ML report 2

TODO:

I.          K-means Problem

You will get a dataset (**data\_noah.csv**). It is Noah Syndergaard’s pitches that have been tracked by the PITCHf/x system in the MLB Regular Season.

You have to do the following:

1.            Dataset including 1321 number of instances with many attributes.

**2.**            ***Don’t*use the library related to K-means.**(i.e. Construct a K-means function by yourself).

3.            Use **Attribute x** (horizontal movement)and**y**(vertical movement) to partition 1322 pitches into 3 clusters.

4.            3 clusters will represent FF (four-seam fastball), CH (changeup) and CU (curveball).

**5.**            **Construct a cost function to check the accuracy of pitch types.**

6.            **Generate a figure** to show the result of K-Means clustering.

7.            Try to use another two or more attributes (like speed) to partition.

8.            Try to explain why k = 3 is the best, and write in your report.

9.            Show your **code**, **accuracy**, the reason of k = 3 and the result of K-Means clustering **(figure) in your report**.

II.        Kd-tree Problem

You will get a set of points (**points.txt**) in the unit square (all points have x-coordinates and y-coordinates). You have to build a 2d-tree.

You have to do the following:

**1.**            **You *can* use the library related to Kd-tree.**

**2.**            **Draw a 2d-tree divides the unit square (Use two colors).**

3.            Show your **code** and the result of 2d-tree**(figure) in your report**.

The report should contain the following:

1. 1. What environments the members are using (5%)
2. 2. K-means code (30%)
3. 3. Cost function and accuracy (15%)
4. 4. The result of K-Means clustering (10%)
5. 5. Use another two or more attributes to partition and the reason of k = 3 (10%)
6. 6. Kd-tree code (15%)
7. 7. The result of Kd-tree (15%)

Introduction to Machine Learning Homework 2

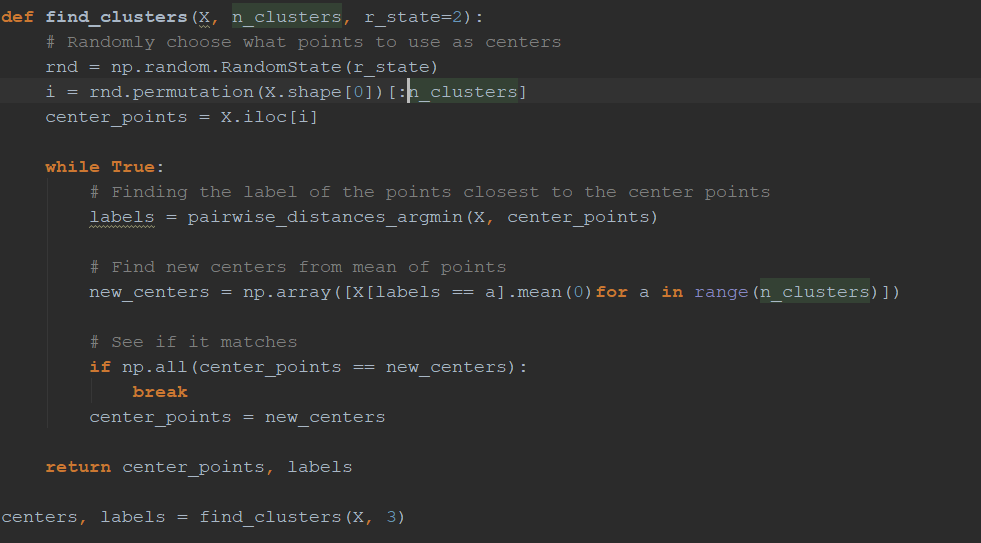
# Environment used

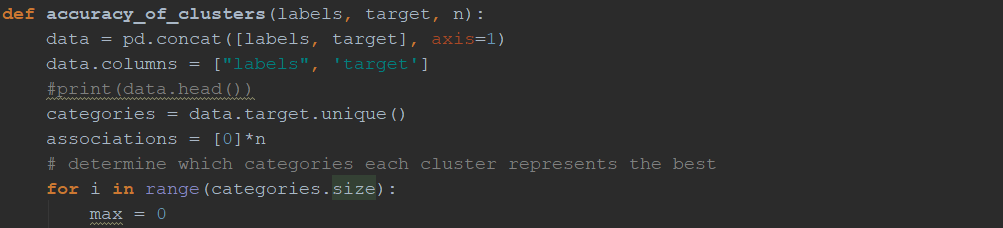
For this homework we used pyCharm IDE and GitHub as a remote repository.

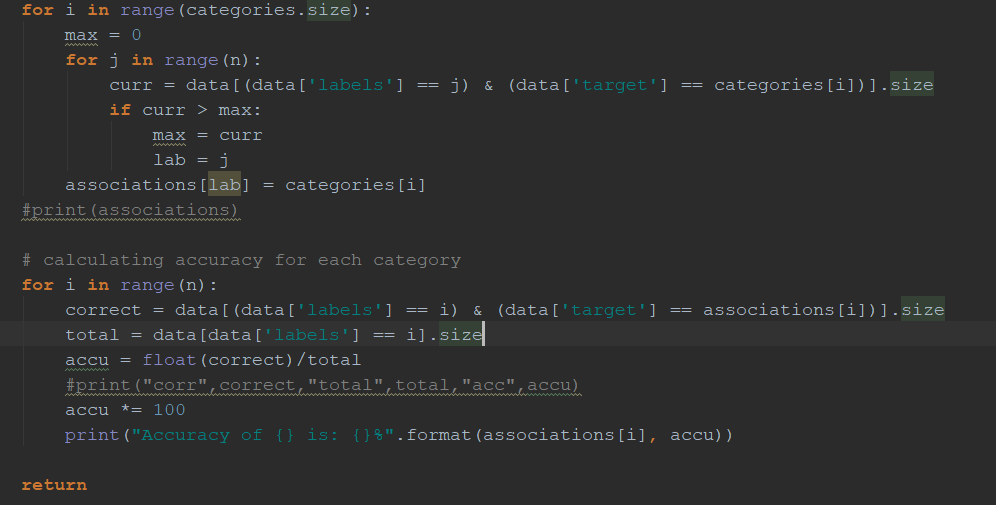
# K-means

K-means Code

The function find\_clusters finds a center point for the number of clusters specified and returns it as center\_points. It also returns a numpy array with the estimated labels:



