Machine Learning Project 1: Pose classification with Naive Bayes

Team member: Aoi Fujii (1086220), Sahasvichni Chou (1068172)

1. Model Evaluation: Macro averaging and Micro averaging

Upon implementing macro-averaging and micro-averaging evaluation approaches to compute the precision, recall, and F-score of the Gaussian-Naïve-Bayes classifier, it resulted with the latter approach having a slightly higher evaluation.

Macro-averaging being the approach that treats each class equally, evaluates how well the model predicts each class. From figure-1.2, it illustrates the model's great potential when predicting certain classes including "mountain", "downward-dog", and "childs" but lacking regarding "seated-forward-bend", and especially, "bridge" where only about 35% was correctly predicted. Hence, accounting for each class's predicting performance, it amounted to 69.76% precision, 70.78% recall, and 68.80% F-score.

Contrastingly, micro-averaging treats each instance equally, meaning regardless of the class, each instance contributes equally to the overall performance. As the model correctly predicted 83 out of 116 instances, it amounted to 71.55% for all evaluation metrics. Interestingly, the overall performance is also portrayed by the frequent classes' performance, particularly, class "mountain", "downward-dog", and "childs", occupying over half of the dataset as shown in figure-1.1. Since these majority classes performed quite well, contributing about 60% to the correct prediction, it would already resulted with reasonable micro-averaging evaluations. Due to this, macro-averaging scored lower as it accounted for the poorly performing classes because each class equally influences the model's performance. Moreover, the slight difference of 2-3% in the evaluation metrics is because, as depicted by figure-1.2, some minor classes also performed well, otherwise, the difference would have been significant.

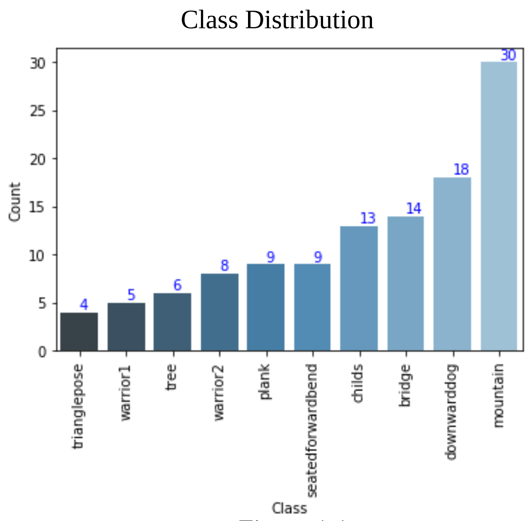


Figure 1.1

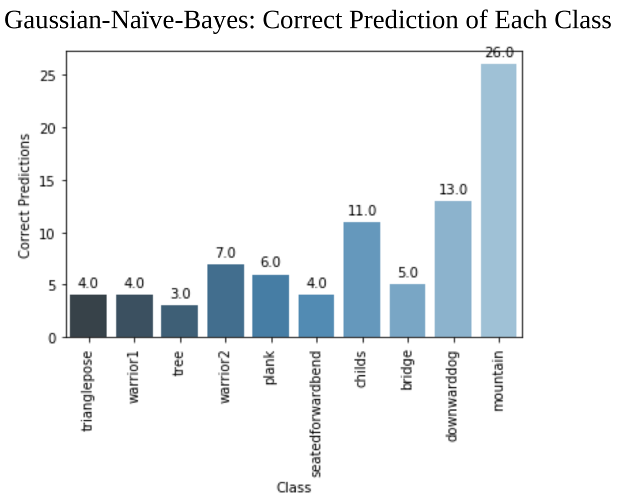


Figure 1.2

Thus, a suitable evaluation approach depends on the implementation's main interest. Macro-averaging provides a better understanding if each class has the same priority while micro-averaging is more appropriate if the interest is the majority classes or the model's general performance, disregarding the classes.

2. Gaussian Assumption Violation

The assumption that the attributes come from a Gaussian distribution was violated for part of the yoga pose dataset.

To identify the distribution of the dataset, 22 attributes for each class were analysed using histograms. It was found that 14 attributes (x1, x2, x7, x8, x10, y1-7, y9 and y11) in the class "tree" have violated Gaussian assumption as shown in table 2.1, figure 2.3-13 and 2.3-14.

