PREDICTION OF DIABETES USING MACHINE LEARNING ALGORITHMS

|  |  |
| --- | --- |
| Number of Techniques implemented in Bucket 1: | 4 |
| 1.Names of the techniques implemented: | Without Null values, Mean, Median and Mode |
| 2. Data set used and its link: | Diabetes https://www.kaggle.com/uciml/pima-indians-diabetes-database |
| Number of Techniques implemented in Bucket 2: | 6 |
| 1. Names of the techniques implemented: | * Do Nothing, * Imputation Using (Mean/Median) Values * Imputation Using (Most Frequent) or (Zero/Constant) Values, * Imputation Using k-NN * Imputation Using Multivariate Imputation by Chained Equation (MICE) * Imputation Using Deep Learning ([Data wig](https://github.com/awslabs/datawig)) |
| 2. Data set used and its link: | Diabetes https://www.kaggle.com/uciml/pima-indians-diabetes-database |

BUCKET 1

|  |  |  |
| --- | --- | --- |
| S.No | Name | Justification |
| 1. | Technique Name | Without missing values |
|  | Dataset | Diabetes |
|  | Process | Step 1: Consider the Input Dataset without null values  Step 2: Train and Test the Input Data set  Step 3: Apply the Classification Algorithms like Decision tree and Gaussian Naive Bayes.  Step 4: Evaluate the accuracy of the non -existing null value dataset. |
|  | Output | Accuracy of Decision Tree :73.16  Accuracy of Gaussian Naive Bayes:76.19 |
|  | Result |  |
| 2. | Technique Name | Missing values by Mean |
|  | Dataset | Diabetes |
|  | Process | Step 1: Consider the Input Dataset with null values  Step 2: Train and Test the Input Data set  Step 3: Fill the missing values by calculation mean  Step 4: Apply the Classification Algorithms like Decision tree and Gaussian Naive Bayes.  Step 5: Evaluate the accuracy of the existing null value dataset. |
|  | Output | Accuracy: 0.79 |
|  | Result |  |
| 2. | Technique Name | Missing values by Median |
|  | Dataset | Diabetes |
|  | Process | Step 1: Consider the Input Dataset with null values  Step 2: Train and Test the Input Data set  Step 3: Fill the missing values by calculation median  Step 4: Apply the Classification Algorithms like Decision tree and Gaussian Naive Bayes  Step 5: Evaluate the accuracy of the existing null value dataset. |
|  | Output | Accuracy :0.78 |
|  | Result |  |
| 3. | Technique Name | Missing values by Mode |
|  | Dataset | Diabetes |
|  | Process | Step 1: Consider the Input Dataset with null values  Step 2: Train and Test the Input Data set  Step 3: Fill the missing values by mode  Step 4: Apply the Classification Algorithms like Decision tree and Gaussian Naive Bayes.  Step 5: Evaluate the accuracy of the existing null value dataset. |
|  | Output | Accuracy of Decision Tree :71.42  Accuracy of Gaussian Naive Bayes:76.19 |
|  | Result |  |

BUCKET 2

There are 6 Different Ways to Compensate for Missing Values in a Dataset

|  |  |  |
| --- | --- | --- |
| 1. | Technique Name | Do Nothing |
|  | Dataset | Diabetes |
|  | Process | Step 1: drop rows with missing values  Step 2: summarize the number of rows and columns in the dataset |
| 2. | Technique Name | Imputation Using (Mean/Median) Values: |
|  | Dataset | Diabetes |
|  | Process | Step 1: fill missing values calculating the mean/median of the non-missing values in a column and then replacing the missing values within each column separately |
| 3. | Technique Name | Imputation Using (Most Frequent) or (Zero/Constant) Values |
|  | Dataset | Diabetes |
|  | Process | Step 1: Impute the values using scikit -learn Simple Impute Class  Step 2: replacing missing data with the most frequent values within each column.  Step 3: replaces the missing values with either zero or any constant value specified |
| 4. | Technique Name | Imputation Using k-NN |
|  | Dataset | Diabetes |
|  | Process | Step 1: Consider the Input Dataset with null values  Step 2: Train and Test the Input Data set  Step 3: Apply the KNN Classification Algorithm  Step 4: Evaluate the accuracy of the non -existing null value dataset. |
|  | Output | Accuracy of KNN is 0.74 |
|  | Result |  |
| 5. | Technique Name | Imputation Using Multivariate Imputation by Chained Equation (MICE) |
|  | Dataset | Diabetes |
|  | Process | Step 1: imputation works by filling the missing data multiple times.  Step 2: start the MICE training |
| 6. | Technique Name | Imputation Using Deep Learning ([Datawig](https://github.com/awslabs/datawig" \t "_blank)) |
|  | Dataset | Diabetes |
|  | Process | Step 1: Initialize a SimpleImputer model  Step 2: Input column(s) containing information about the column we want to impute, the column we'd like to impute values for  Step 3: stores model data and metrics  Step 4: Impute missing values and return original dataframe with prediction as a separate file |
|  | Output | Predicted Output File. |

