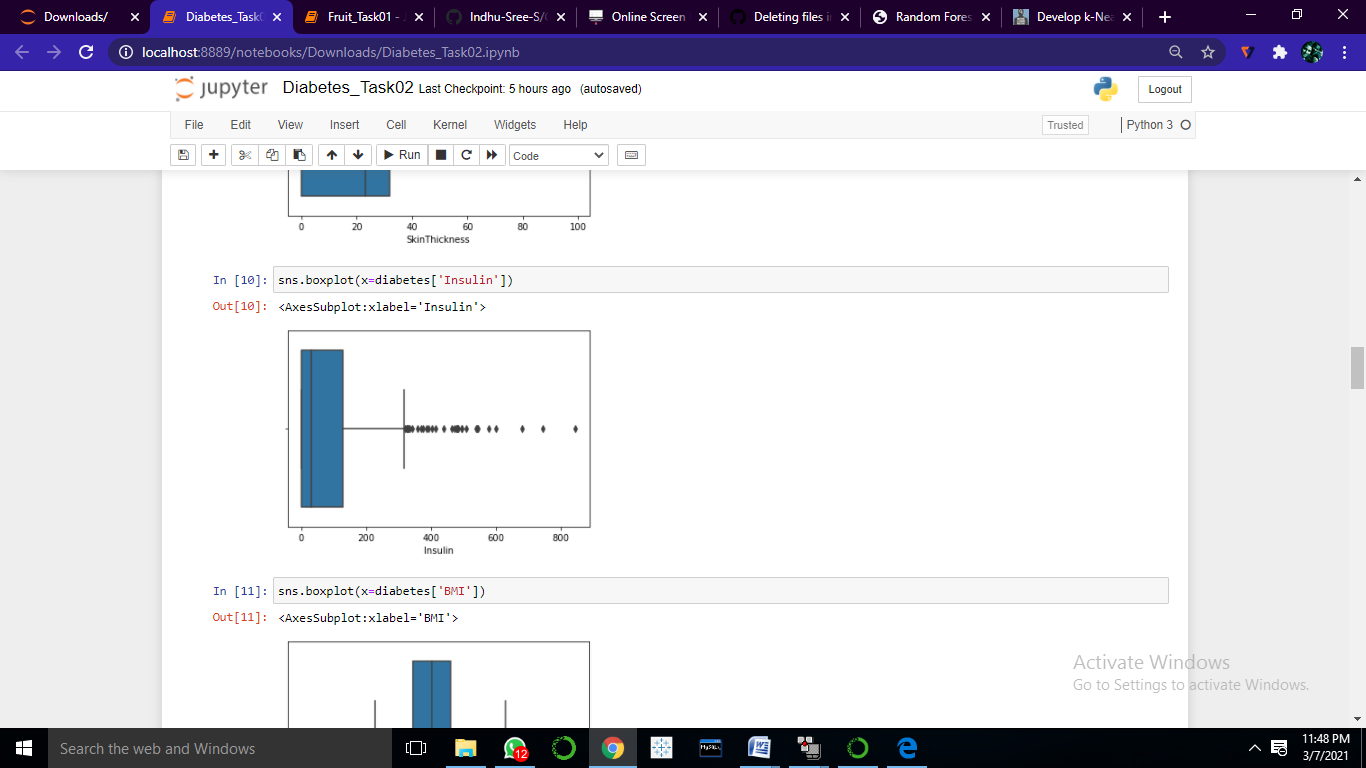
* Bloodpressure



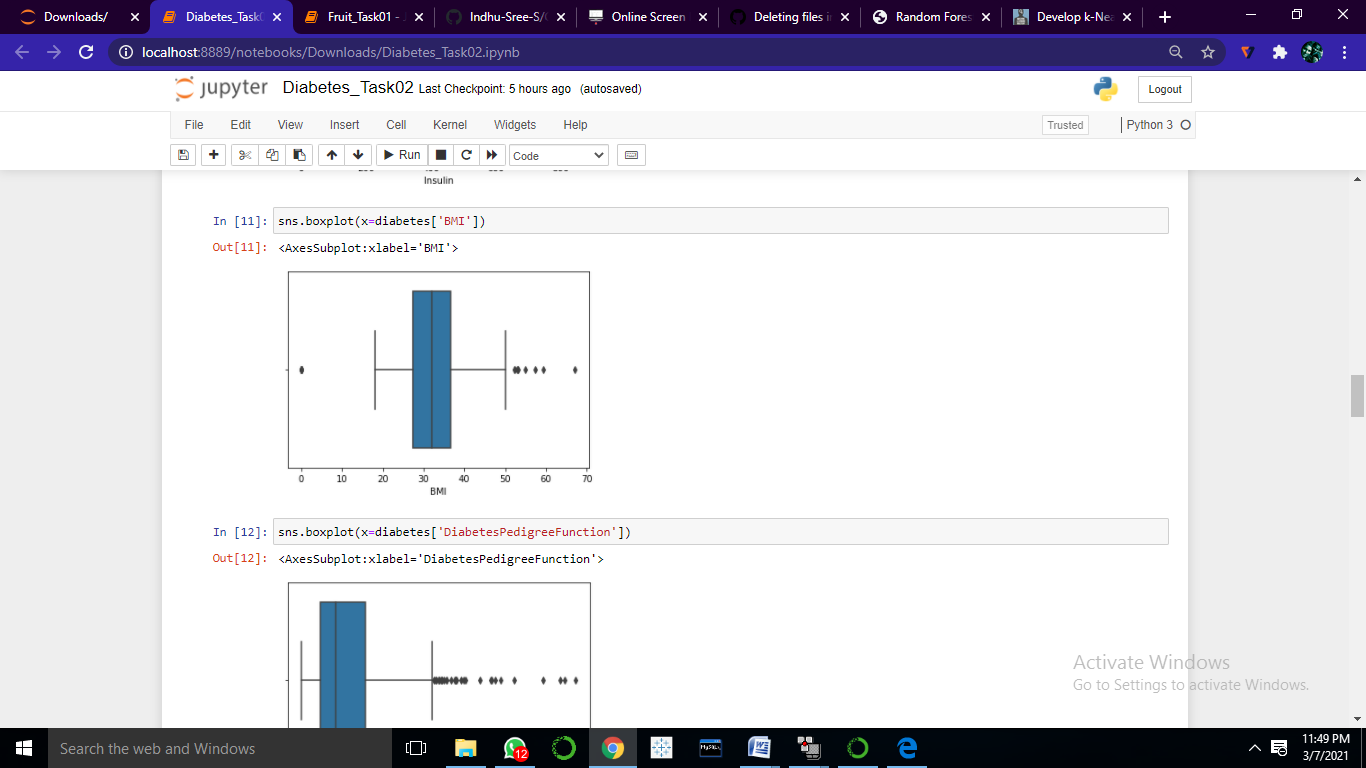
* Skinthickness



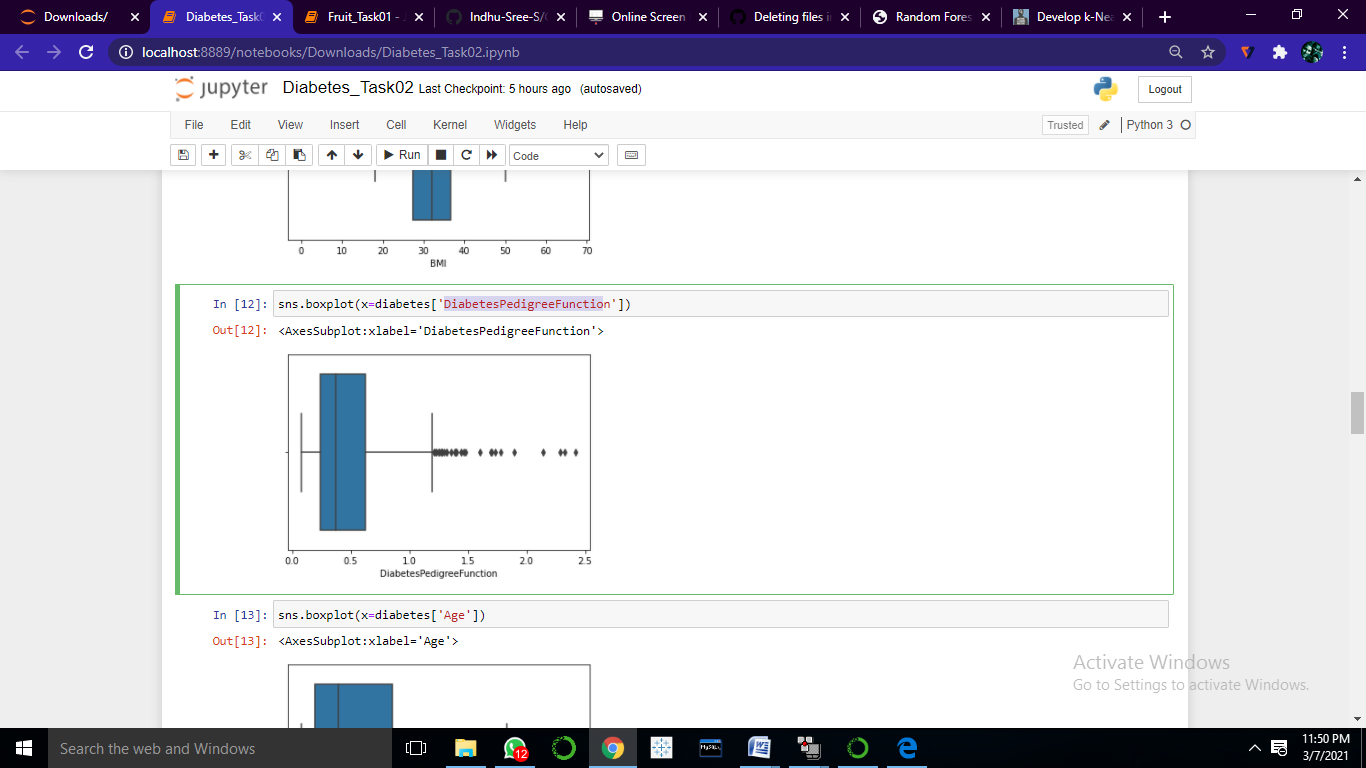
* Insulin



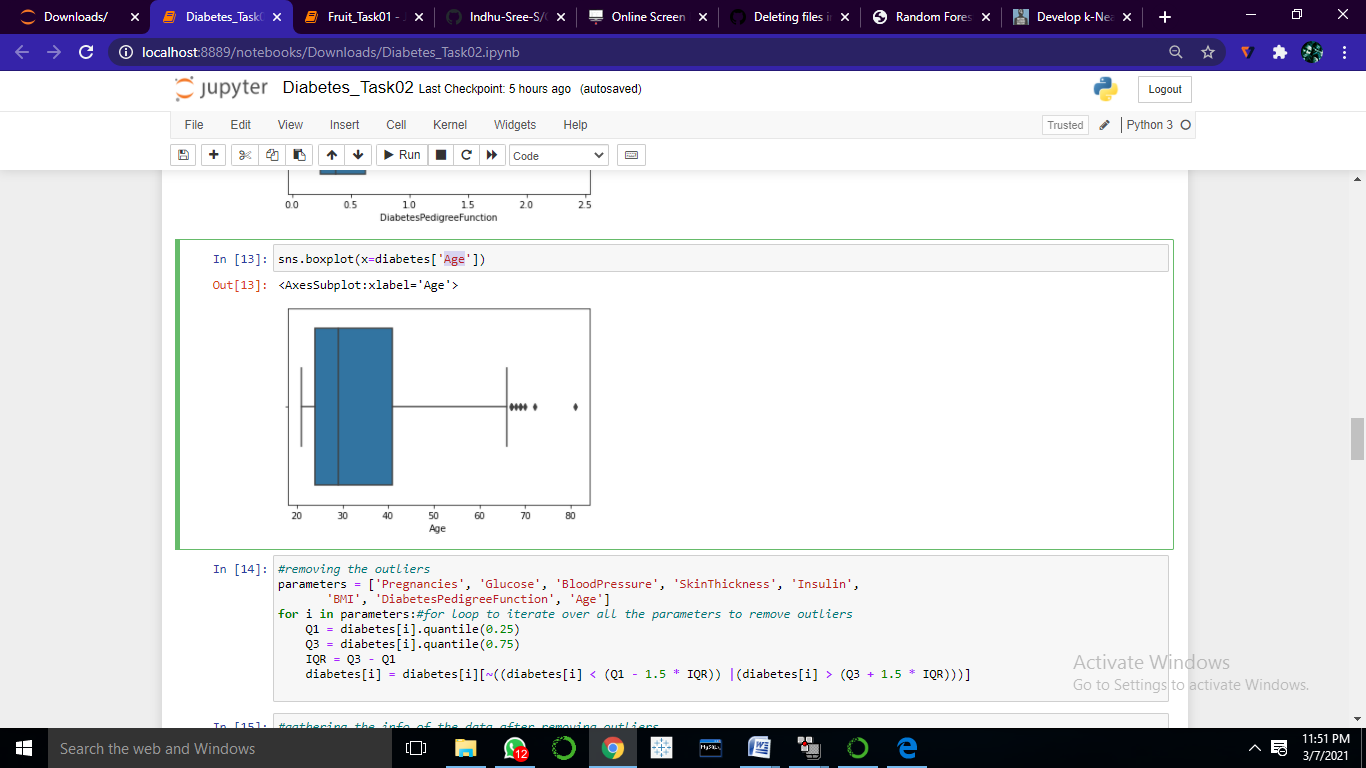
* BMI



* DiabetesPedigreeFunction



* Age



**k-Nearest Neighbors:**

* Calculate Euclidean Distance.
* Get Nearest Neighbors.
* Make Predictions.

**Euclidean Distance:**

* 1. Rows of data are mostly made up of numbers and an easy way to calculate the distance between two rows or vectors of numbers is to draw a straight line

