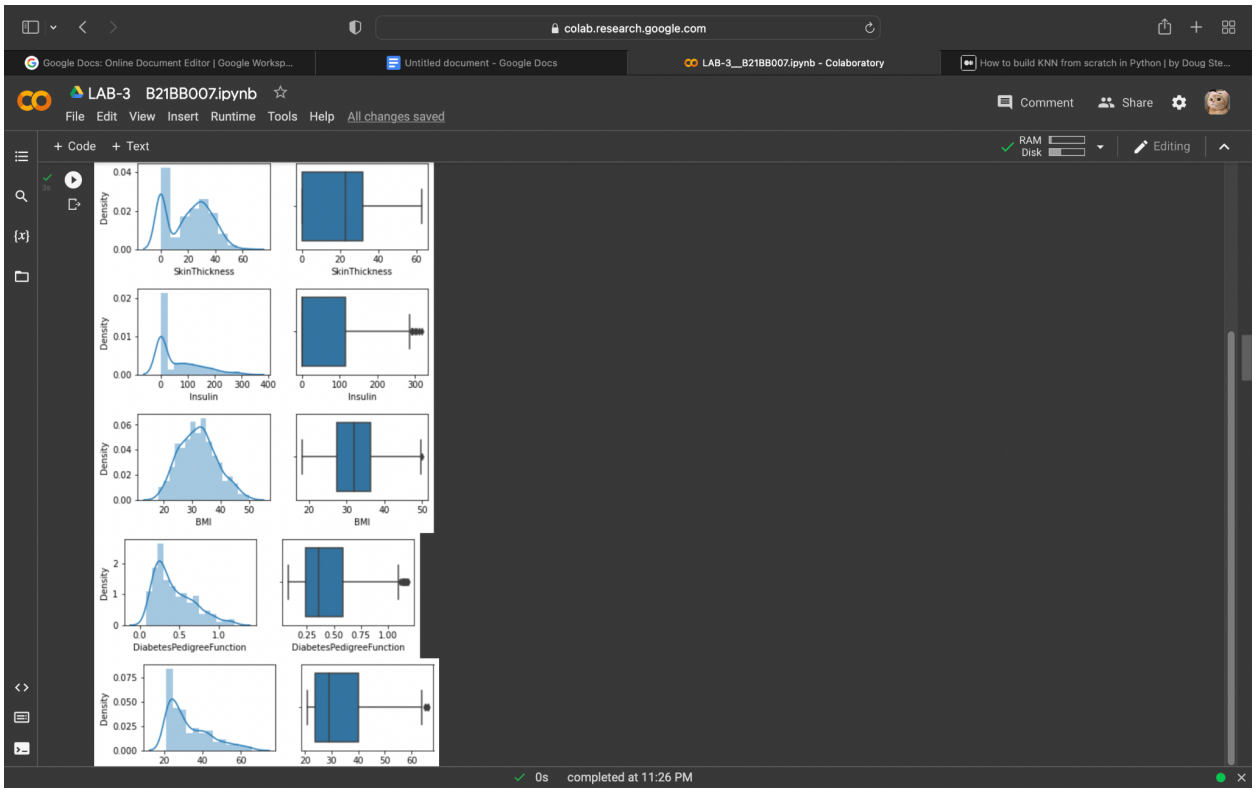
Smruti Dawale
B21BB007
Lab Assignment - 3

Que 1)

Here we have performed various for sorting the data ,we removed Nan values and also scaled the data so we could get minimum errors .

