Assignment 1: Naive Bayes & KDE Classifier Report

Student ID: 1086213, 1090015

Q1.

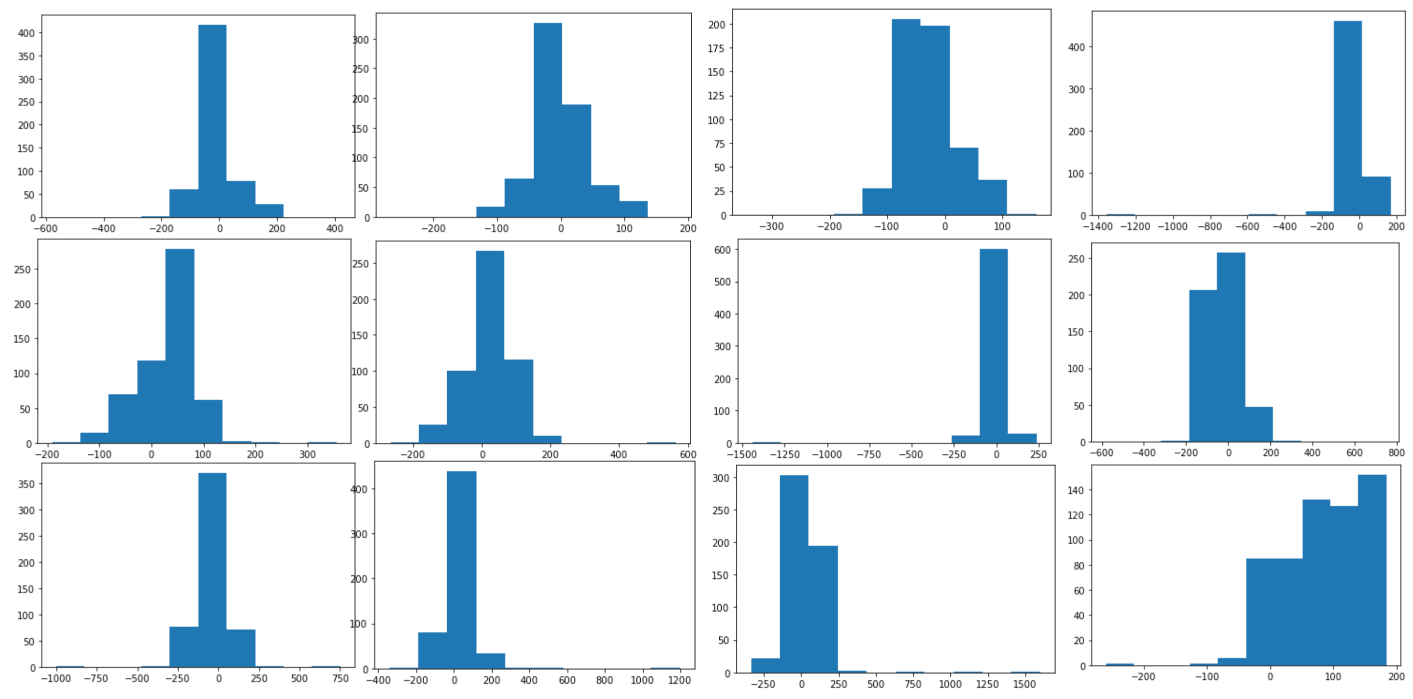
A macro-averaging computes the metrics for each class independently, then takes the average based on all the classes equally. Meanwhile, micro-average aggregates the contributions of all classes to compute the average metric. Since both of them use a different computation, their interpretation differs as well.

After calculating using both the methods, we found out that macro-averaging gives precision of 0.7189602683178534, while micro-averaging gives precision of 0.75. This is because in a multi-class classification setup, we may have a lot more examples from one class than from other classes. Hence, our micro average tends to be higher (i.e. class that has more sample data, in this case, 'mountain' with TP = 26). On the other hand, macro-average doesn’t take imbalance into account so it is encouraged to try to recognize every class correctly. Thus, in this data, micro-averaging gives higher accuracy because there is class imbalance.

If we would like to know how the system performs overall across the sets of data, macro average is recommended. On the other hand, it may be helpful to measure our dataset using micro-average when the size of classes varies.

Q2.

The test showed the normal distribution assumption is unlikely to be true because the histogram plots show that not all of the dataset looks bell-shaped and is normally distributed. Below are the histogram plots of each feature that we generated:



Notice that some of the histograms are not bell-shaped, indicating that the distribution is not normal such as skewed to the left, skewed to the right, or even bimodal. As a result, the gaussian assumption is violated.

