PROJECT REPORT

WARDROBE SORTER

**ABSTRACT AND MOTIVATION**

It is known fact that neural networks are essentially a combination of complex algorithms and interconnected nodes, prepared to make a software and/or hardware function like human neurons. This technology using artificial intelligence is an emerging field and can be used to solve several daily life problems. As hostellers, a problem that we often are confronted with, is a pile of used up clothes that ends up cluttering our rooms. We often give up to procrastination till it amounts to a mountain ready for an avalanche. Henceforth, we decided to come up with a solution using neural networks and image processing. The main aim during this project will be to figure out a mechanism and a system that sorts out the pile of clothing into different categories based on their contours (i.e. cataloguing shirts, jeans etc.)

The reason we chose this particular topic is because it simplifies the lives of tons of students like us who could utilise the time doing something more productive or of their desire that they would usually spend on just clearing up the heap of clothing around them.

**METHODOLOGY**

For better understanding of the process involved, we have divided the procedure broadly into 3 sections namely: identification, selection and separation.

**Identification**: The primary section of the system consists of the camera that will be used to focus on the clothing stack for the commencement of the process. Now, using background subtraction and adjustment of threshold values, specific clothes will be identified. This ensures the beginning of the upcoming selection technique.

**Selection**: Next, the contours of the clothes will be mapped. Accordingly, the contour with the largest surface area is given greater importance in the order of picking up of the clothes. This makes sure of the fact that the clothes on top of the pile are picked up first and then the rest in the line. This systematic selection is vital since it also prevents falling out of clothes and further disorder.

**Seperation**: Without further ado, the following step will eventually be the separation of the picked up cloth and allotment to the appropriate category (jeans, shirts etc.)

This is where the “brain” of the established system is developed and the learning takes place by the process of practice and training.

The agenda now, will be to train the neural structure the art of identification of types of clothing by pattern and cluster formation.

**SOFTWARES AND LIBRARIES USED:**

**Python**: is a high-level, interpreted and general-purpose dynamic programming language that focuses on code readability. The syntax in Python helps the programmers to do coding in fewer steps as compared to Java or C++.  It has a comprehensive and large standard library that has automatic memory management and dynamic features.

**NumPy**: is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

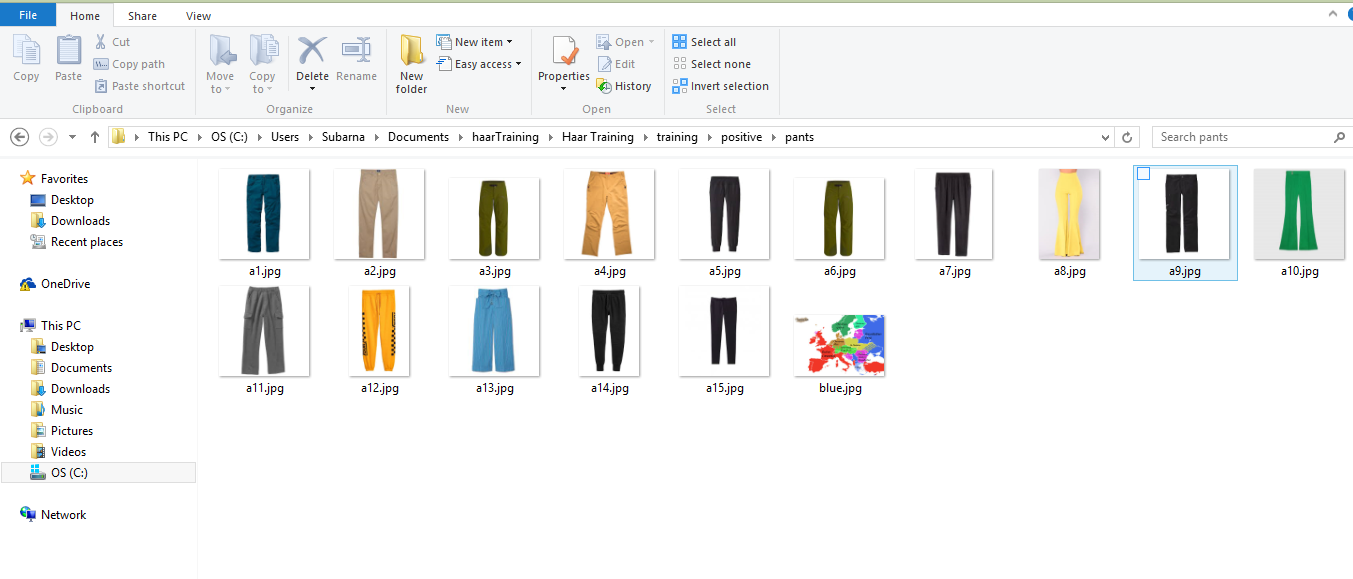
**OpenCV** **(Open Source Computer Vision):** is a [library of programming functions](https://en.wikipedia.org/wiki/Library_(computing)) mainly aimed at real-time [computer vision](https://en.wikipedia.org/wiki/Computer_vision). Originally developed by [Intel](https://en.wikipedia.org/wiki/Intel_Corporation), it was later supported by [Willow Garage](https://en.wikipedia.org/wiki/Willow_Garage) and is now maintained by Itseez. The library is [cross-platform](https://en.wikipedia.org/wiki/Cross-platform) and free for use under the [open-source](https://en.wikipedia.org/wiki/Open-source) [BSD license](https://en.wikipedia.org/wiki/BSD_license).

