COMP30027 Machine Learning
Project 1 Report

Tony (Hoang) Dang
Jennifer Xiang

1. Since this is a multiclass classification problem, there are multiple ways to compute precision, recall, and F-score for this classifier. Implement at least two of the methods for computing multiclass precision and recall from the "Model Evaluation" lecture slide 37 (e.g., macro-averaging) and discuss any differences between them. (The implementation should be your own and should not just call a pre-existing function.)

```
---Macro-Averaging----
Precision: 1.0000, Recall: 0.7078, F-Score: 0.8289
---Micro-Averaging----
Precision: 1.0000, Recall: 0.7155, F-Score: 0.8342
```

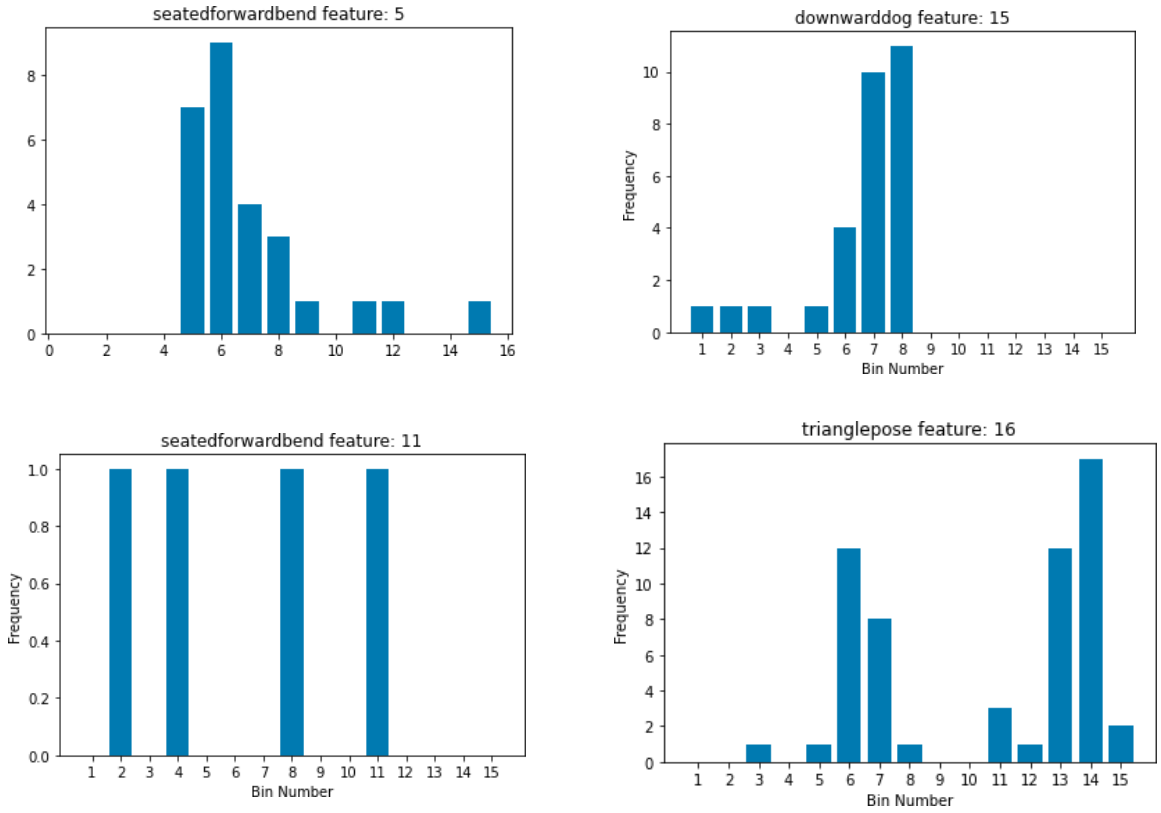
The output of our implementation of macro-averaging and micro-averaging is shown above. Here, micro-averaging yields higher recall and f-score than macro-averaging. Precision is the same for both methods.

As macro-averaging calculates metrics independently for each class and then takes the average, it weighs all classes equally. This works best when there are a relatively equal number of instances for each class. On the other hand, micro-averaging works best when there is a class imbalance as it aggregates the contributions of all classes to calculate the average metric. The fact that micro-averaging produces higher metrics than macro-averaging implies that there is slight imbalance in the number of instances for each class within our dataset.

If the goal of our classifier was to solely maximise the number of correct predictions and minimise the number of incorrect predictions, we would use micro-averaging as our measure of accuracy. However, if we were to value one particular minority class the most, we would likely use macro-averaging as it is insensitive to the imbalance of classes and treats them all as equal. For example, this is crucial in rare disease diagnosis where importance is placed in minority classes such as instances where a patient is carrying such a disease. In our case, as we do not value one class over another and are interested in simply maximising the number of correct predictions, we would likely choose micro-averaging as our measure of performance.