**BUCKET -1**

**Code:**

Before preprocessing

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import pandas as pd

from sklearn import tree

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

from sklearn.metrics import accuracy\_score

from sklearn.externals import joblib

import pandas as pd

import numpy as np

newpath="C:\\Users\\VITAP\\Desktop\\pima-indians-diabetes-database\\diabetes1.csv"

df=pd.read\_csv(newpath)

print(df.shape)

print(df.describe())

print(df.describe(include="all"))

missing\_data=df.isnull()

print(missing\_data)

print(df)

print(df.dtypes)

from sklearn.metrics import accuracy\_score

y = df['Outcome']

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

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

test\_size=0.3,

random\_state=0

)

clf = tree.DecisionTreeClassifier()

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

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

print ('Accuracy Score')

print (accuracy\_score(y\_test, y\_pred)\* 100)

from sklearn.naive\_bayes import GaussianNB

# Initialize our classifier

gnb = GaussianNB()

# Train our classifier

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

preds = gnb.predict(X\_test)

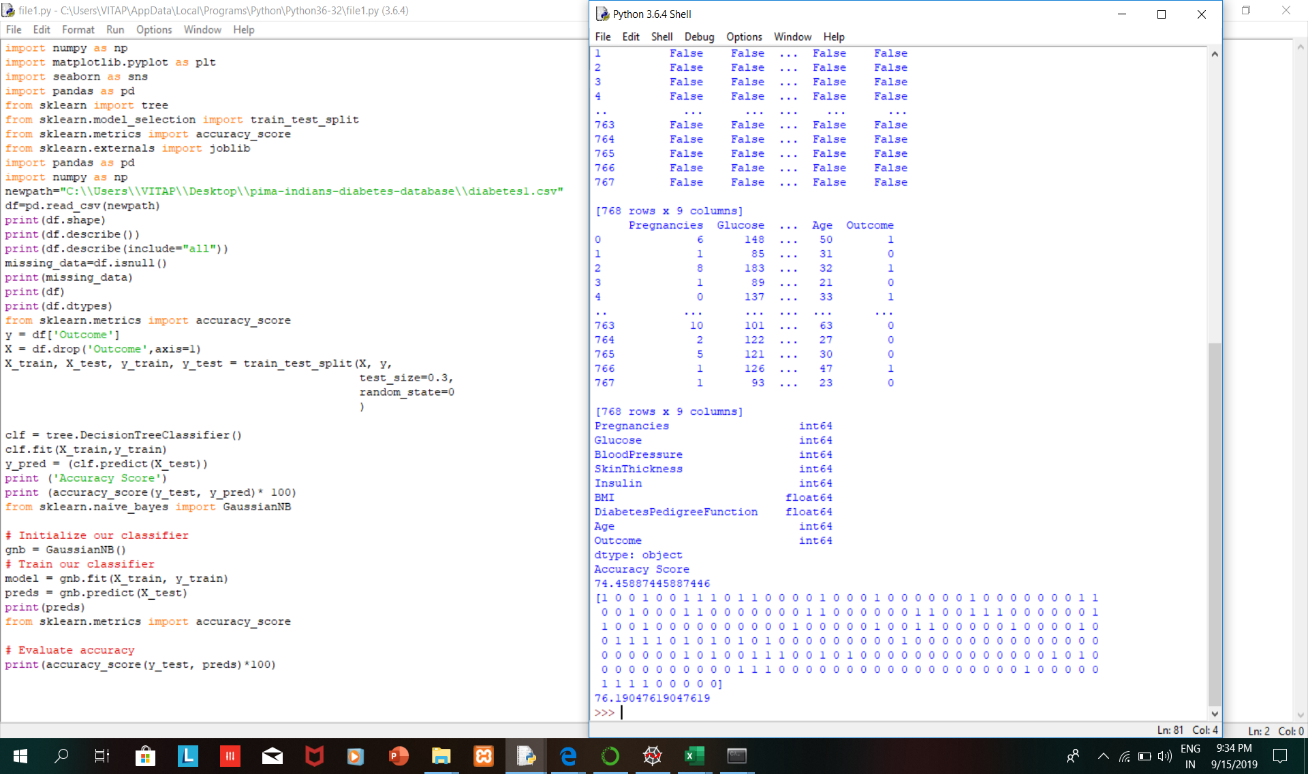
print(preds)

from sklearn.metrics import accuracy\_score

# Evaluate accuracy

print(accuracy\_score(y\_test, preds)\*100)

**Output:**



## Intentionally created missing values for pre-processing the dataset

**Fill the missing values using mean**

**Code**:

import random

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

dataset=pd.read\_csv('diabetes1.csv')

dataset

#generating null values randomly

random.seed(500)

column = dataset['Glucose']

missing\_pct = int(column.size \* 0.30)

i = [random.choice(range(column.shape[0])) for \_ in range(missing\_pct)]

column[i] = np.NaN

dataset.isnull().sum()

#fillling null values with mean

mean\_value=dataset['Glucose'].mean()

dataset['Glucose']=dataset['Glucose'].fillna(mean\_value)

X=pd.DataFrame(np.c\_[dataset['Pregnancies'],dataset['Glucose'],dataset['BloodPressure'],dataset['SkinThickness'],dataset['Insulin'],dataset['BMI'],dataset['DiabetesPedigreeFunction'],dataset['Age']],columns=['Pregnancies','Glucose','BloodPressure','SkinThickness','Insulin','BMI','DiabetesPedigreeFunction','Age'])

Y=pd.DataFrame(np.c\_[dataset['Outcome']],columns=['Outcome'])