The Effect of Gaussian Assumption Violation on Predictions

To analyse the effect of assumption violation on the prediction, the "childs" and "tree" class were selected based on the following 2 reasons.

(1) Both "tree" and "childs" classes have a similar training set size as shown in figure 2.2 (67 and 69 instances). This is to minimize the effect of data size on the prediction by receiving a similar number of training sets.

(2) The "childs" class has more attributes from Gaussian distribution which is comparable to the "tree" class with 14 violation of the Gaussian assumption. As you can observe from figure 2.3-3 and 2.3-4, the "childs" class violates the Gaussian assumption for 5 attributes, x1, x6, x7, x9 and x11.

From figure1.1 and 1.2, the "child" class has an accuracy of 0.846% (11/13 correct predictions) while the "tree" has 0.5% (3/6 correct predictions). Thus, the class "tree" with more violation of the Gaussian assumption had lower prediction accuracy for this model.

However, since this dataset is small, further analysis with more data is required to ensure that the lower prediction performance is caused by the violation of Gaussian assumption.

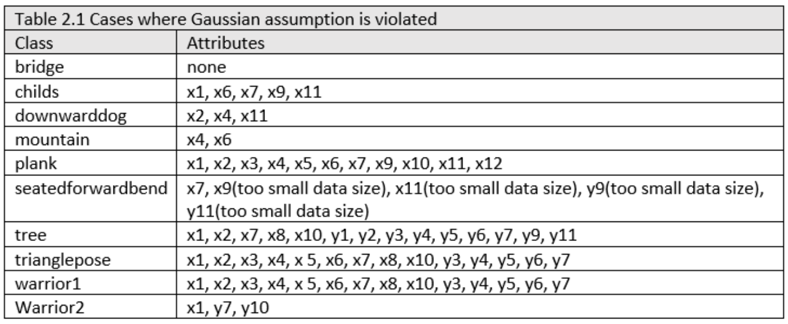
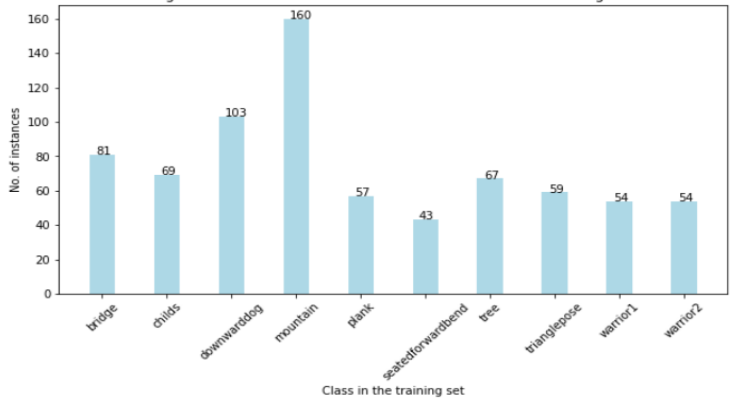


Table 2.1 Cases where Gaussian assumption is violated

Class	Attributes
bridge	none
childs	x1, x6, x7, x9, x11
downwarddog	x2, x4, x11
mountain	x4, x6
plank	x1, x2, x3, x4, x5, x6, x7, x9, x10, x11, x12
seatedforwardbend	x7, x9(too small data size), x11(too small data size), y9(too small data size), y11(too small data size)
tree	x1, x2, x7, x8, x10, y1, y2, y3, y4, y5, y6, y7, y9, y11
trianglepose	x1, x2, x3, x4, x 5, x6, x7, x8, x10, y3, y4, y5, y6, y7
warrior1	x1, x2, x3, x4, x 5, x6, x7, x8, x10, y3, y4, y5, y6, y7
Warrior2	x1, y7, y10

Figure2.2 Total number of each class instances in the training set



Note: the attributes x1..x11 is noted as 1..11 and y1..y11 as 12..20 in the following graph.