The function for measuring the clusters accuracy:



K-means result

The results we got from using our K-means clustering function with k=3 and accuracy function was:

Total accuracy: 80.01514004542014%  
Accuracy of CH: 37.971698113207545%  
Accuracy of CU: 100.0%  
Accuracy of FF: 99.83221476510067%

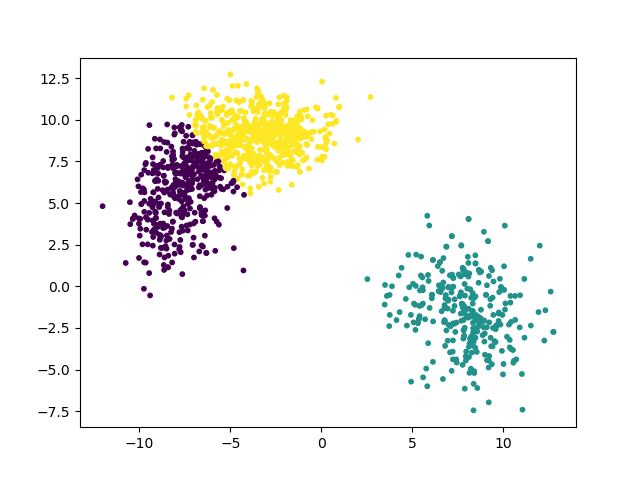
With a cluster that looks like this:

Figure 1, purple is FF, yellow is CH, turquoise is CU

Results with other parameters

For our second experiment we used ‘speed’ and ‘sz\_bot’ to partition the data with good results:

Total accuracy: 99.47009841029522%  
Accuracy of FF: 100.0%  
Accuracy of CH: 95.85798816568047%  
Accuracy of CU: 100.0%

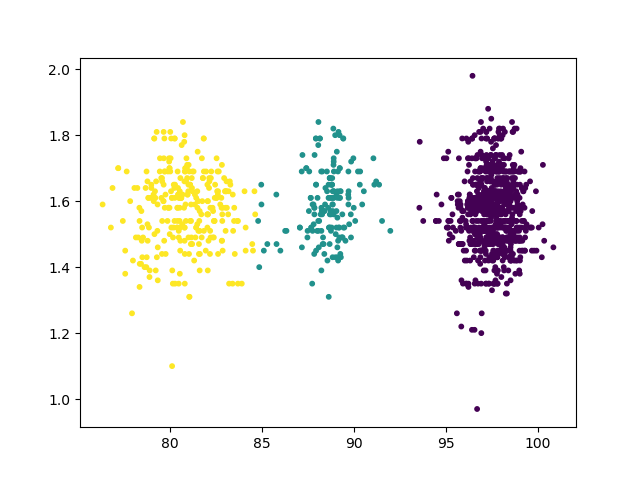
With a cluster that looks like this:

Figure 2, yellow is CU, turquoise is CH, purple is FF

Figure 3, purple is FF, yellow is CH and turquoise is CU

Why does k=3 work best?

Since we have 3 different targets the data forms 3 nice clusters. Therefore k=3 works best in this case.

# Kd-tree

Kd-tree Code

To construct Kd-tree we used Node class:

class Node:  
 def \_\_init\_\_(self, value, left\_child, right\_child):  
 self.value = value  
 self.left\_child = left\_child  
 self.right\_child = right\_child

Construction algorithm:

def kdtree(points, axis=0):  
  
 if len(points) == 0: #Recursion ending condition  
 return None  
  
 points.sort(key=itemgetter(axis))  
 median = len(points) // 2  
  
 return Node(points[median], kdtree(points[:median], 1 - axis), kdtree(points[median + 1:], 1 - axis))

Axis selection is might be done before construction:

x\_variance = np.var([point[0] for point in points])  
y\_variance = np.var([point[1] for point in points])  
  
axis = 0 if x\_variance >= y\_variance else 1

Result of Kd-tree

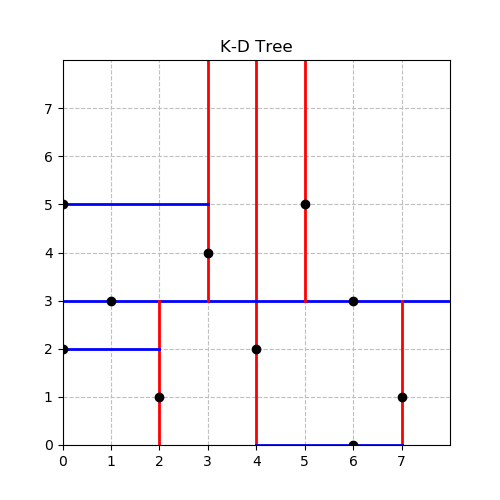


Figure 4 Result of Kd-tree on points.txt data