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

X\_train,X\_test,Y\_train,Y\_test=train\_test\_split(X,Y,test\_size=0.2,random\_state=0)

from sklearn.naive\_bayes import GaussianNB

from sklearn.metrics import accuracy\_score

gnb = GaussianNB()

# Train our classifier

model = gnb.fit(X\_train, Y\_train)

# Make predictions

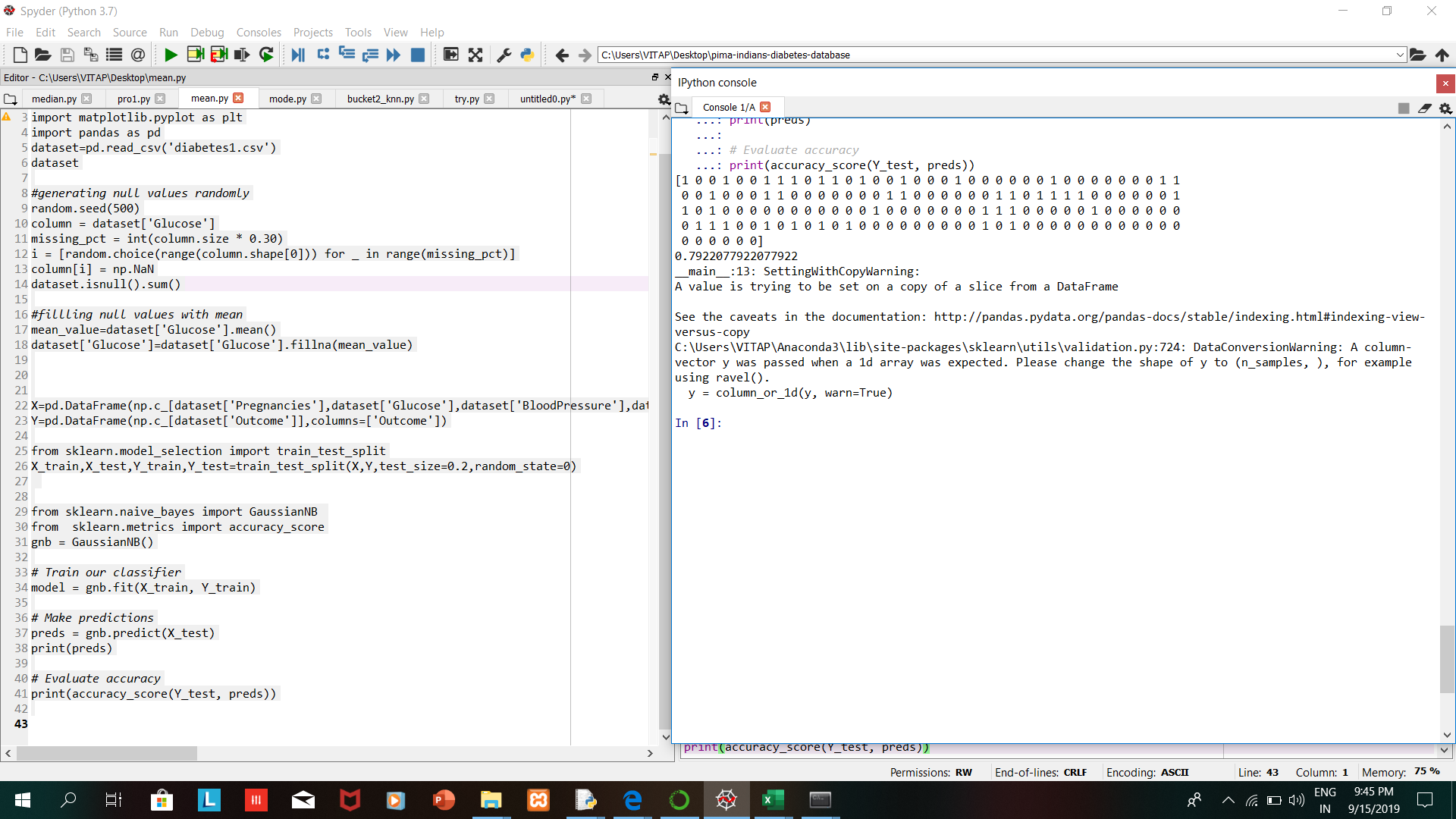
preds = gnb.predict(X\_test)

print(preds)

# Evaluate accuracy

print(accuracy\_score(Y\_test, preds))

**Output:**



**Fill the missing values using median**

**Code:**

import random

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

dataset=pd.read\_csv('diabetes1.csv')

dataset

#generating null values randomly

random.seed(500)

column = dataset['Glucose']

missing\_pct = int(column.size \* 0.30)

i = [random.choice(range(column.shape[0])) for \_ in range(missing\_pct)]

column[i] = np.NaN

dataset.isnull().sum()

#fillling null values with mean

median\_value=dataset['Glucose'].median()

dataset['Glucose']=dataset['Glucose'].fillna(median\_value)

X=pd.DataFrame(np.c\_[dataset['Pregnancies'],dataset['Glucose'],dataset['BloodPressure'],dataset['SkinThickness'],dataset['Insulin'],dataset['BMI'],dataset['DiabetesPedigreeFunction'],dataset['Age']],columns=['Pregnancies','Glucose','BloodPressure','SkinThickness','Insulin','BMI','DiabetesPedigreeFunction','Age'])