Figure 2.3-3 x attributes for childs

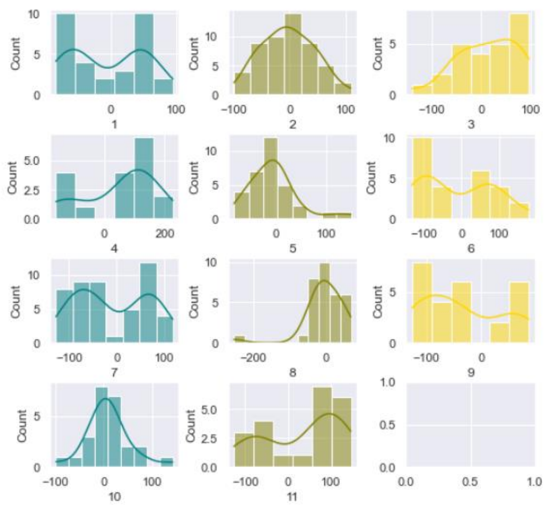


Figure 2.3-4 y attributes for childs

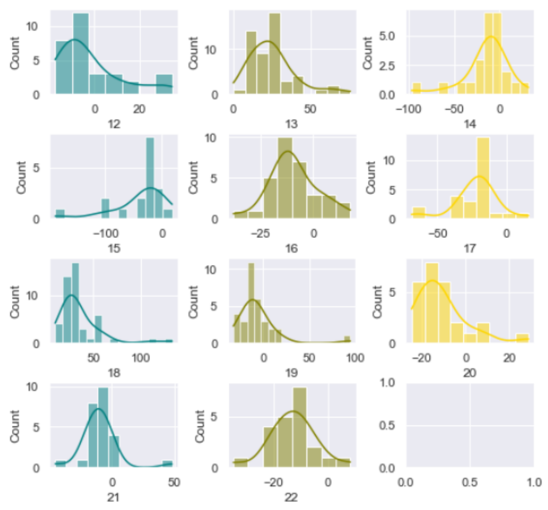


Figure 2.3-13 x attributes for tree

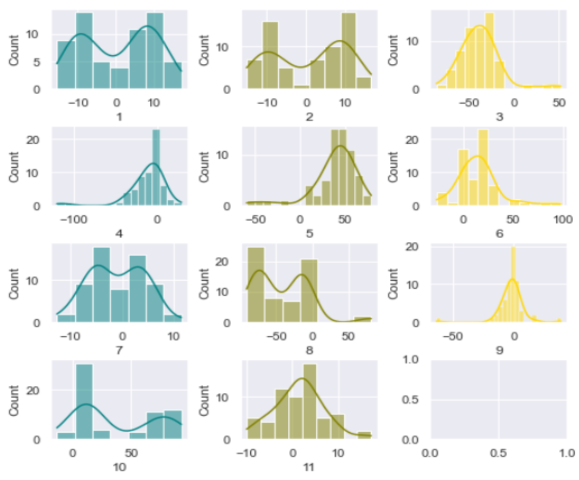
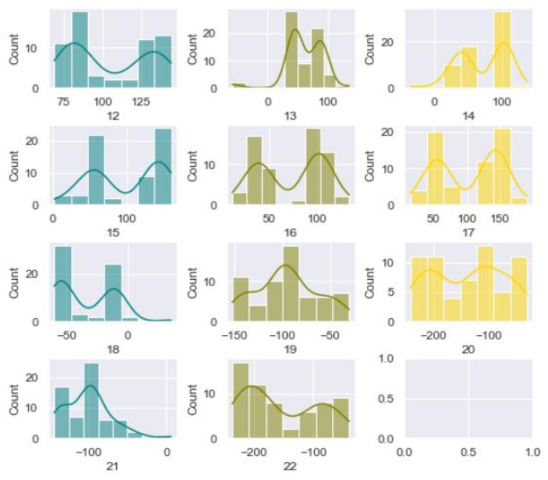


Figure 2.3-14 y attributes for tree



3. Kernel density estimate (KDE) Naïve Bayes Classifier compared to Gaussian Naïve Bayes Classifier

After implementing KDE-Naïve-Bayes as an alternative to modelling continuous data, Gaussian-Naïve-Bayes has a slightly higher micro-average evaluation, by a minor 1%, but KDE-Naïve-Bayes performed better regarding macro-average precision, and recall. Nonetheless, both methods still amounted to roughly the same macro-average f-score at 68%. These results are plausibly due to either Gaussian-Naïve-Bayes being a more suitable model or the arbitrarily chosen bandwidth $\sigma = 15$ is a poor choice.