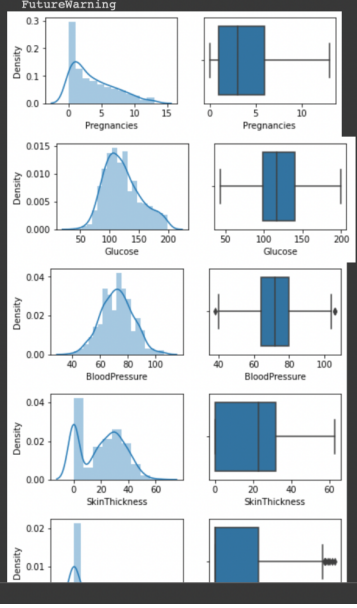


colab.research.google.com

LAB-3 B21BB007.ipynb

File Edit View Insert Runtime Tools Help All changes saved

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the on
  FutureWarning
```



0s completed at 11:26 PM

colab.research.google.com

LAB-3 B21BB007.ipynb

File Edit View Insert Runtime Tools Help All changes saved

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()

X = new_data.iloc[:,:9 ]
scaler.fit(X)
datascaled = scaler.transform(X)
```

```
datascaled
```

```
array([[ 0.67550897,  0.86698085, -0.01898104, ...,  0.78973882,
         1.55351335,  1.37620471],
       [-0.85047818, -1.19974348, -0.54053655, ..., -0.31671258,
        -0.15743859, -0.72663608],
       [ 1.28590383,  2.01516103, -0.71438838, ...,  0.9701385 ,
        -0.06738849,  1.37620471],
       ...,
       [ 0.37031154, -0.01875815, -0.01898104, ..., -0.74165405,
        -0.24748869, -0.72663608],
       [-0.85047818,  0.14526759, -1.06209205, ..., -0.32473034,
         1.28336304,  1.37620471],
       [-0.85047818, -0.93730229, -0.19283288, ..., -0.46103232,
        -0.8778394 , -0.72663608]])
```

```
[79] df_data = pd.DataFrame(datascaled)
     df_data
```

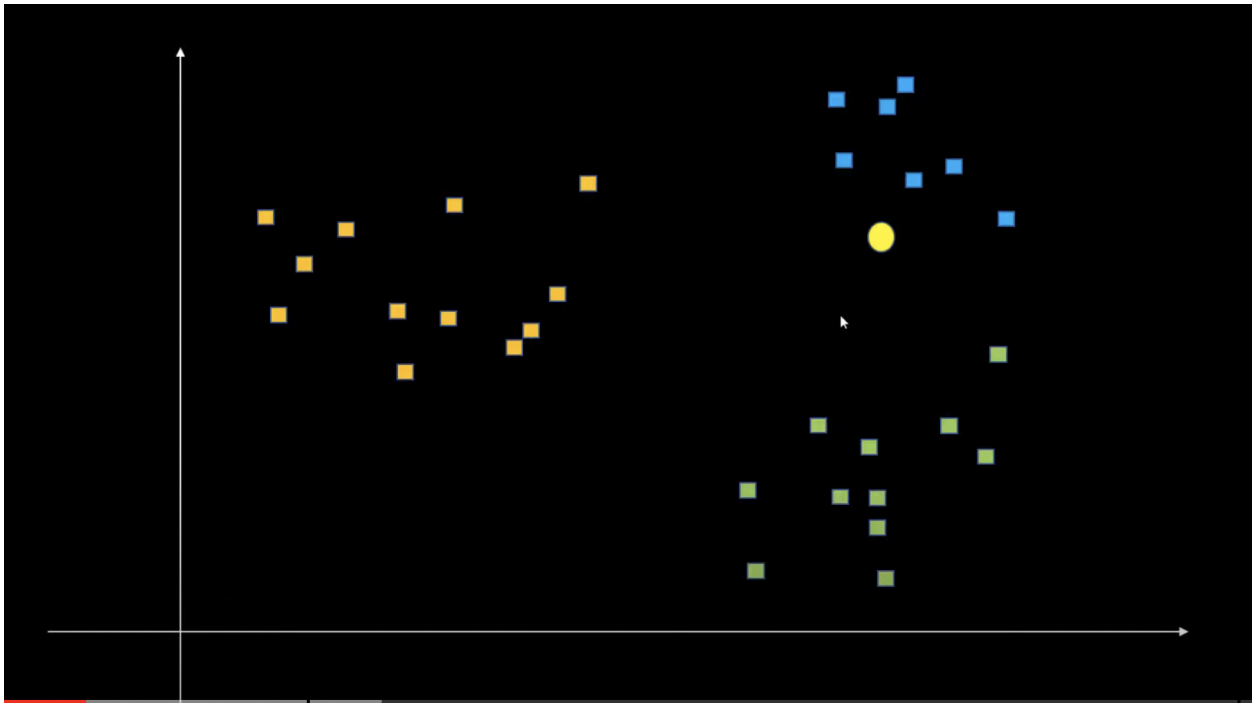
	0	1	2	3	4	5	6	7	8
0	0.675509	0.866981	-0.018981	0.930948	-0.785317	0.219044	0.789739	1.553513	1.376205
1	-0.850478	-1.199743	-0.540537	0.549059	-0.785317	-0.859222	-0.316713	-0.157439	-0.726636
2	1.285904	2.015161	-0.714388	-1.296737	-0.785317	-1.367547	0.970139	-0.067388	1.376205

