Machine Learning Algorithm Intro (PYTHON)

Install TensorFlow + Keras

- **create environment:**

```
  verify
(base) Haos-MacBook-Pro-2:~ lamtuhao$ conda create -n tensor_envir python=3.6
Collecting package metadata (current_repodata.json): done
Solving environment: done
```

Name environment: tensor_envir

- **activate/deactivate environment**

```
#
# To activate this environment, use
#
#     $ conda activate tensor_envir
#
# To deactivate an active environment, use
#
#     $ conda deactivate
```

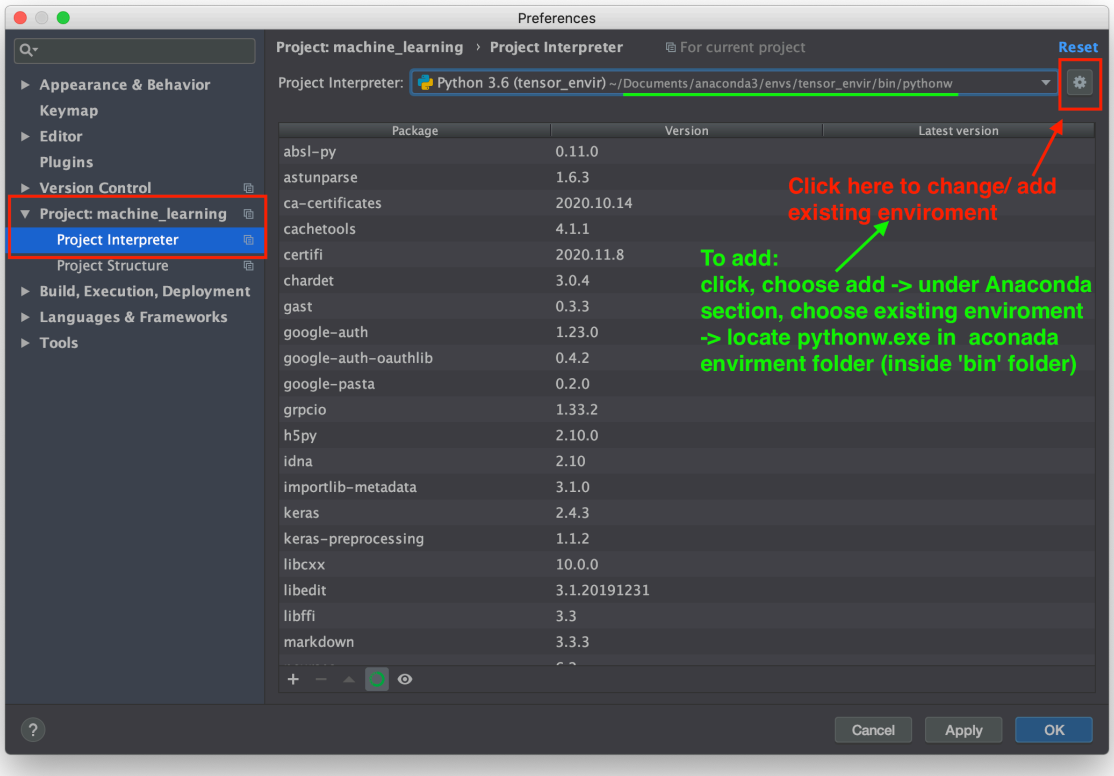
- **install tensorflow + install keras**
  (in terminal - after activate environment)
  pip install tensorflow
  pip install keras

Set Up Project interpreter in Pycharm

- *Reference link:*
https://www.jetbrains.com/help/pycharm/configuring-python-interpreter.html#add_new_project_interpreter
- *Use terminal to get interpreter path (FAST WAY):*
https://docs.anaconda.com/anaconda/user-guide/tasks/integration/python-path/
- *Cannot find pythonw.exe:*
https://stackoverflow.com/questions/43480073/anaconda-python-3-6-pythonw-and-python-supposed-to-be-equivalent
+ install python.app : terminal -> activate environment -> conda install python.app