2. The Gaussian naive Bayes classifier assumes that numeric attributes come from a Gaussian distribution. Is this assumption always true for the numeric attributes in this dataset? Identify some cases where the Gaussian assumption is violated and describe any evidence (or lack thereof) that this has some effect on the NB classifier's predictions.

To see if the numeric attributes in this dataset align with the Gaussian assumption, we discretised values into equal-width bins and plotted the attributes on a bar graph. From our plots, we could see that some attributes clearly do not follow a normal distribution. Some of these plots are illustrated below.



We can observe that the distributions of the first two plots are non-symmetric, which violates the Gaussian assumption of symmetricity. Furthermore, the plot "seatedforwardbend feature: 11" is more indicative of a uniform distribution and lacks the characteristic bell-curve shape of a Gaussian distribution. The data on the plot "trianglepose feature: 16" contains two peaks, but a Gaussian distribution should only contain one.

It is apparent that there are cases in the data that violate the Gaussian assumption. As our Naive Bayes classifier assumes that our numeric attributes come from a Gaussian distribution, if the actual distribution is wildly different, the predictions of certain labels may not be completely accurate. However, when a feature with invalid assumption (say "trianglepose feature: 16" as implemented on the jupyter notebook), is removed from the dataset (replacing all values with NaN), this in fact decreases the accuracy of our Naive Bayes classifier (from 71.55% to 68.97% accurate). This most likely indicates that the loss of information from removing such a feature has a greater impact than that of an incorrect assumption.

Hence, in our case, we cannot conclude definitively that the violation of the Gaussian assumption has a detrimental effect on the accuracy of our classifier.

3. Implement a kernel density estimate (KDE) naive Bayes classifier and compare its performance to the Gaussian naive Bayes classifier. Recall that KDE has kernel bandwidth as a free parameter – you can choose an arbitrary value for this, but a value in the range 5-25 is recommended. Discuss any differences you observe between the Gaussian and KDE naive Bayes classifiers. (As with the Gaussian naive Bayes, this KDE naive Bayes implementation should be your own and should not just call a pre-existing function.)

```
---Naive Bayes Classifier---
Accuracy: 71.55%

---KDE---
Kernel Bandwidth: 5
Accuracy: 72.41%

Kernel Bandwidth: 15
Accuracy: 70.69%
```

From the figures provided above, the KDE NB classifier performed similar to Gaussian NB classifier in terms of accuracy. A bandwidth of 5 and 15 was randomly chosen for this task.

Unlike the Gaussian NB model, which assumes that all the data is normally distributed, the KDE NB model does not make any assumption about the distribution of the dataset. The KDE model learns the probability distribution of each feature for each class straight from the data as it iterates through the training dataset. Another key difference is that the KDE model takes longer to compute probabilities for each data point due to the fact that it has to compare each test data point to all the training data points for each class, whereas the simple Gaussian model simply calculates a likelihood using a mean and standard deviation derived from the training dataset. Since the KDE model is very reliant on the training data's distributions, there are also potential problems with overfitting and which may reduce the accuracy and generalisability of the model.

Despite modelling arbitrary distributions of the data instead of making assumptions, our KDE model produced a similar accuracy to our Gaussian model. This may be because the data provided was mostly normally distributed; and thus the Gaussian assumption was suitable for the task. That being said, the choice of kernel bandwidth proved important in influencing the accuracy of the classifier. As shown in the figure above, a kernel bandwidth of 5 yielded higher accuracy than the Gaussian model, whereas a bandwidth of 15 produced a lower accuracy.

5. Naive Bayes ignores missing values, but in pose recognition tasks the missing values can be informative. Missing values indicate that some part of the body was obscured and sometimes this is relevant to the pose (e.g., holding one hand behind the back). Are missing values useful for this task? Implement a method that incorporates information about missing values and demonstrate whether it changes the classification results.

Following the logic proposed in the question, we reasoned that if there are a higher number of missing values in a certain feature-class set, it is more likely that an instance with a corresponding missing value will also belong to that class. For example, if a test instance is missing feature x, and, for a given class, feature x is missing 90% of the time in the training set, it is likely that this test instance belongs to that class, and that the missing values are missing on purpose.

Hence, we implemented a simple method that takes this into account. If a test feature is missing, we would compute the proportion of missing values for that feature for each class in the dataset, and use this as a probability that the instance belongs to that class. The accuracy of our new implementation is shown below.

```
---Naive Bayes Classifier (Ignoring NaNs)---
Accuracy: 71.55%

---Naive Bayes w/ NaNs---
Accuracy: 68.1%
```

From these figures, the NB classifier that made use of missing values (NaNs in our model) performed similar to the NB classifier that ignored such values. This could be due to the fact that most of the poses in our dataset do not contain positions that naturally obscure certain body parts, thus making it naive to assume that **all** missing data is explainable as such. However, in the case where data is not missing at random and consistently contains obscured data points, our new method has the potential to outperform the original method of simply ignoring missing data.