Y=pd.DataFrame(np.c\_[dataset['Outcome']],columns=['Outcome'])

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

X\_train,X\_test,Y\_train,Y\_test=train\_test\_split(X,Y,test\_size=0.2,random\_state=0)

from sklearn.naive\_bayes import GaussianNB

from sklearn.metrics import accuracy\_score

gnb = GaussianNB()

# Train our classifier

model = gnb.fit(X\_train, Y\_train)

# Make predictions

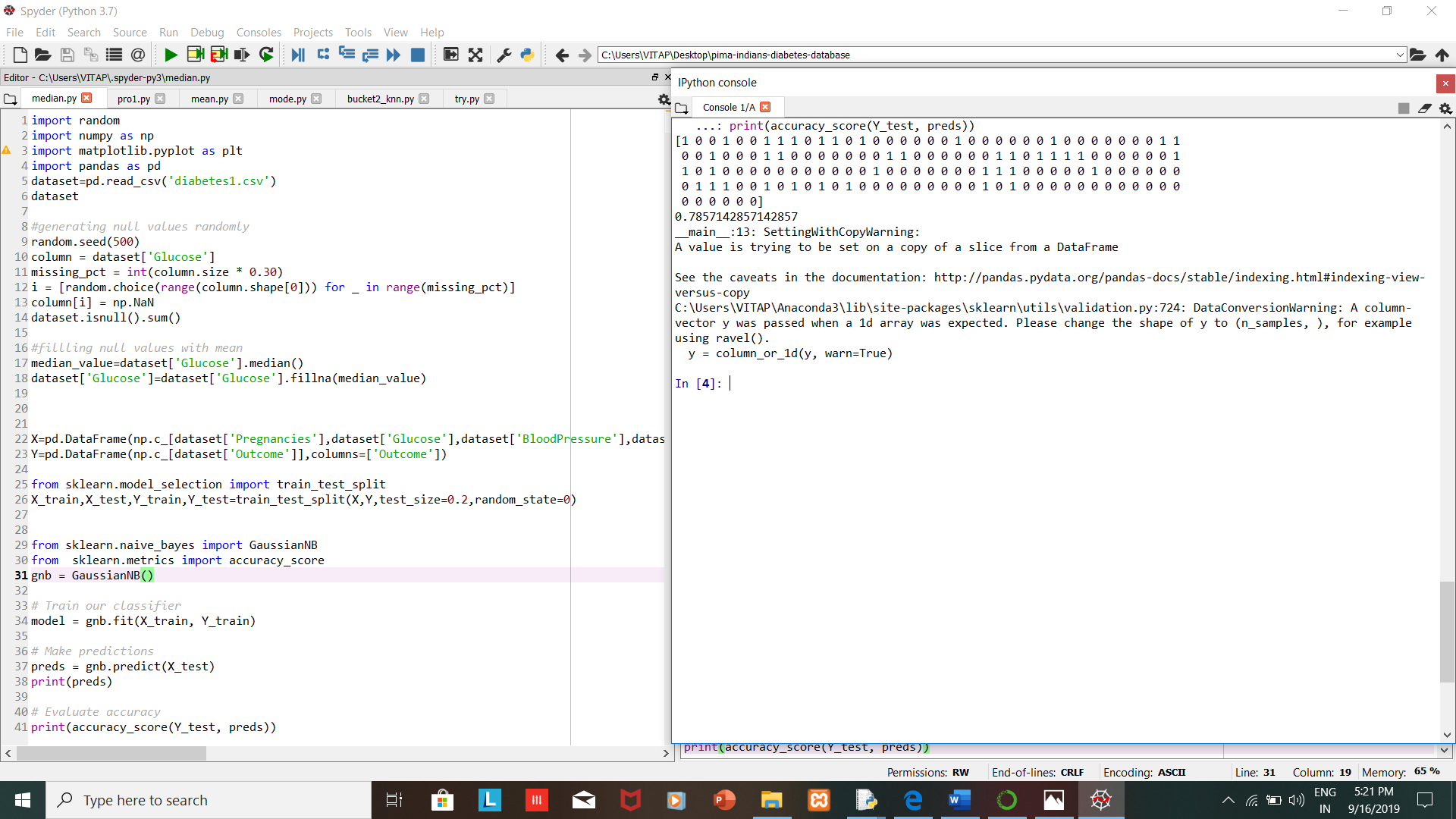
preds = gnb.predict(X\_test)

print(preds)

# Evaluate accuracy

print(accuracy\_score(Y\_test, preds))

**Output:**



**Fill the missing values using mode**

**Code:**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import pandas as pd

from sklearn import tree

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

from sklearn.metrics import accuracy\_score

from sklearn.externals import joblib

newpath="C:\\Users\\VITAP\\Desktop\\pima-indians-diabetes-database\\diabetes1.csv"

df=pd.read\_csv(newpath)

print(df.shape)

print(df.describe())

print(df.describe(include="all"))

missing\_data=df.isnull()

print(missing\_data)

print(df)

print(df.dtypes)

df['BMI']=df['BMI'].replace(" ",np.nan)