- **Steps:**

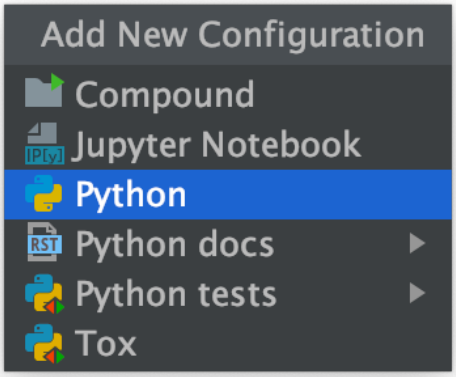
- create project
- Pycharm-> prerferences -> project name -> project interpreter

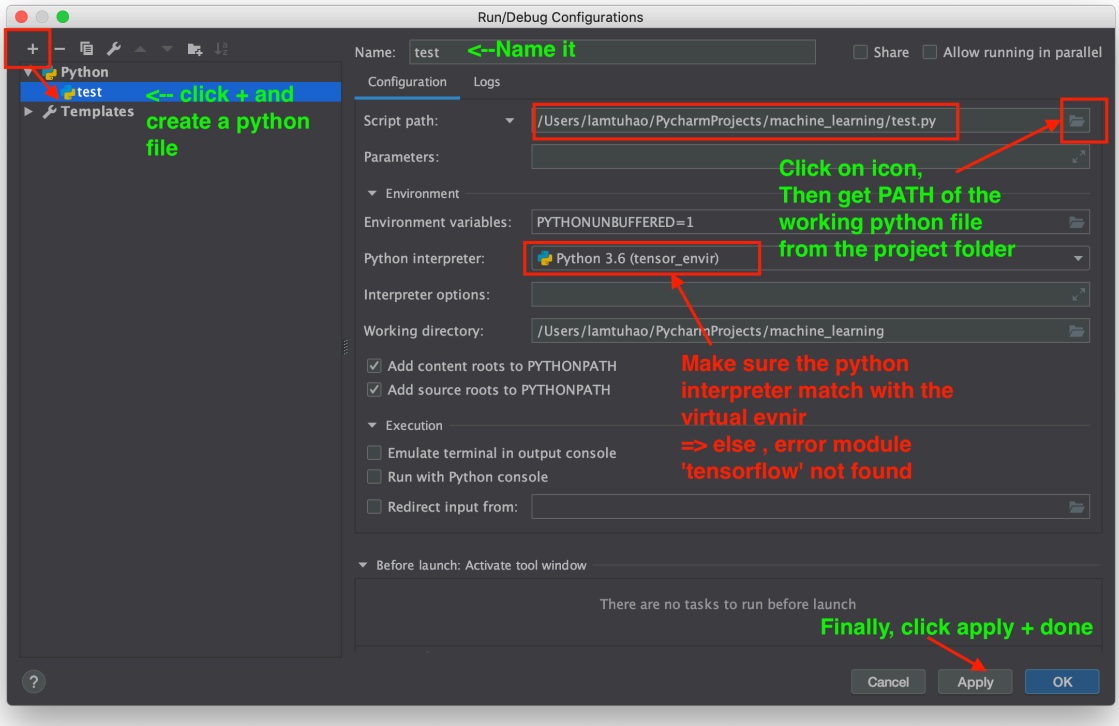


- **OR fast way to get Path :**

```
For more details, please visit https://support.apple.com/kb/HT208050.
(base) Haos-MacBook-Pro-2:~ lamtuhao$ conda activate tensor_envir
(tensor_envir) Haos-MacBook-Pro-2:~ lamtuhao$ which python
/Users/lamtuhao/Documents/anaconda3/envs/tensor_envir/bin/python  <-- PATH
```

Add configuration





To check if running ok:

- In the python file, type:

```
import tensorflow
import keras
```

=> Then run to see if there is error appear

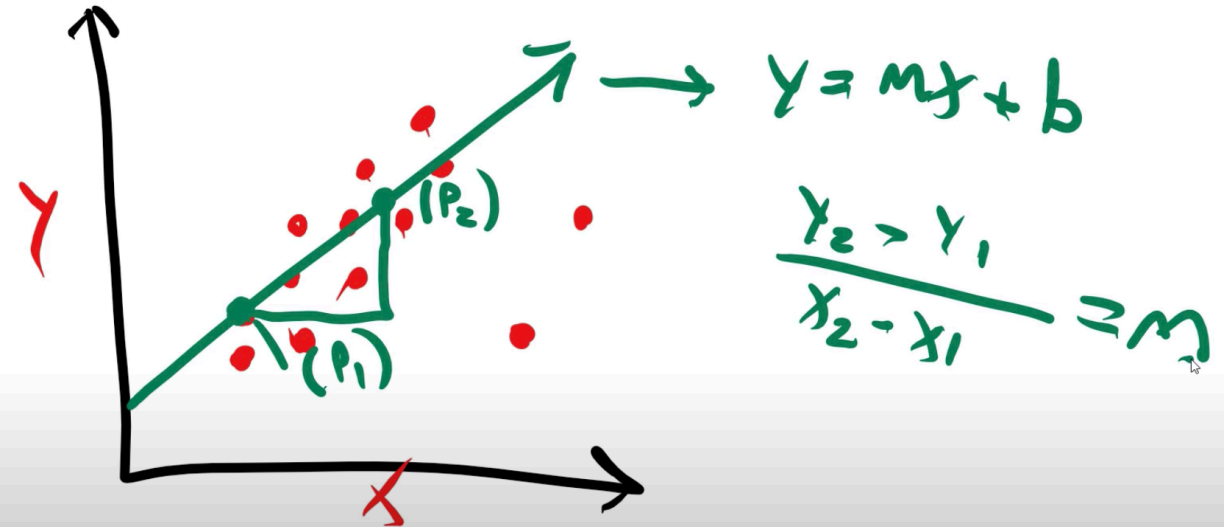
Linear Regression:

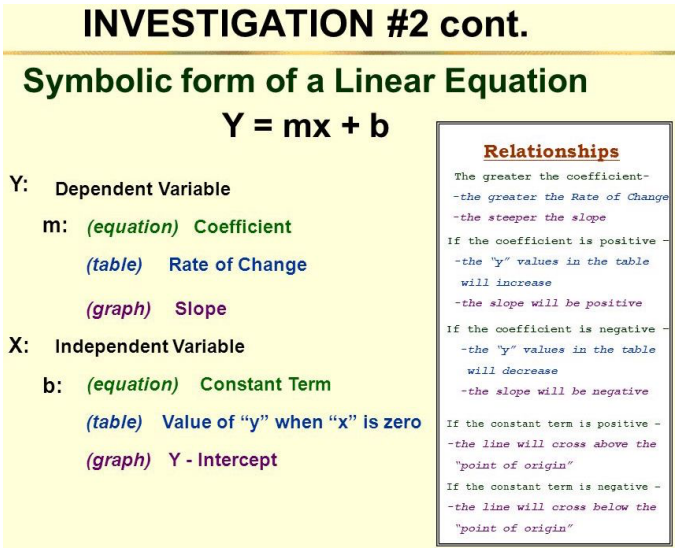
- Finding the best-fit-line => show us a linear prediction
- y = mx + b
- In 2-dimension space, a line y=mx+b need only 1 coefficient. In multiple

dimensional space, a line has more than 1 coefficients. EX: a line in 5D need 5 coefficients (5 m's) => Therefore, base on the data set and its attributes, we would have as many as variables and coefficients regarding to that from the line we get.

- The bigger the value of one coefficient has than the others, the more weigh it could put affect on the equation

EX: y = mx + nz. => 2 coefficients : m,n => If m > n, input x of m likely to to give bigger affect on how the line would be like on the graph => In AI, it would affect the prediction results

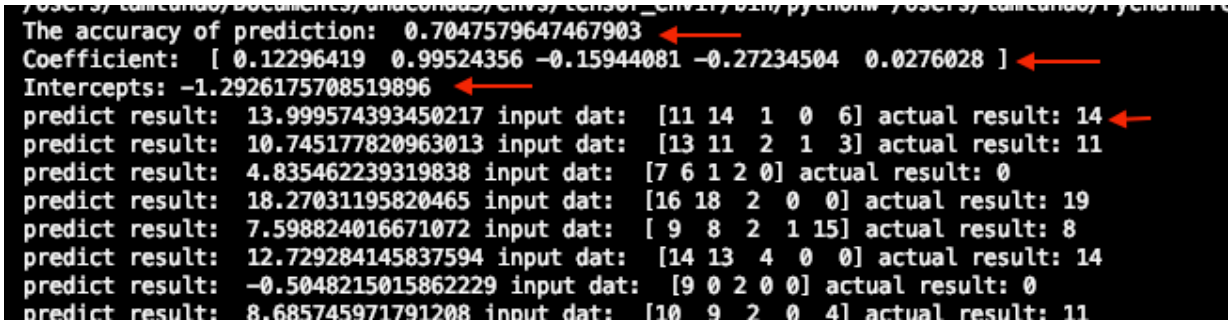




- ***Data set for project:*** **https://archive.ics.uci.edu/ml/datasets/Student+Performance**
  - download Data Folder zip file -> Unzip -> Drag **...-mat.cvs** file to the project
- **package needed:**