Euclidean Distance = sqrt(sum i to N (x1\_i – x2\_i)^2)

def euclid\_dist(X1,X2):

dist = np.sum((X1 - X2)\*\*2)

return np.sqrt(dist)

**Function of knn predict:**

def knn\_predict(x\_train, x\_test, y\_train, y\_test, k):

# Counter to help with label voting

from collections import Counter

# Make predictions on the test data

# Need output of 1 prediction per test data point

y\_pred = []

for test\_point in x\_test:

distances = []

for train\_point in x\_train:

distance = euclid\_dist(test\_point, train\_point)

distances.append(distance)

# Storing distances in a dataframe

df\_dists = pd.DataFrame(data=distances, columns=['dist'],

index=y\_train.index)

# Sort distances and considering the k closest points

df\_nn = df\_dists.sort\_values(by=['dist'], axis=0)[:k]

# Create counter object to track the labels of k closest neighbors

counter = Counter(y\_train[df\_nn.index])

# Get most common label of all the nearest neighbors

prediction = counter.most\_common()[0][0]

# Append prediction to output list

y\_pred.append(prediction)

return y\_pred

* Neighbors for a new piece of data in the dataset are the k closest instances, as defined by our distance measure.
* To locate the neighbors for a new piece of data within a dataset we must first calculate the distance between each record in the dataset to the new piece of data. We can do this using our distance function prepared above.
* Once distances are calculated, we must sort all of the records in the training dataset by their distance to the new data. We can then select the top k to return as the most similar neighbors.

**Prediction**

196 0

374 0

590 1

660 0

271 0

..