#df['BMI'].fillna(df.BMI.mode(), inplace=True)

mode = df['BMI'].mode()[0]

df['BMI'] = df['BMI'].fillna(mode)

newfilepath="C:\\Users\\VITAP\\Desktop\\pima-indians-diabetes-database\\missing.csv"

df.to\_csv(newfilepath)

y = df['Outcome']

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

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

test\_size=0.3,

random\_state=0,

)

clf = tree.DecisionTreeClassifier()

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

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

print ('Accuracy Score')

print (accuracy\_score(y\_test, y\_pred)\* 100)

from sklearn.naive\_bayes import GaussianNB

# Initialize our classifier

gnb = GaussianNB()

# Train our classifier

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

preds = gnb.predict(X\_test)

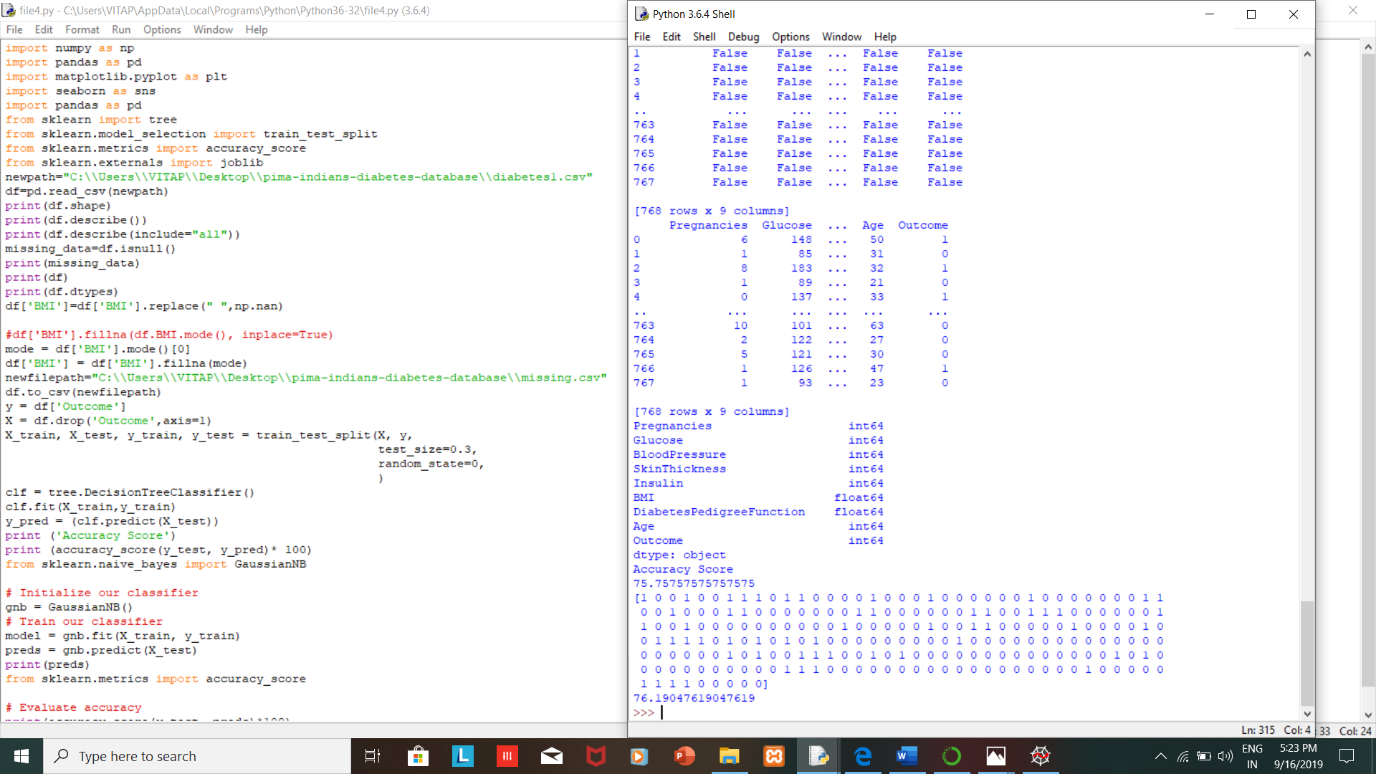
print(preds)

from sklearn.metrics import accuracy\_score

# Evaluate accuracy

print(accuracy\_score(y\_test, preds)\*100)

**Output:**



**BUCKET-2**

**The technique Do Nothing we used to handle the missing data**

**Code:**

dm=pd.read\_csv("C:\\Users\\VITAP\\Desktop\\pima-indians-diabetes-database\\missing.csv", header=None)

print(dm.describe())

dm.head()

print(dm.shape)