- sklearn
- pandas -> allow read data sets easily
- numpy -> allow arrays. Python only have List, we need array

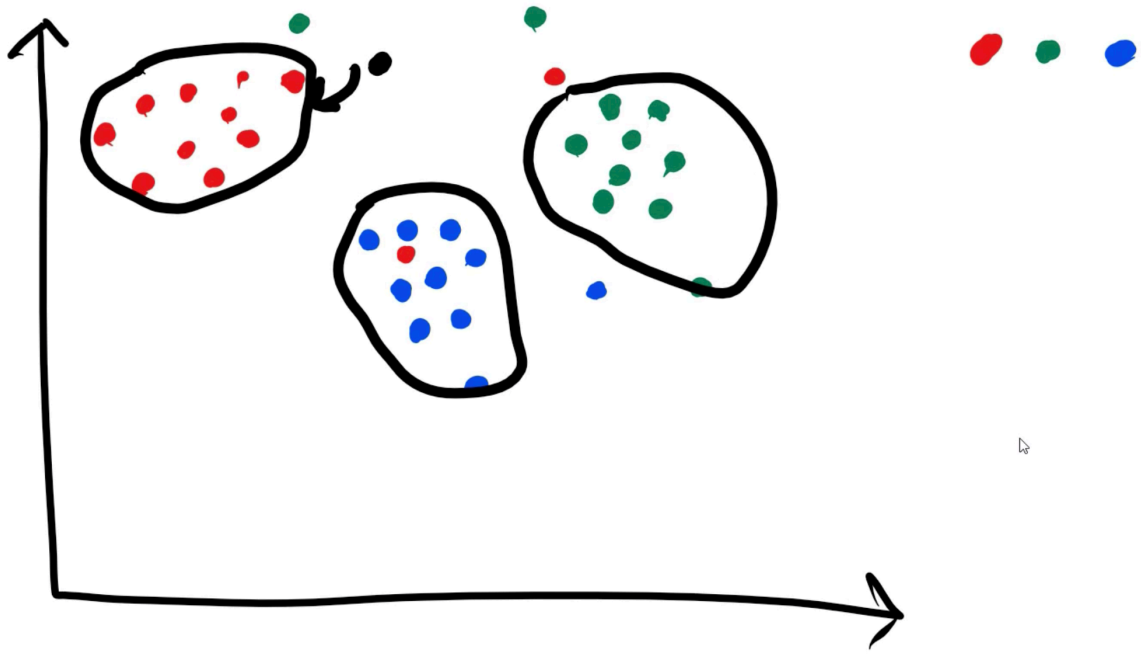
- **Output from machine training:**



=> we could see that the prediction is pretty close to the actual result. For example, in 1st line: machine predict : 13.999 ..~.. actual result: 14

K-Nearest Neighbor:

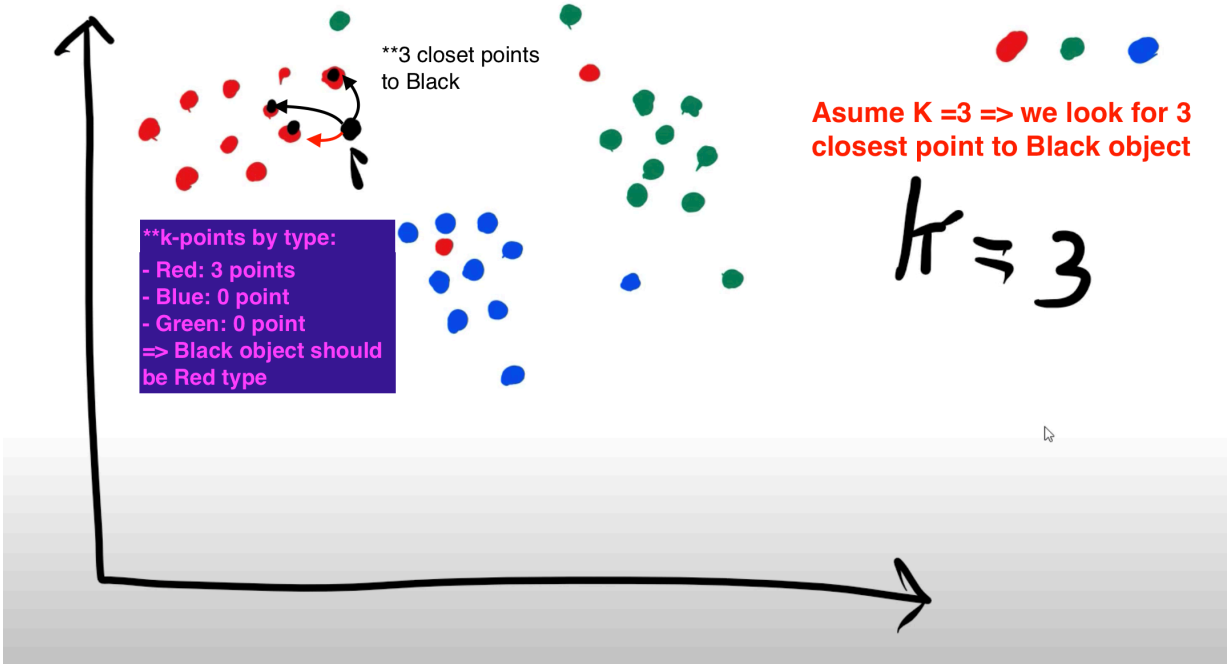
- Based on the data inputs, machine classifying object to specific type a,b,c,...
  Ex: Car
  -> inputs: price, comfort, technology, maintenance, safety.
  -> machine evaluate, then specify the car into 1 of 4 types : T1 - unqualified, T2 -qualified, T3-good, T4-very good
- Demonstration:
  + Assume we have 3 types represent in 3 different colors Red, Green and Blue.
  + We have an undefined-type Black object that need to be classified.
  => So, will the Black object be type Red, Blue or Green?
  => As normal human logic, as Black obj is close to group Red, we could assume Black belong to type Red.
  => K-Nearest neighbor algorithm.



- 'k' represent the amount of neighbors we're going to look for. In this demonstration, k is a number of points that close to Black object. Once we get all the k-points, the machine will see the majority of k-points lying on which type. For example, if k-points type Red has the highest occurrence, then we indicate Black object should be in Red type as an appropriate classification.
- We should pick 'k' to be a odd number instead of even value, so that we always has the final winner!

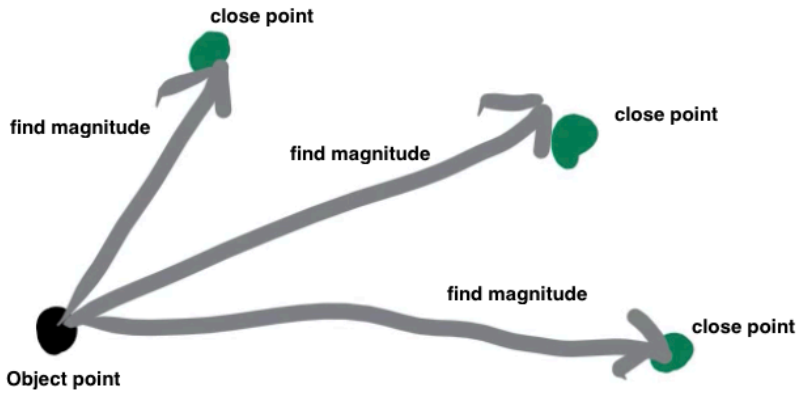
EX: if k =2 => we can have 2-2 for each type Red & Blue => we cannot specify Black object to be Red or Blue as they have the same occurrence

EX: if k= 5 => we can have 2-3 for (Red-Blue) => we can clearly classify Black object to be Blue type as Blue type has higher occurrence (3>2)