KDE-Naïve-Bayes: Correct Prediction of Each Class

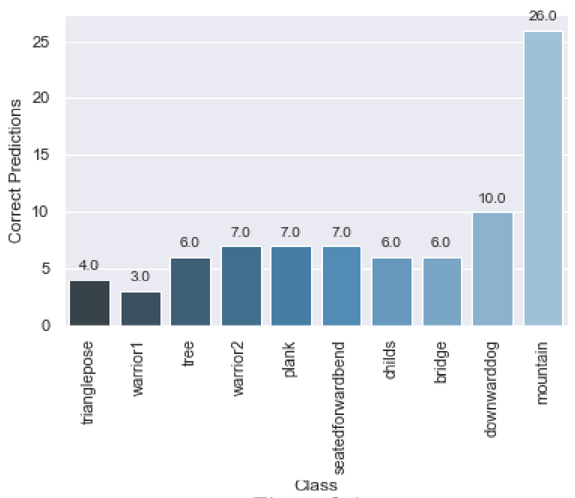


Figure 3.1

Observing the correctly made prediction from both classifiers, shown by figure-1.2 and figure-3.1, some classes performed roughly the same while some experienced a considerable change. Particularly, class “childs” had a 45% decreased, illustrating that for this class, Gaussian-Naïve-Bayes performed significantly better. Further illustrating this, figure-3.2 also shows that more than half of “childs” feature distributions have an approximately normal distribution. Contrastingly, class “seated-forward-bend” performance increased by around 42%, indicating KDE-Naïve-Bayes outperforming Gaussian-Naïve-Bayes. Furthermore, according to its feature distribution in figure-3.3, more than half of the features are non-normal, thus, KDE-Naïve-Bayes would indeed be more effective. Another intriguing class is “bridge” as both implementations often mistook this class for either “childs” or “downward-dog”. Further intuitively studying these three poses, it can be deduced that the y-coordinates are crucial in distinguishing them. Since half of “bridge” y-coordinates’ distributions are non-normal, shown in figure-3.4, it suggests that KDE-Naïve-Bayes is more appropriate. However, from figure-3.1, KDE-Naïve-Bayes only performed slightly better, thus, it is highly likely due to the poor bandwidth choice.

