**Question 1**

The two methods implemented in Question 1 are micro-averaging and macro-averaging. For micro-averaging, precision and recall are calculated by making the sum of TP, FP, FN of each class and compute the formation.

For macro-averaging, we calculate precision and recall for each class and take the mean.

F-score is calculated by taking the value of beta as 1.

The difference between micro and macro averaging is that micro-averaging weights each sample equally while macro-averaging weights each class equally.

|  |  |  |
| --- | --- | --- |
|  | Micro-averaging | Macro-averaging |
| Precision | 0.75 | 0.719 |
| Recall | 0.75 | 0.736 |
| F1-score | 0.75 | 0.727 |

According to the table above, macro-averaging gives worse results than micro-averaging. The reason behind this is because classes other than “Mountain” have smaller proportion and their results average down the score in micro-averaging. For instance, “Tree”, “Trianglepose” and “Warrior1” have only 6, 4, and 5 instances in the test dataset, which are much smaller than other classes such as “Mountain” which has 30 instances. Inversely, ‘Mountain’ class have relatively larger proportion and its score has averaged up the overall result in micro-averaging. Thus, micro-averaging evaluation will be preferred over micro-averaging as the dataset is slightly imbalanced as we have class imbalance in this assignment.

**Question 2**

In order to see the distribution of the dataset, we grouped every feature with missing values discarded and have selected 3 classes (Mountain, Downwarddog, Childs) to draw their QQ-plots. The QQ-plot will form a straight line if the data is normally distributed. Please note that the dataset we used is from *all.csv*, which combined all data from both *train.csv* and *test.csv*.

Chart, line chart

Description automatically generated

Figure 2.1: Mountain

Chart, line chart

Description automatically generated

Figure 2.2 Downwarddog

Chart, scatter chart

Description automatically generatedFigure 2.3 Childs

According to the plots above, there is a bunch of features having right or left skewed distribution in the dataset and the distribution of each feature is not always consistent for every class. For instance, even though *x1* in Mountain class forms a straight line in QQ-plot, it forms 2 peaks (which is also known as a bimodal distribution) in Childs class. This applied to *x7* as well. Also, all the plots above did not show a certain straight line, some of them are either right-skewed, left skewed, light tailed or heavy tailed. The reason behind this may be the size of dataset for each feature in class is not large enough to generalise a Gaussian distribution. Therefore, the assumption that the numeric data comes from a Gaussian distribution is not always true in this dataset.

**Question 3**

|  |  |
| --- | --- |
|  | **Accuracy** |
| **Gaussian** | 0.750 |
| **KDE （sigma=5）** | 0.795 |

The performance of KDE is slightly higher than Gaussian Naïve Bayes classifier. Naïve Bayes would actually perform well if the assumption of the Gaussian distribution of attribute values is valid. In this dataset, some values do not follow the Gaussian as stated in Q2. KDE does not care about the shape of the distribution. It only needs to compare the test values and train values. The only impactable parameter is kernel bandwidth. The statistics is the accuracy of KDE with evaluating test and train set for choosing kernel bandwidth from 5 to 25. All of them are higher than Naïve Bayes classifier.

**Question 4**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **SIGMA** | **Accuracy 1** | **Accuracy 2** | **Accuracy 3** | **Accuracy 4** | **Accuracy 5** | **Average** |
| **5** | 0.834 | 0.783 | 0.779 | 0.775 | 0.775 | 0.789 |
| **6** | 0.827 | 0.788 | 0.779 | 0.780 | 0.777 | 0.790 |
| **7** | 0.863 | 0.797 | 0.786 | 0.792 | 0.772 | 0.797 |
| **8** | 0.841 | 0.795 | 0.789 | 0.787 | 0.772 | 0.797 |
| **9** | 0.841 | 0.802 | 0.786 | 0.787 | 0.777 | 0.799 |
| **10** | 0.841 | 0.807 | 0.789 | 0.790 | 0.770 | 0.799 |
| **11** | 0.863 | 0.807 | 0.784 | 0.790 | 0.782 | 0.800 |
| **12** | 0.841 | 0.802 | 0.784 | 0.790 | 0.784 | 0.800 |
| **13** | 0.839 | 0.800 | 0.786 | 0.794 | 0.777 | 0.799 |
| **14** | 0.836 | 0.800 | 0.789 | 0.797 | 0.779 | 0.800 |
| **15** | 0.832 | 0.795 | 0.784 | 0.799 | 0.777 | 0.797 |
| **16** | 0.827 | 0.797 | 0.784 | 0.794 | 0.770 | 0.794 |
| **17** | 0.827 | 0.795 | 0.779 | 0.792 | 0.768 | 0.792 |
| **18** | 0.815 | 0.800 | 0.779 | 0.787 | 0.765 | 0.789 |
| **19** | 0.815 | 0.797 | 0.779 | 0.792 | 0.763 | 0.789 |
| **20** | 0.810 | 0.795 | 0.779 | 0.790 | 0.763 | 0.787 |
| **21** | 0.806 | 0.792 | 0.779 | 0.783 | 0.763 | 0.785 |
| **22** | 0.799 | 0.792 | 0.774 | 0.783 | 0.754 | 0.780 |
| **23** | 0.799 | 0.790 | 0.772 | 0.780 | 0.754 | 0.779 |
| **24** | 0.794 | 0.788 | 0.767 | 0.775 | 0.754 | 0.776 |
| **25** | 0.789 | 0.788 | 0.767 | 0.773 | 0.756 | 0.775 |