124 1

538 0

27 0

257 0

209 1

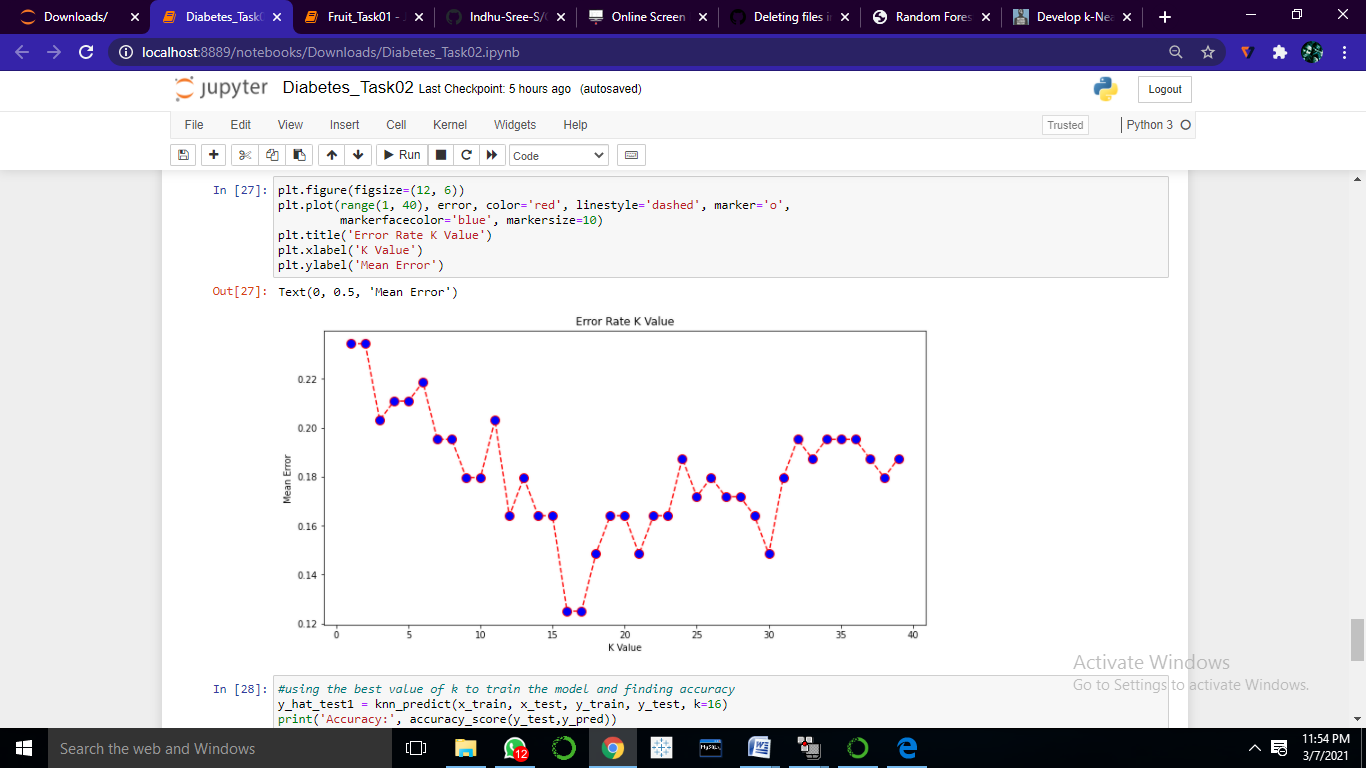
Name: Outcome, Length: 128, dtype: int64

[0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1]

Accuracy: 0.8359375

r 2\_score : 0.18840579710144922

**Mean Error vs K value:**

****

**Interpretation:**

Presence of Diabetes is classified by using K-Nearest Neighbor Classifier. It has an accuracy of 83% for this classification. The r2 score of the classifier is having 19%. The best K value is 16, which is calculated by r score value and the minimum error value. Thus the K-Nearest Neighbor classifier was implemented to classify and predict the presence of Diabetes.