Childs: 22 Features Distribution

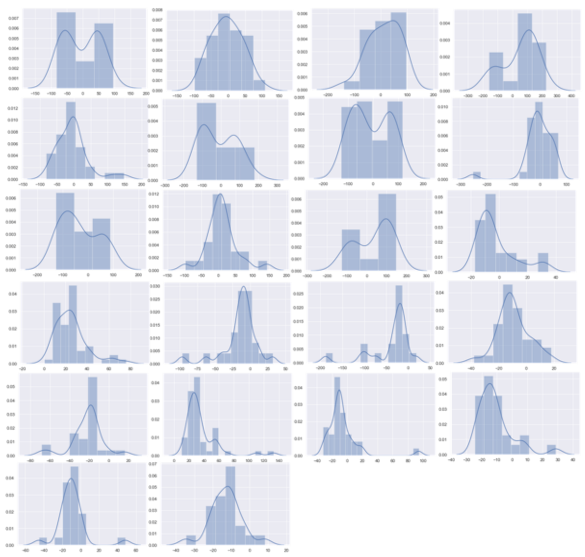


Figure 3.2

Seated Forward Bend: 22 Features Distribution

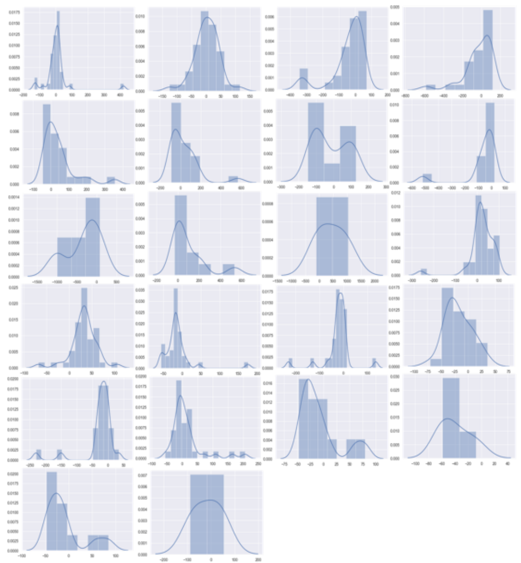


Figure 3.3

Bridge: 11 y-coordinate Features Distribution

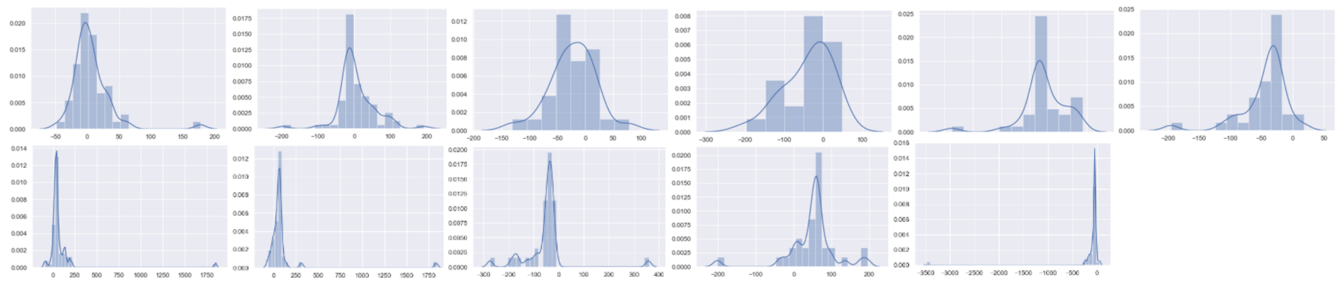


Figure 3.4

Overall, figure-1.2 and 3.1 suggests that compared to Gaussian-Naïve-Bayes, the performance improved for most classes with KDE-Naïve-Bayes, giving a better macro-average evaluation. Furthermore, since KDE-Naïve-Bayes achieved similar micro-average evaluation to Gaussian-Naïve-Bayes, then with an optimal bandwidth, it is likely to outperform Gaussian-Naïve-Bayes.

4. Bandwidth Selection

The kernel bandwidth chosen by random holdout gave a better accuracy for the model than the arbitrary bandwidth.

The bandwidth selection with random hold-out

The dataset from train.csv was separated into a train and validation set with an 8:2 ratio, and the KDE model was trained on the training set. Then, the class label for the validation set was predicted for each bandwidth between 5 and 25. The true labels and predicted labels were used to calculate the accuracy of the model.

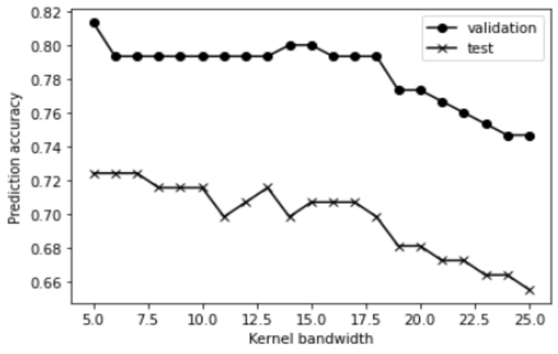
Accuracy for the test set with the selected bandwidth

The kernel bandwidth that performed the best accuracy for the validation set was used to predict the label of the test set. As shown in Figure 4.1, bandwidth = 5 gave the best accuracy of 0.813 for the validation set. Similarly, in the test set, bandwidth = 5 (also 6 and 7) performed the best accuracy of 0.724.

Performance Comparison of arbitrary and selected bandwidth

The prediction accuracy with the arbitrary bandwidth varies significantly depending on the selected bandwidth. In comparison, the random hold-out method gives the better performing bandwidth consistently unless the model overfits the validation set. This is because the test and training set are assumed to have the same distribution and the highest performing bandwidth in the training set tends to give the highest accuracy in the test set. Hence, utilising random holdout for the bandwidth selection will provide a better model performance in most cases than the arbitrary bandwidth.

Figure 4.1 Prediction accuracy of KDE for the bandwidth between 5-25


validation
test
Prediction accuracy
Kernel bandwidth