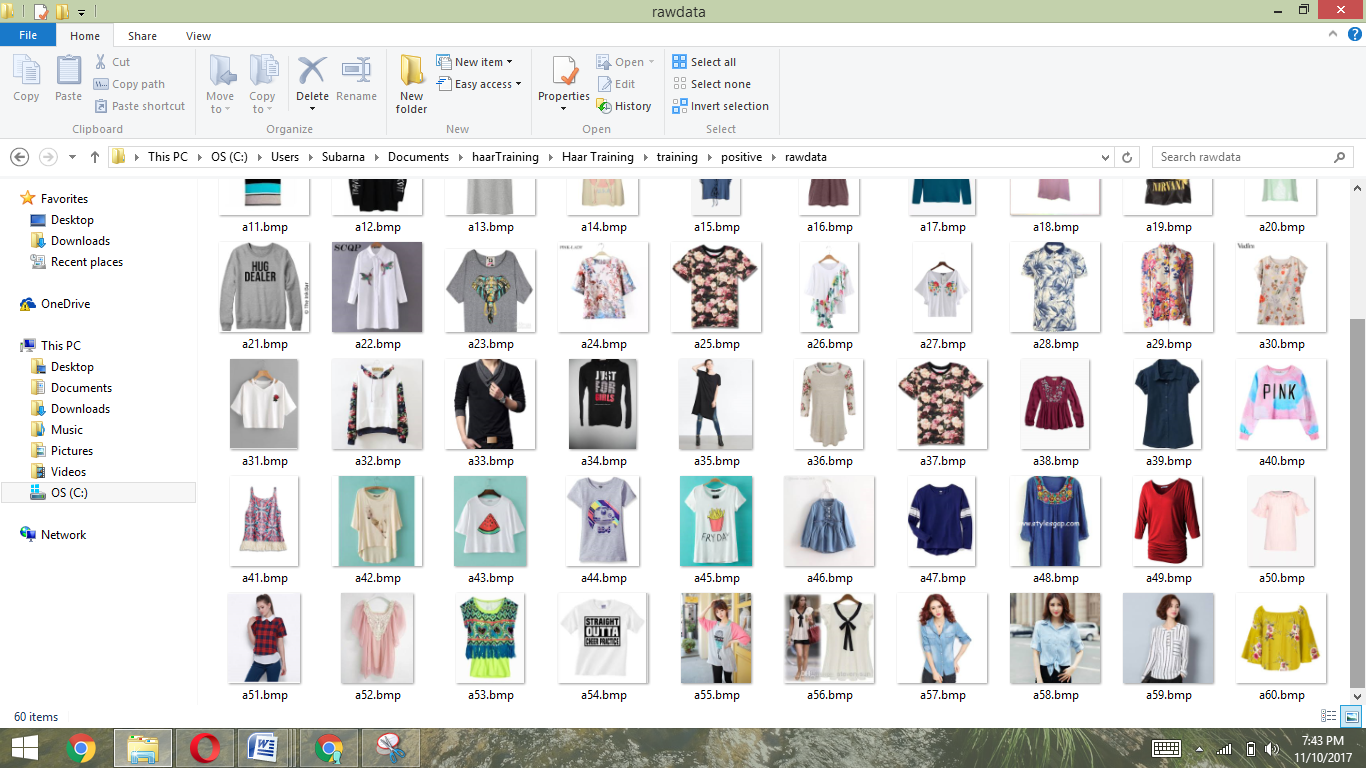
OpenCV supports the [Deep Learning](https://en.wikipedia.org/wiki/Deep_Learning) frameworks [TensorFlow](https://en.wikipedia.org/wiki/TensorFlow" \o "TensorFlow), [Torch](https://en.wikipedia.org/wiki/Torch_(machine_learning))/PyTorch and [Caffe](https://en.wikipedia.org/wiki/Caffe_(software)" \o "Caffe (software)).