0s completed at 11:48 PM

Que 2)

Here I have used the euclidean distance .The basic principle of Knn is - a specific data value acts like its neighbor. In Knn you first figure out the value of K , suppose k = 3 which implies you need to figure out the most nearest 3 data points (Here using euclidean distance).

Here in this graph , the yellow dot is the missing data point

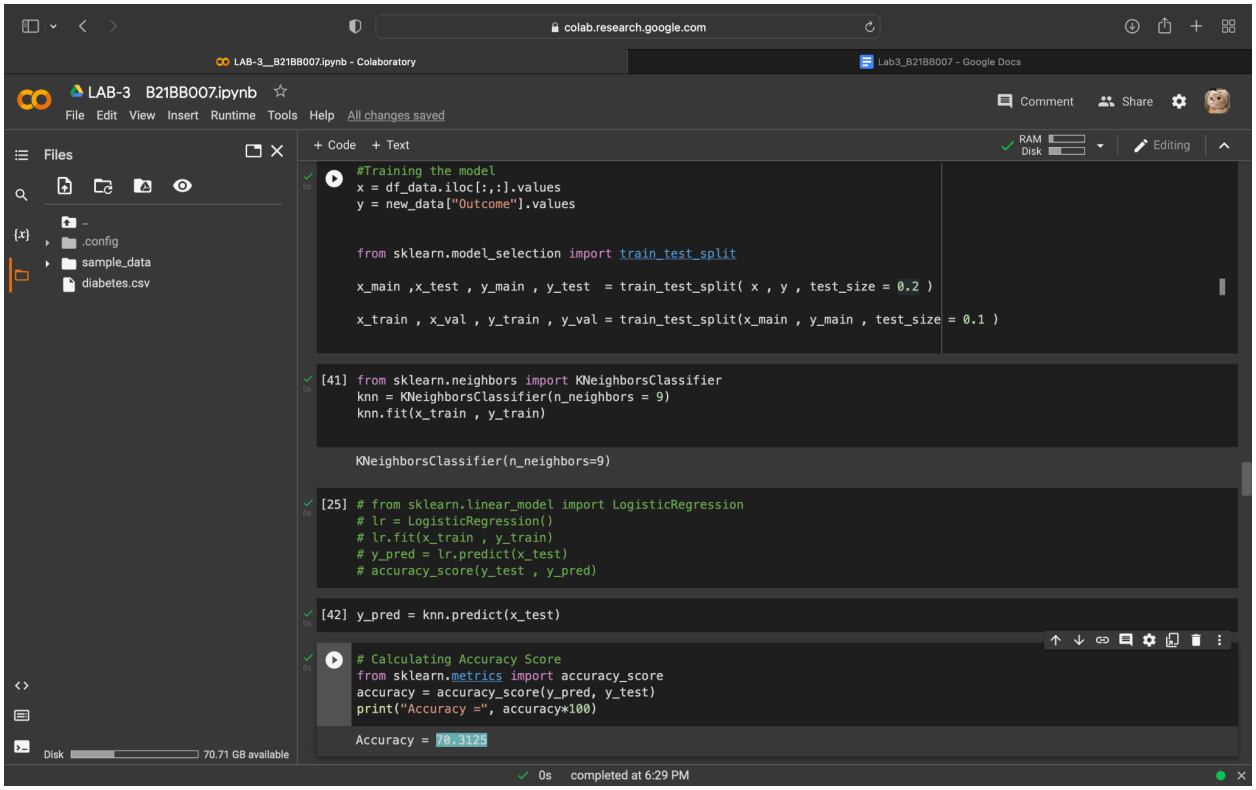


when k = 3 its nearest neighbors are the blue dots.
Let's say you take k = 10 , it considers the highest count of particular dots and assigns the missing data that type (still blue dots are more).Now consider a case k = 20 ,then the total no of blue points are less and data type gets mismatched , thats why you need to find the correct value of k neither too high nor too low.