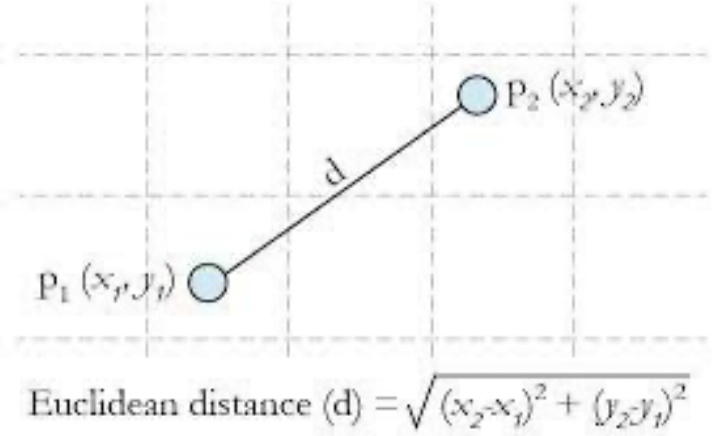


- **How machine find the closest points to the object point?**

+ The machine will a draw a line from the object point to every data points. Then it will calculate each magnitude (m)



+ The machine can calculate the magnitude (m) by the Euclidean distance:



+ The Euclidean distance is applied to mult-dimension point:

EX: P1 (x1,y1,z1) and P2 (x2,y2,z2) => d(P1,P2) = sqrt((x1-x2)^2 + (y1-y2)^ + (z1-z2)^2)

$$d(\mathbf{p},\mathbf{q}) = d(\mathbf{q},\mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}$$

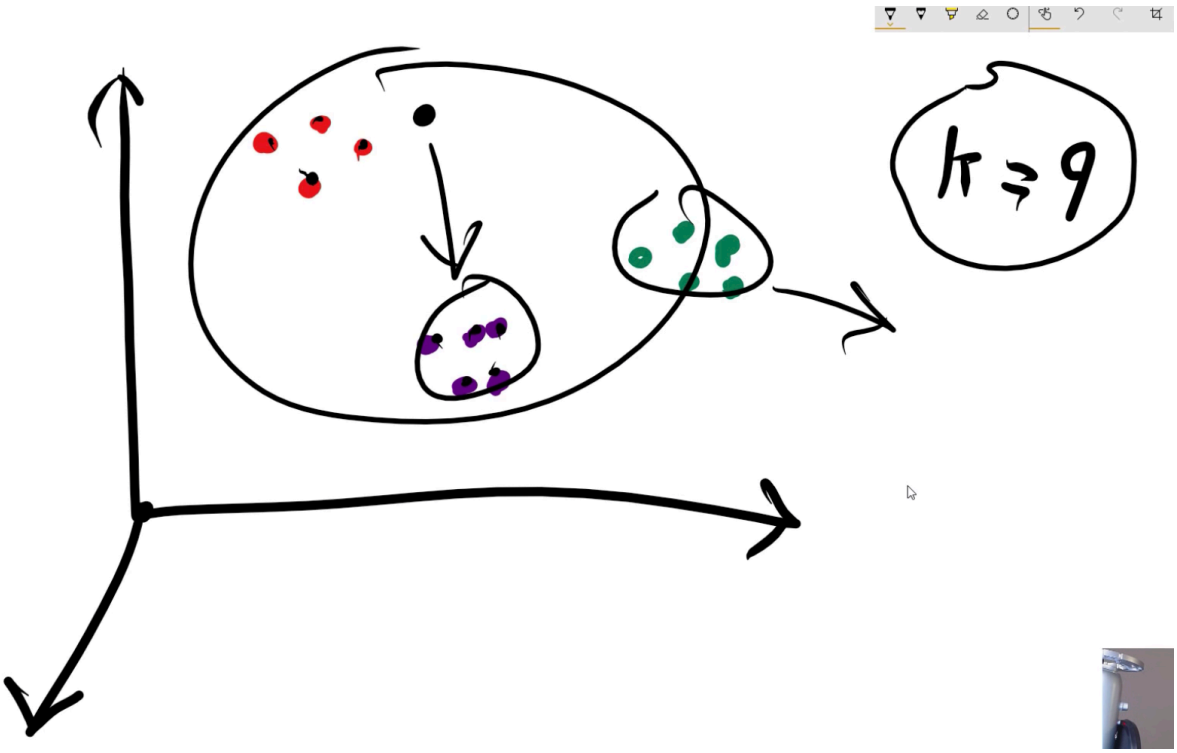
$$= \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}.$$

– Error for picking large k:

+ In this demonstration, we pick k=9. Assume 9 k-points we have are the Red and Purple types which are close to the Black object.