**Orange**: is an [open-source](https://en.wikipedia.org/wiki/Open-source_software) [data visualization](https://en.wikipedia.org/wiki/Data_visualization), [machine learning](https://en.wikipedia.org/wiki/Machine_learning) and [data mining](https://en.wikipedia.org/wiki/Data_mining) toolkit. It features a [visual programming](https://en.wikipedia.org/wiki/Visual_programming) front-end for explorative [data analysis](https://en.wikipedia.org/wiki/Data_analysis) and interactive data [visualization](https://en.wikipedia.org/wiki/Information_visualization), and can also be used as a Python library.

**DATA SET:**

A compilation of various pants and shirts/tops.





**CLUSTER FORMATION USING K-MEANS:**

The Algorithm  
*K*-means clustering is a type of unsupervised learning, which is used when you have unlabeled data (i.e., data without defined categories or groups). The goal of this algorithm is to find groups in the data, with the number of groups represented by the variable *K*. The algorithm works iteratively to assign each data point to one of *K* groups based on the features that are provided. Data points are clustered based on feature similarity. The results of the *K*-means clustering algorithm are:

1. The centroids of the *K* clusters, which can be used to label new data

2. Labels for the training data (each data point is assigned to a single cluster)

Rather than defining groups before looking at the data, clustering allows you to find and analyze the groups that have formed organically. The "Choosing K" section below describes how the number of groups can be determined.

Each centroid of a cluster is a collection of feature values which define the resulting groups. Examining the centroid feature weights can be used to qualitatively interpret what kind of group each cluster represents.

**FEATURE EXTRACTION:**

For our project, we decided to take up 4 features for extraction:

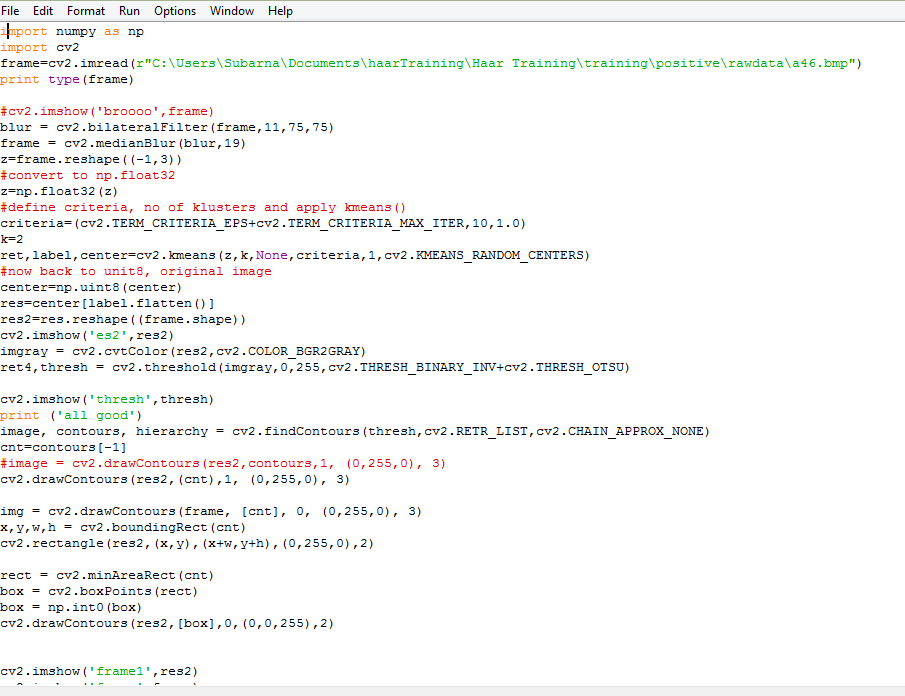
1. Aspect Ratio: The ratio between the breadth and the height of the cloth under inspection.
2. Thresholding: This calculated the area covered by the clothing in a rectangle placed touching the edges of the cloth.
3. Solidity
4. The largest diameter of the circle that can be obtained in that piece of cloth

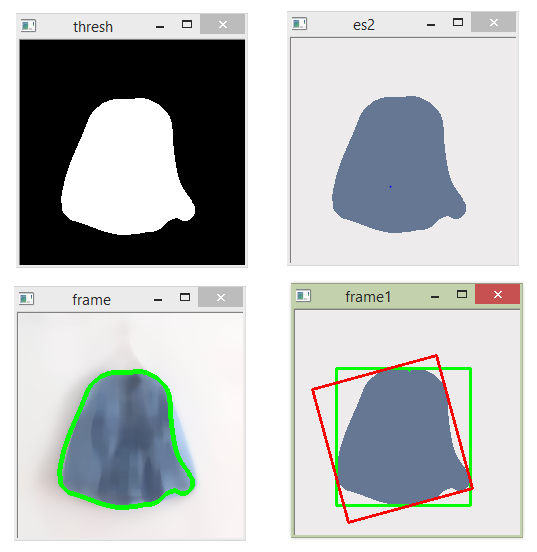
For more clarity, we blurred the cloth a little, in order for the computer to ignore the smaller insignificant patterns and designs on the clothing and not mistake them for multiple entities in the picture.

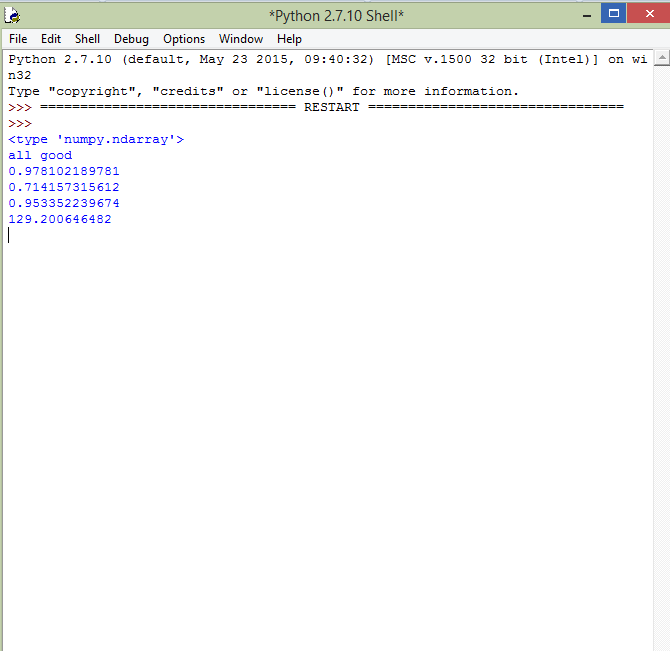
**INPUT:**



**The obtained output:**







**PYTHON CODE FOR FEATURE EXTRACTION:**

import numpy as np

import cv2

frame=cv2.imread(r"C:\Users\Subarna\Documents\haarTraining\Haar Training\training\positive\pants\a4.jpg")

print type(frame)