We train the model first taking our input as X (i have dropped the “Outcome”) , Y as output i.e “Outcome”.
Import `from sklearn.neighbors import KNeighborsClassifier`

```
knn = KNeighborsClassifier(n_neighbors= 10)
```

Later train the model using knn.fit

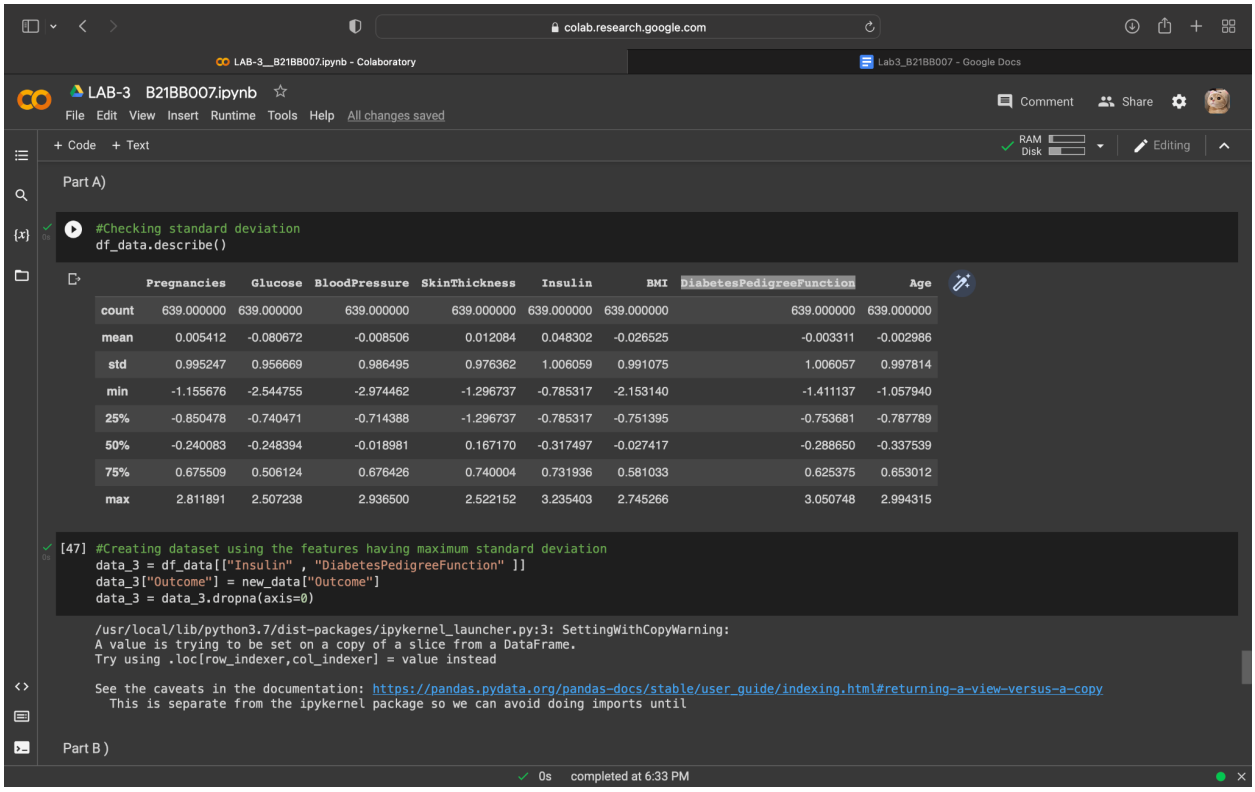


Here k = 10 , and we got accuracy of model 76.5625

When k = 4 , accuracy = 72.65625
k = 5 , accuracy = 74.21875
k = 6 , accuracy = 69.53125
k = 7 , accuracy = 71.875
k = 8 , accuracy = 68.75
k = 9 , accuracy = 70.3125

Que 3)

a) In this I checked the standard deviation using data.describe( )