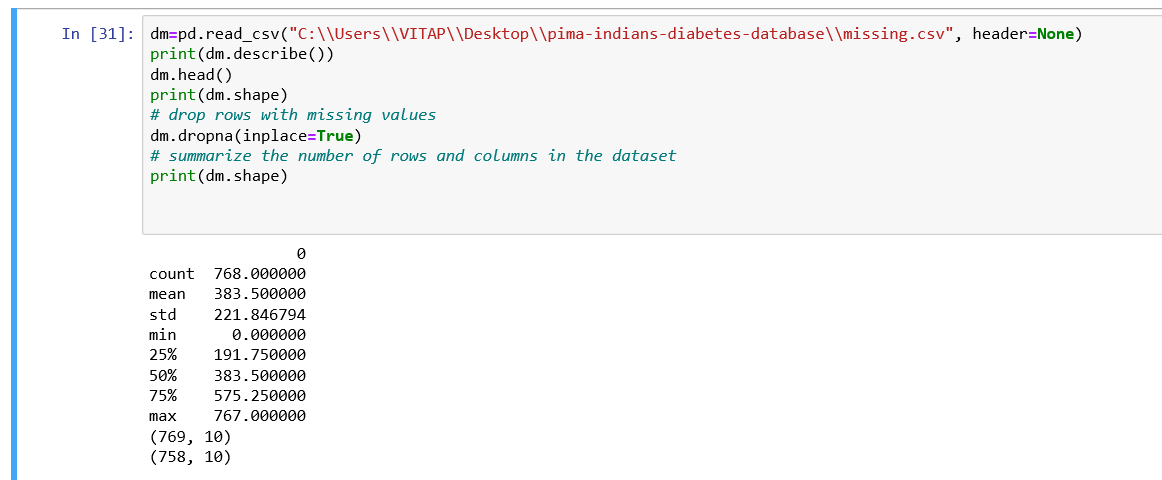
# drop rows with missing values

dm.dropna(inplace=True)

# summarize the number of rows and columns in the dataset

print(dm.shape)

**Output:**



**The technique Imputation Using (Mean/Median) Values we used to handle the missing data**

**Code1:**

dm=pd.read\_csv("C:\\Users\\VITAP\\Desktop\\pima-indians-diabetes-database\\missing.csv", header=None)

print(dm.describe())

# print the first 20 rows of data

print(dm.head(20))

# fill missing values with mean column values

print(dm.fillna(dm.mean(), inplace=True))

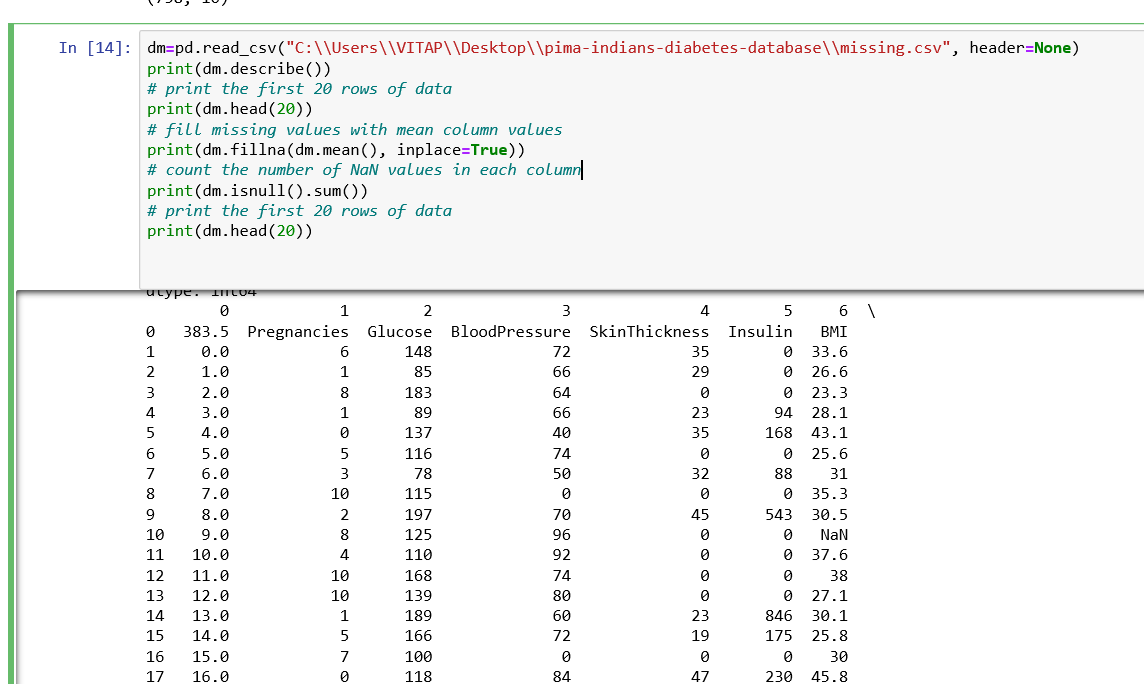
# count the number of NaN values in each column

print(dm.isnull().sum())

# print the first 20 rows of data

print(dm.head(20))

**Output:**



**Code2:**

import pandas as pd

dm=pd.read\_csv("C:\\Users\\VITAP\\Desktop\\pima-indians-diabetes-database\\missing.csv", header=None)

print(dm.describe())