#cv2.imshow('broooo',frame)

blur = cv2.bilateralFilter(frame,11,75,75)

frame = cv2.medianBlur(blur,19)

z=frame.reshape((-1,3))

#convert to np.float32

z=np.float32(z)

#define criteria, no of klusters and apply kmeans()

criteria=(cv2.TERM\_CRITERIA\_EPS+cv2.TERM\_CRITERIA\_MAX\_ITER,10,1.0)

k=2

ret,label,center=cv2.kmeans(z,k,None,criteria,1,cv2.KMEANS\_RANDOM\_CENTERS)

#now back to unit8, original image

center=np.uint8(center)

res=center[label.flatten()]

res2=res.reshape((frame.shape))

cv2.imshow('es2',res2)

imgray = cv2.cvtColor(res2,cv2.COLOR\_BGR2GRAY)

ret4,thresh = cv2.threshold(imgray,0,255,cv2.THRESH\_BINARY\_INV+cv2.THRESH\_OTSU)

cv2.imshow('thresh',thresh)

print ('all good')

image, contours, hierarchy = cv2.findContours(thresh,cv2.RETR\_LIST,cv2.CHAIN\_APPROX\_NONE)

cnt=contours[-1]

#image = cv2.drawContours(res2,contours,1, (0,255,0), 3)

cv2.drawContours(res2,(cnt),1, (0,255,0), 3)

img = cv2.drawContours(frame, [cnt], 0, (0,255,0), 3)

x,y,w,h = cv2.boundingRect(cnt)

cv2.rectangle(res2,(x,y),(x+w,y+h),(0,255,0),2)

rect = cv2.minAreaRect(cnt)

box = cv2.boxPoints(rect)

box = np.int0(box)

cv2.drawContours(res2,[box],0,(0,0,255),2)

cv2.imshow('frame1',res2)

cv2.imshow('frame',frame)

x,y,w,h = cv2.boundingRect(cnt)

aspect\_ratio = float(w)/h

print (aspect\_ratio)

area = cv2.contourArea(cnt)

x,y,w,h = cv2.boundingRect(cnt)

rect\_area = w\*h

extent = float(area)/rect\_area

print (extent)

area = cv2.contourArea(cnt)

hull = cv2.convexHull(cnt)

hull\_area = cv2.contourArea(hull)

solidity = float(area)/hull\_area

print (solidity)

area = cv2.contourArea(cnt)

equi\_diameter = np.sqrt(4\*area/np.pi)

print(equi\_diameter)

cv2.waitKey(0)

cv2.destroyAllWindows()

**PREDICTION ALGORITHM; CLASSIFIER:**

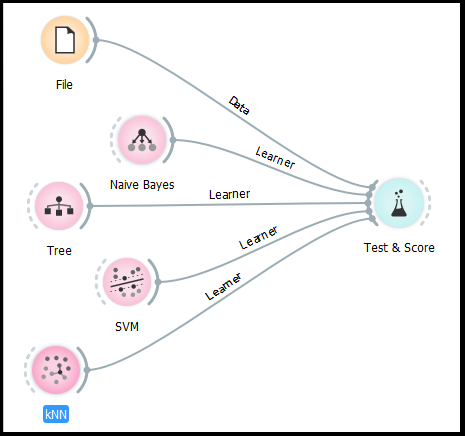
**Orange Software:**

We used the Orange software, to figure out the best prediction method/classifier for our data set.This required for the data set of the extracted features to be going through a series of different types of classifiers in-built in the software.