+ Since the occurrence of Purple type is higher than Red type (5 > 4), we conclude Black object is Purple type

+ However, from the graph, we could clearly see that Black object is closer to group Red type than group Purple => Our classification above is wrong due to the large value pick of k!



- In this type of problem using the K-nearest neighbor algorithm, ***we should not save back the data or train it before hand because it would cost machine a lot of time*** saving all the points and computing all the distances between 2 points.

  => Therefore, the best approach is to have a function where we could have the machine run with the data rather than pre-training

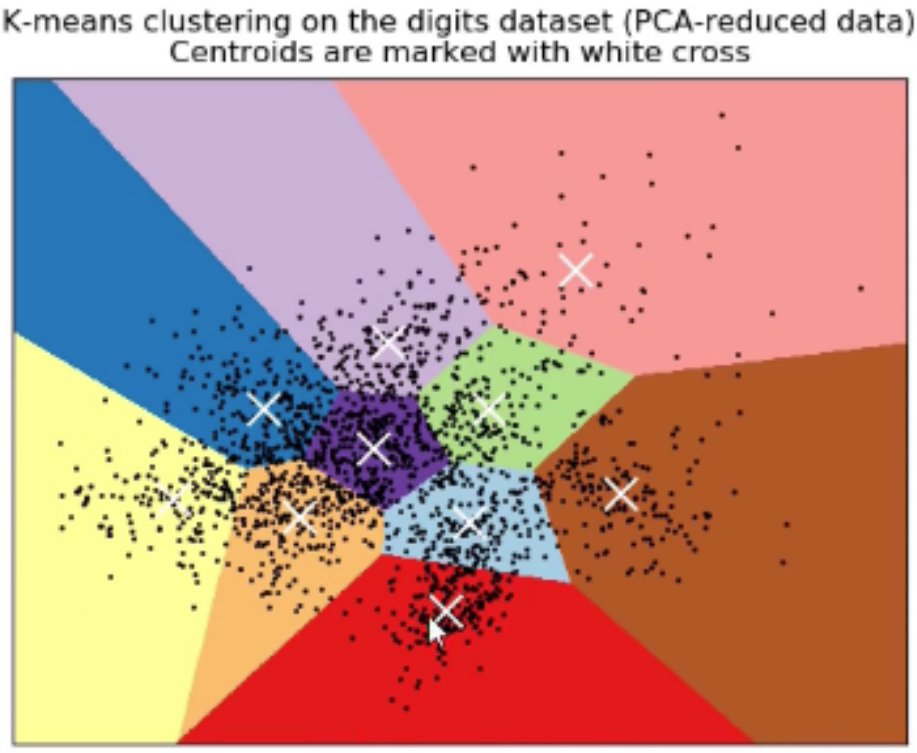
- **Output from machine training:**

```
/Users/lamtuhao/Documents/anaconda3/envs/tensor_envir/bin/pythonw /Users/lamtuhao/Pycharm
The accuracy of prediction:  0.9190751445086706
predicted result:  2 input dat:  (2, 2, 1, 2, 1, 1) actual result: good
predicted result:  2 input dat:  (1, 2, 0, 2, 1, 1) actual result: good
predicted result:  0 input dat:  (0, 1, 3, 2, 1, 2) actual result: unacc
predicted result:  2 input dat:  (0, 3, 0, 0, 2, 1) actual result: good
predicted result:  1 input dat:  (1, 2, 1, 1, 2, 0) actual result: acc
predicted result:  0 input dat:  (0, 1, 1, 1, 2, 0) actual result: unacc
predicted result:  2 input dat:  (2, 2, 3, 0, 0, 2) actual result: good
predicted result:  2 input dat:  (2, 3, 3, 0, 0, 1) actual result: good
predicted result:  2 input dat:  (1, 0, 1, 1, 1, 1) actual result: good
predicted result:  2 input dat:  (0, 3, 3, 1, 2, 1) actual result: good
predicted result:  0 input dat:  (1, 3, 3, 2, 2, 0) actual result: unacc
```

Support Vector Machines (SVM)

Reference link: https://towardsdatascience.com/svm-and-kernel-svm-fed02bef1200

K Means Clustering



sklearn KMeans demo : https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_digits.html#sphx-glr-auto-examples-cluster-plot-kmeans-digits-py
sklearn KMeans doc: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html