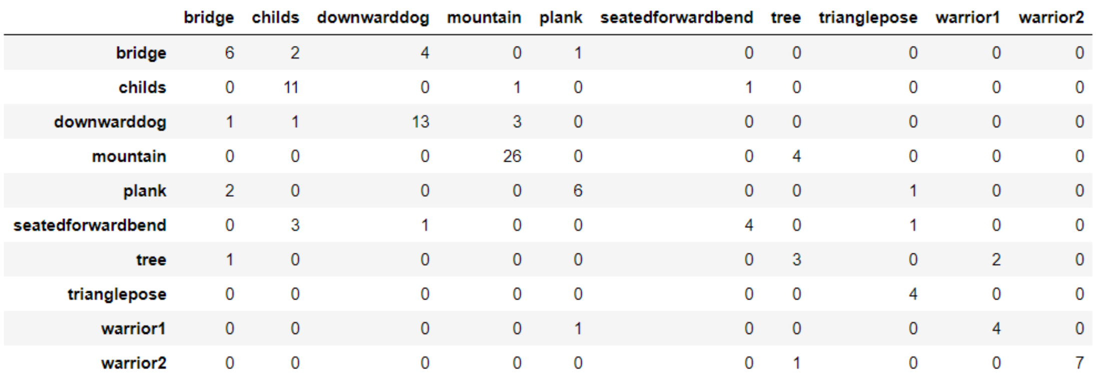
Question 1:

	Precision	Recall	F-score
Macro	0.7094281045751635	0.7182692307692308	0.7138212929932494
Micro	0.7304347826086957	0.7304347826086957	0.7304347826086957



	bridge	childs	downwarddog	mountain	plank	seatedforwardbend	tree	trianglepose	warrior1	warrior2
bridge	6	2	4	0	1	0	0	0	0	0
childs	0	11	0	1	0	1	0	0	0	0
downwarddog	1	1	13	3	0	0	0	0	0	0
mountain	0	0	0	26	0	0	4	0	0	0
plank	2	0	0	0	6	0	0	1	0	0
seatedforwardbend	0	3	1	0	0	4	0	1	0	0
tree	1	0	0	0	0	0	3	0	2	0
trianglepose	0	0	0	0	0	0	0	4	0	0
warrior1	0	0	0	0	1	0	0	0	4	0
warrior2	0	0	0	0	0	0	1	0	0	7

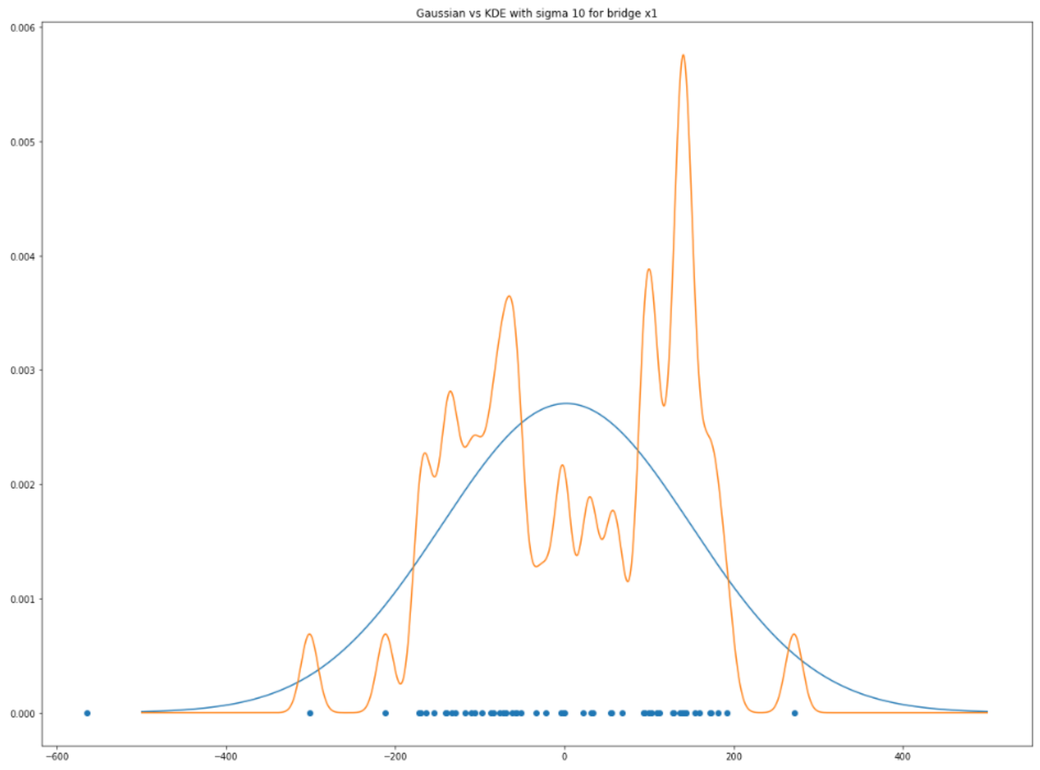
Confusion Matrix

Macro-averaging calculates metric of each class and the take average whereas a micro-average will aggregate the contributions of all classes to compute the average metric. Micro-averaged precision and micro-averaged recall are both equal to the accuracy when each data point is assigned to exactly one class. They are all calculated by using number of total true positive divided by the number of total instances (=sum of all TP+FP=sum of all TP+FN). For example, number of FN for class 'bridge' in the first row is 7, followed by 2, 5……If we add all TP and FP which would be the whole sample size, so as TP plus FN, the denominators would be the number of TP.
If the data is balanced in each class, macro-averaging would make little difference with micro-averaging. However, if the class is imbalanced, micro-averaging is preferable if you want to weight the metric towards largest ones since macro-averaging would treat each class' metrics 'equally' which class imbalance would not be captured. In the dataset, micro-averaging metrics are higher since classes with most instances have higher-valued metrics where class 'mountain' has precision of 0.867 which is the highest precision.
If you want to weight the metric towards the smaller ones, use macro-averaging metrics. From the dataset, macro metrics are smaller than micro metrics, which means smaller classes tend to perform worse. The worst precision is 0.375 of class 'tree' with only 6 instances. The reason why micro-averaging is more commonly used is because in most tasks, we would be interested in simply maximizing the number of correct predictions. In these situations, no class is more important than the others.

Question 3:

The most obvious change is that the accuracy improved. For bandwidth equals 10, accuracy improved to 0.7739 which increased by 0.0435. This shows that the dataset may not have Gaussian distributions independently. Gaussian Naïve Bayes Classifiers assumes each feature has a normal distribution conditioned on each class independently. However, KDE weights a defined density (Gaussian in this dataset) around each observation equally. Kernel density estimation doesn't assume Gaussian distribution of the features which is an advantage because it can be widely used among data with different types of distributions and it can model more complicated patterns. If the probability distribution for a variable is complex or unknown, it can be a good idea to use KDE to approximate the distribution directly from the data samples. Take feature ‘x1’ from class ‘bridge’ as an example, the data is not exactly Gaussian distributed since the data becomes less dense centered around mean. As we can see from KDE, these part forms lower density. It could be inferred that this dataset is not a perfect normal distribution.

Moreover, KDE Naïve Bayes runs for longer time compared to Gaussian Naïve Bayes because KDE need to sum all the normal distribution centered around each data while Gaussian only calculate density based on each feature. KDE requires more running time. Also, KDE requires more hyperparameters like kernel bandwidth. If we use Gaussian for KDE then sigma could lead to very different accuracy. KDE has more hyperparameters to be turned when training.


Gaussian vs KDE with sigma 10 for bridge x1