Henceforth, it underwent through KNN, Tree, SVM, GaussianNB classifiers.

Now, observing through a set of different parameters (mainly Accuracy), gaussianNB was chosen as the appropriate predictor algorithm for our project.

**SET OF PARAMETER DATA THROUGH THE FOLLOWING:**



Naive Bayes algorithm?

It is a classification technique based on [Bayes’ Theorem](https://en.wikipedia.org/wiki/Bayes%27_theorem) with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. For example, a fruit may be considered to be an apple if it is red, round, and about 3 inches in diameter. Even if these features depend on each other or upon the existence of the other features, all of these properties independently contribute to the probability that this fruit is an apple and that is why it is known as ‘Naive’.

Naive Bayes model is easy to build and particularly useful for very large data sets. Along with simplicity, Naive Bayes is known to outperform even highly sophisticated classification methods.

**Scikit learn (python library)** will help here to build a Naive Bayes model in Python.

**GAUSSIAN NAÏVE BAYES:**

It is used in classification and it assumes that features follow a normal distribution.

This prediction method undergoes the training through the input of collection of possible inputs and their expected outputs.

**PROJECT REQUIREMENT:**

For our project, we chose to take 14 training sets and 1 corresponding output for each.

So, 7 sets of features corresponded to shirts and the rest of them to pants.

**PREDICTION CODE:**

from sklearn.naive\_bayes import GaussianNB

import numpy as np

#assigning predictor and target variables

x= np.array([[0.810572687,0.329259242,0.355554694,132.3262137],

             [1.018691589,0.64848238,0.878796241,196.2633841],

             [0.852040816,0.673377734,0.840473603,167.5215592],

             [0.926108374,0.66560109,0.855747204,179.8411269],

             [1.0,0.240423093,0.242593945,123.3809223],

             [0.857142857,0.810090029,0.946558825,190.8729324],

             [0.959537572,0.750835713,0.829502395,165.6931733],

             [0.485148515,0.700772883,0.867112542,132.9022783],

             [0.417085427,0.892656051,0.925607383,137.0132981],

             [0.434210526,0.816099592,0.90810944,153.1481167],

             [0.388059701,0.855083557,0.905504897,130.6485719],

             [0.413612565,0.794552323,0.868233335,123.5510781],

             [0.388059701,0.855083557,0.905504897,130.6485719],

             [0.439393939,0.862939742,0.909619386,137.5743647]])

y = np.array([2,2,2,2,2,2,2,4,4,4,4,4,4,4])

#Create a Gaussian Classifier

model = GaussianNB()

# Train the model using the training sets

model.fit(x, y)

print ('all good')

#Predict Output

predicted= model.predict([0.967105263158,0.605375044755,0.751242675849,131.234426512])

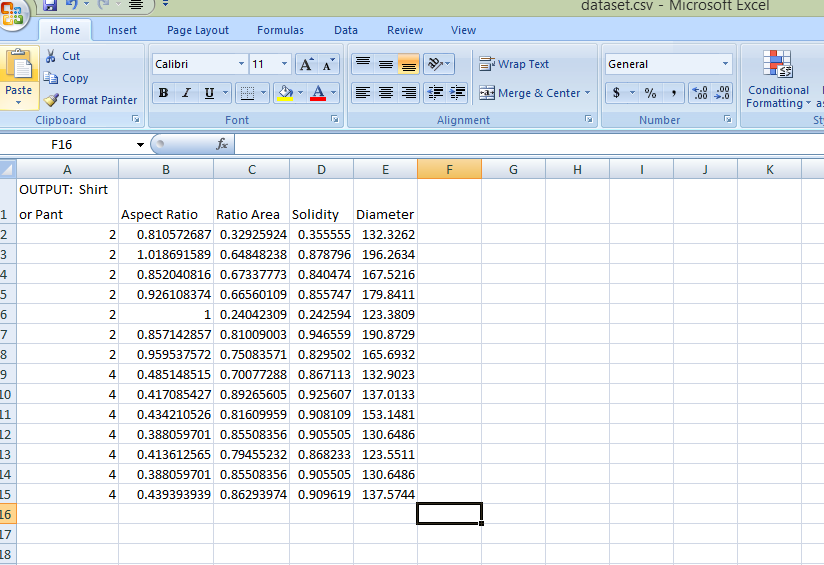
if predicted==2:

    print ('shirt')