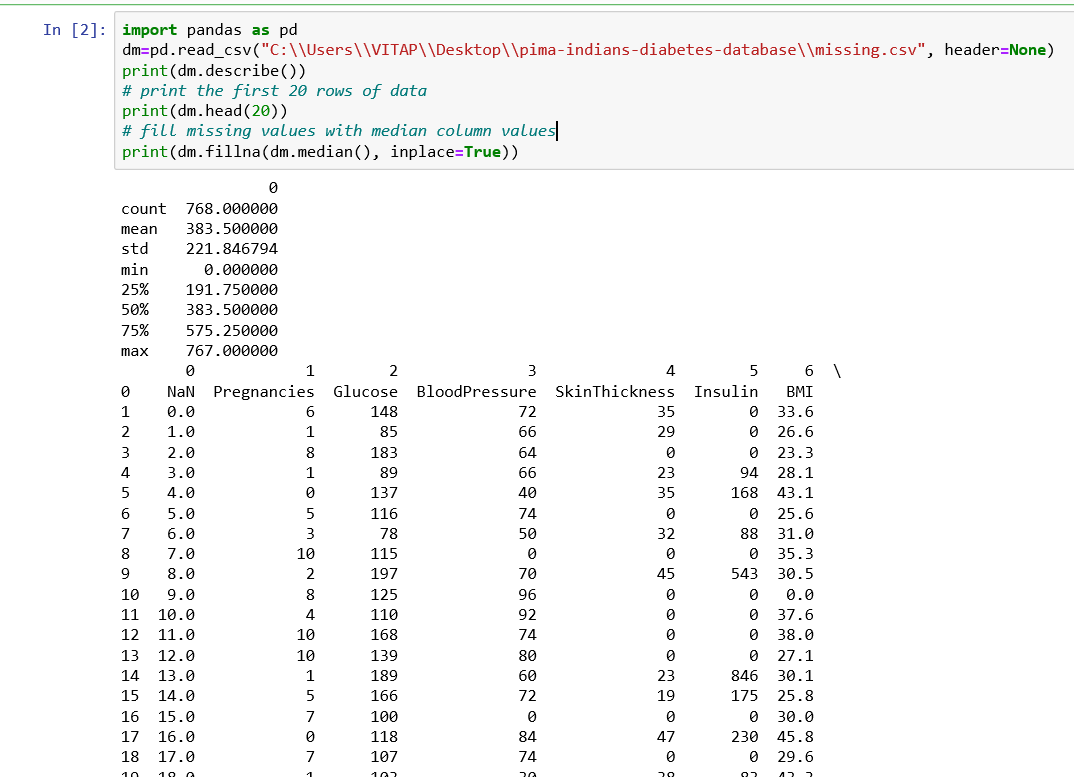
# print the first 20 rows of data

print(dm.head(20))

# fill missing values with median column values

print(dm.fillna(dm.median(), inplace=True))

**Output:**



**The technique Imputation Using (Most Frequent) or (Zero/Constant) Values we used to handle the missing data**

**Code:**

dm=pd.read\_csv("C:\\Users\\VITAP\\Desktop\\pima-indians-diabetes-database\\missing.csv", header=None)

dm.head()

#Impute the values using scikit-learn SimpleImpute Class

from sklearn.impute import SimpleImputer

imp\_mean = SimpleImputer( strategy='most\_frequent')

imp\_mean.fit(dm)

imputed\_dm\_df = imp\_mean.transform(dm)

dm.fillna(0)#here we can replaces the missing values with either zero or any constant value specified

**Output:**



**The technique Imputation Using k-NN we used to handle the missing data**

**Code:**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import pandas as pd

from sklearn import tree

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

from sklearn.metrics import accuracy\_score

from sklearn.externals import joblib

newpath="C:\\Users\\VITAP\\Desktop\\pima-indians-diabetes-database\\diabetes1.csv"

df=pd.read\_csv(newpath)

print(df.shape)

print(df.describe())

print(df.describe(include="all"))

missing\_data=df.isnull()

print(missing\_data)

print(df)

print(df.dtypes)

df['BMI']=df['BMI'].replace(" ",np.nan)

#mean=df['BMI'].astype("float").median(axis=0)

#print(mean)

#df['BMI']=df['BMI'].replace(np.nan,mean)

#df['BMI'].fillna(df.BMI.mode(), inplace=True)

mode = df['BMI'].mode()[0]

df['BMI'] = df['BMI'].fillna(mode)

newfilepath="C:\\Users\\VITAP\\Desktop\\pima-indians-diabetes-database\\missing.csv"

df.to\_csv(newfilepath)

y = df['Outcome']

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

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

test\_size=0.3,

random\_state=0,

)

#Import knearest neighbors Classifier model

from sklearn.neighbors import KNeighborsClassifier

#Create KNN Classifier

knn = KNeighborsClassifier(n\_neighbors=5)