	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
count	639.000000	639.000000	639.000000	639.000000	639.000000	639.000000	639.000000	639.000000
mean	0.005412	-0.080672	-0.008506	0.012084	0.048302	-0.026525	-0.003311	-0.002986
std	0.995247	0.956669	0.986495	0.976362	1.006059	0.991075	1.006057	0.997814
min	-1.155676	-2.544755	-2.974462	-1.296737	-0.785317	-2.153140	-1.411137	-1.057940
25%	-0.850478	-0.740471	-0.714388	-1.296737	-0.785317	-0.751395	-0.753681	-0.787789
50%	-0.240083	-0.248394	-0.018981	0.167170	-0.317497	-0.027417	-0.288650	-0.337539
75%	0.675509	0.506124	0.676426	0.740004	0.731936	0.581033	0.625375	0.653012
max	2.811891	2.507238	2.936500	2.522152	3.235403	2.745266	3.050748	2.994315

b) Here , first train the model defining your input and output variable.
Bins are for discretization of data on the basis of various parameters.

colab.research.google.com

LAB-3__B21BB007.ipynb - Colaboratory | Lab3_B21BB007 - Google Docs

LAB-3 B21BB007.ipynb

File Edit View Insert Runtime Tools Help All changes saved

```
[48] x = np.array(data_3["Insulin"]).reshape(-1 , 1)
     y = np.array(data_3["Outcome"]).reshape(-1 , 1)

     from sklearn.model_selection import train_test_split

     from sklearn.neighbors import KNeighborsClassifier


     x_main ,x_test , y_main , y_test  = train_test_split( x , y , test_size = 0.2 )

     x_train , x_val , y_train , y_val = train_test_split(x_main , y_main , test_size = 0.1 )
```

```
# Discretize the features into n bins
from sklearn.preprocessing import KBinsDiscretizer
bin_disc = KBinsDiscretizer(n_bins= 10, encode='ordinal',strategy='uniform')
x_train_transformed = (bin_disc.fit_transform(x_train))
```

```
[62] Knn = KNeighborsClassifier(n_neighbors = 10)
     Knn.fit(x_train_transformed , y_train)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/neighbors/_classification.py:198: DataConversionWarning: A column-vector y was passed when a 1d array was expe
  return self._fit(X, y)
KNeighborsClassifier(n_neighbors=10)
```

```
[63] # Making Prediction
     y_pred = Knn.predict(x_test)
```

```
[64] # Calculating Accuracy Score
     from sklearn.metrics import accuracy_score
     accuracy = accuracy_score(y_pred, y_test)
     print("Accuracy =", accuracy*100)
```

```
Accuracy = 70.9090909090909
```

0s completed at 6:33 PM

colab.research.google.com

LAB-3__B21BB007.ipynb - Colaboratory | Lab3_B21BB007 - Google Docs

LAB-3 B21BB007.ipynb

File Edit View Insert Runtime Tools Help All changes saved

```
[64] accuracy = accuracy_score(y_pred, y_test)
     print("Accuracy =", accuracy*100)
```

```
Accuracy = 70.9090909090909
```

```
[65] x = np.array(data_3["DiabetesPedigreeFunction"]).reshape(-1 , 1)
     y = np.array(data_3["Outcome"]).reshape(-1 , 1)

     from sklearn.model_selection import train_test_split

     from sklearn.neighbors import KNeighborsClassifier


     x_train ,x_test , y_train , y_test  = train_test_split( x , y , test_size = 0.2 ,random_state = 34 )
```

```
from sklearn.preprocessing import KBinsDiscretizer
bin_disc = KBinsDiscretizer(n_bins= 6, encode='ordinal',strategy='uniform')
x_train_transformed = (bin_disc.fit_transform(x_train))
```

```
[67] y_pred = Knn.predict(x_test)
```

```
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_pred, y_test)
print("Accuracy =", accuracy*100)
```

```
Accuracy = 74.54545454545455
```

0s completed at 6:33 PM

The accuracy of the bins method is more .

Naive Bayes is a **linear classifier** while K-NN is not; It tends to be faster when applied to big data. In comparison, k-nn is usually slower for large amounts of data, because of the calculations required for each new step in the process. If speed is important, choose Naive Bayes over K-NN.