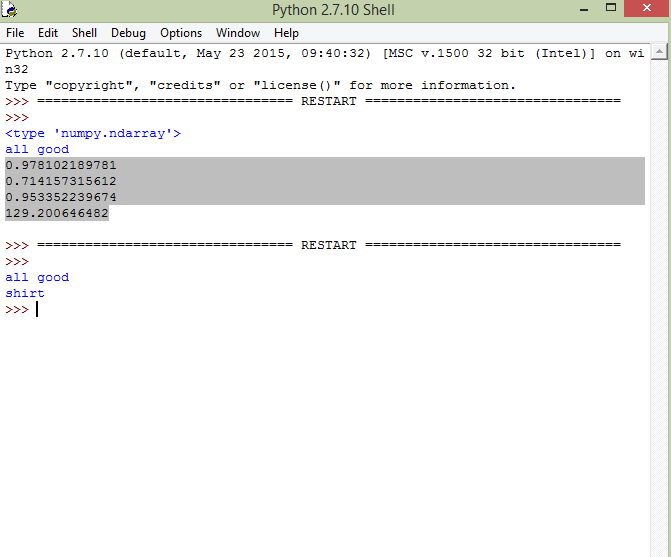
elif predicted==4:

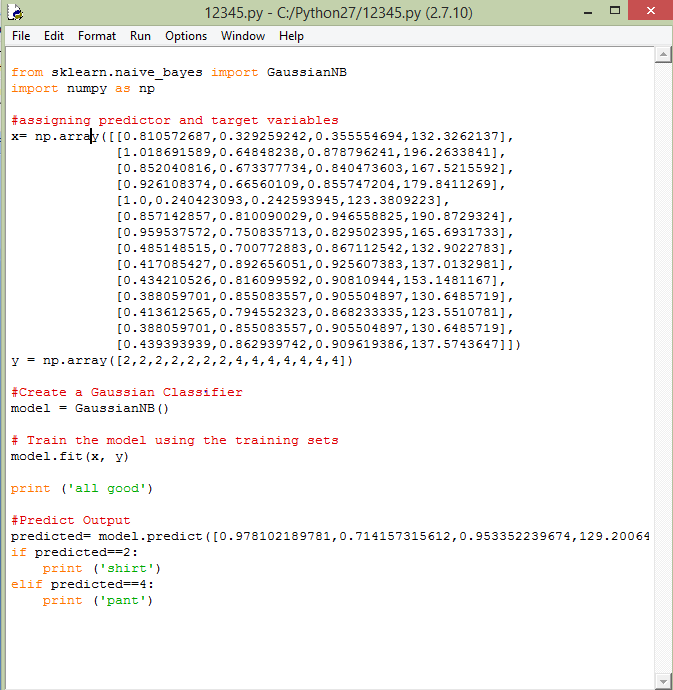
    print ('pant')

**THE EXCEL SHEET CONTAINING TRAINING FEATURE DATA SET:**



OUTPUT OBTAINED:





**INFERENCES:**

1. https://home.deib.polimi.it/matteucc/Clustering/tutorial\_html/kmeans.html

2.

[https://www.analyticsvidhya.com/blog/2017/09/naive-d](https://www.analyticsvidhya.com/blog/2017/09/naive-d3. https://en.wikipedia.org/wiki/Naive_Bayes_classifierayes-explained/)

[3. https://en.wikipedia.org/](https://www.analyticsvidhya.com/blog/2017/09/naive-d3. https://en.wikipedia.org/wiki/Naive_Bayes_classifierayes-explained/)