#Train the model using the training sets

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

#Predict the response for test dataset

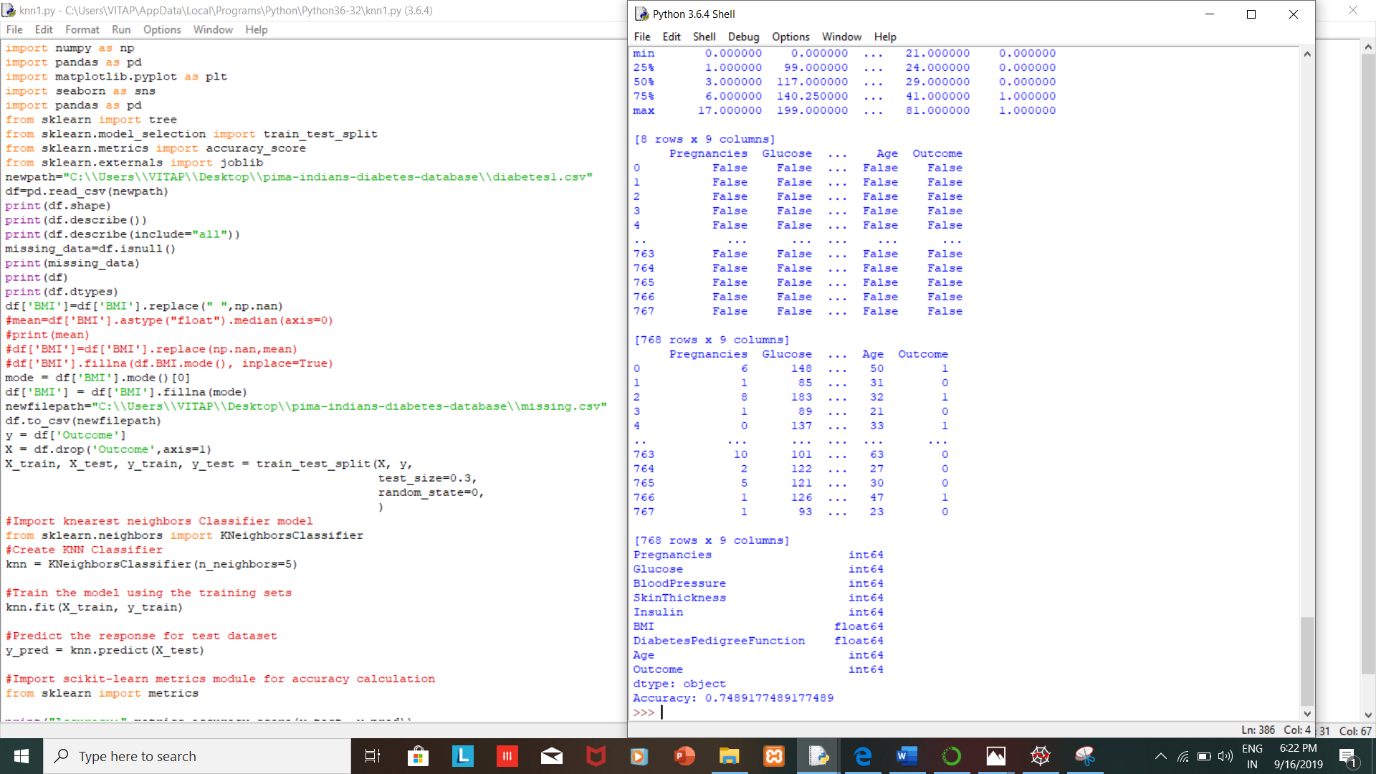
y\_pred = knn.predict(X\_test)

#Import scikit-learn metrics module for accuracy calculation

from sklearn import metrics

print("Accuracy:",metrics.accuracy\_score(y\_test, y\_pred))

**Output:**



**The technique Imputation Using Multivariate Imputation by Chained Equation (MICE) we used to handle the missing data**

**Code:**

dm = pd.read\_csv("C:\\Users\\VITAP\\Desktop\\pima-indians-diabetes-database\\missing.csv")

float\_data=dm.select\_dtypes(include=['float','integer'])

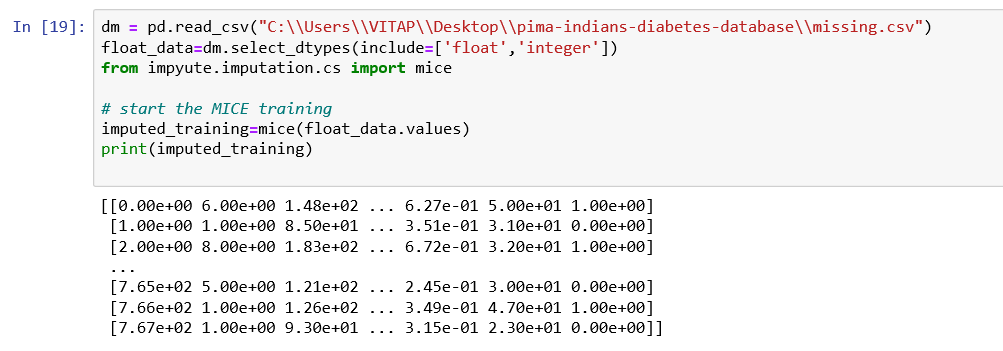
from impyute.imputation.cs import mice

# start the MICE training

imputed\_training=mice(float\_data.values)

print(imputed\_training)

**Output:**



**The technique Imputation Using Deep Learning ([Datawig](https://github.com/awslabs/datawig" \t "_blank)) we used to handle the missing data**

**Code:**

import datawig

df\_train, df\_test = datawig.utils.random\_split(dm)

#Initialize a SimpleImputer model

imputer = datawig.SimpleImputer(

input\_columns=['Pregnancies','Glucose','BloodPressure','SkinThickness','Insulin','BMI','DiabetesPedigreeFunction', 'Age'],

# column(s) containing information about the column we want to impute

output\_column='Outcome', # the column we'd like to impute values for

output\_path = 'imputer\_model' # stores model data and metrics

)

#Fit an imputer model on the train data

imputer.fit(train\_df=df\_train, num\_epochs=50)

#Impute missing values and return original dataframe with predictions

imputed = imputer.predict(df\_test)

print(imputed)

pd.DataFrame(imputed).to\_csv("C:\\Users\\VITAP\\Desktop\\pima-indians-diabetes-database\\file1.csv")

**Output:**