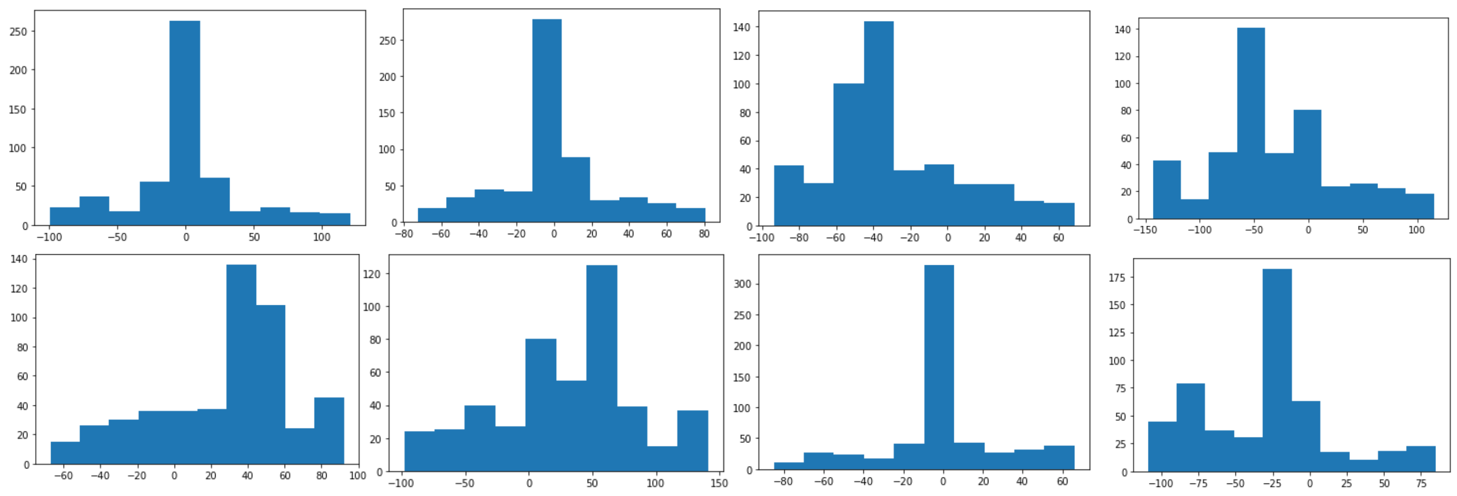
One of the reasons why the gaussian assumption may be violated is because there are some outliers that might affect our prediction. Here, we have implemented a method to remove outliers:

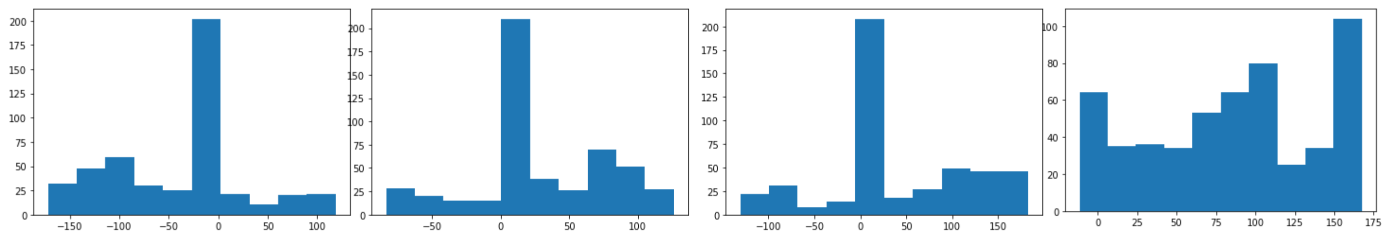
```
lower_bound=train_file.quantile(0.05)
upper_bound=train_file.quantile(0.95)
class_label=train_file[0]
new_feature=train_file.iloc[:,1:][(train_file<upper_bound)&(train_file>lower_bound)]
new_train=pd.concat([class_label, new_feature], axis=1)
mean_list, sd_list, d_prior, keys_list = train(new_train)

prediction = predict(test_file.iloc[:,1:], mean_list, sd_list, d_prior, keys_list)
accuracy=evaluate(prediction, test_file.iloc[:,0])
print(accuracy)
```

```
plotting the first 5 attribute of our data
(('accuracy', 0.7946428571428571), {'macro precision': 0.7829839020628494, 'macro recall': 0.7782142857142857, 'macro F-scor
e': 0.7748490204859374, 'micro precision': 0.7946428571428571, 'micro recall': 0.7946428571428571, 'micro F-score': 0.794642
8571428572})
```

We identify outliers by defining limits on the sample values that are below the 5th percentile or above the 95th percentile. This value is obtained after numerous trials.

It shows that the accuracy increased from 75% to 79% and this shows evidence that it affects our prediction. Below are the histogram plots after we eliminate outliers:





We use histograms to show the first six features after the outlier has been removed.
Overall, although it is not perfect bell-shape, however it shows the familiarity of Gaussian distribution and the approximation of bell shape in each of the attributes which might affect the performance.

Q3.
Using KDE Naive Bayes classifier with bandwidth = 5, we found the accuracy a bit higher, 0.7589 as compared to the Gaussian Naive Bayes classifier. Here, we chose bandwidth = 5 because it gives the highest accuracy compared to other bandwidths between 6-25.
In general, the KDE naive bayes classifier uses the sum of Gaussian distribution with a free parameter σ.
The difference that we observe between Gaussian and Naive Bayes is that the KDE uses a smoothing parameter (the standard deviation) called the bandwidth to control the scope or check the robustness of the estimate. The drawbacks of the kernel density estimation is that it may be biased if larger bandwidth is chosen, particularly near the boundaries (when the data is bounded), because the graph will be more ‘smoothed’. On the other hand, Gaussian did not use bandwidth, instead it uses the standard deviation based on the raw data itself and thus it leads to a lower accuracy. Also, as compared to Gaussian which uses discrete bins, a kde plot smooths the observations with a Gaussian kernel, producing a continuous density estimate.

Q5.
Missing values might be important in some cases. For example, with our algorithm, we ignore the missing values while computing Naive Bayes classifier. Here, we may be missing some important information such as when these missing values are actually some body parts that are undetected in the image because they were hidden, such as when holding one hand behind the back. Not only that, reduction in the data means less data to train the model and as a result it drops the accuracy of our test set.

For this reason, instead of ignoring the missing values, we may want to calculate the mean value of each class of each feature. These mean values will be a reasonable representation of the data since our data is mostly Gaussian distributed. Moreover, imputing them with the mean of each class might also increase the accuracy of our prediction although in general there is no significant difference in the model accuracies when using such imputation, this may be because there are better ways to impute the data other than using the mean. In particular, after imputing the missing values with mean, we got accuracy of 76.7% which is 1% higher than our previous accuracy, 75.8%.