Kobee Raveendran
CAP5610 Assignment 1

K-Nearest Neighbors

Pseudocode:

```
for sample in test set:
    compute and store distance between every training sample to test sample

    sort list of distances
    select subset of the first k elements

    compute counts of each class among the subset
    return class with the highest count
```

Gaussian Naïve Bayes

Pseudocode:

```
split dataset by class

for each class_split:
    for each feature:
        compute mean
        compute biased standard deviation

for each class split:
    class_probabilities = gaussian_pdf(feature vector, mean, standard_dev)

return class_index(max(class_probability))
```

Results

K-nearest neighbors

(average accuracies [%] across 5 folds):

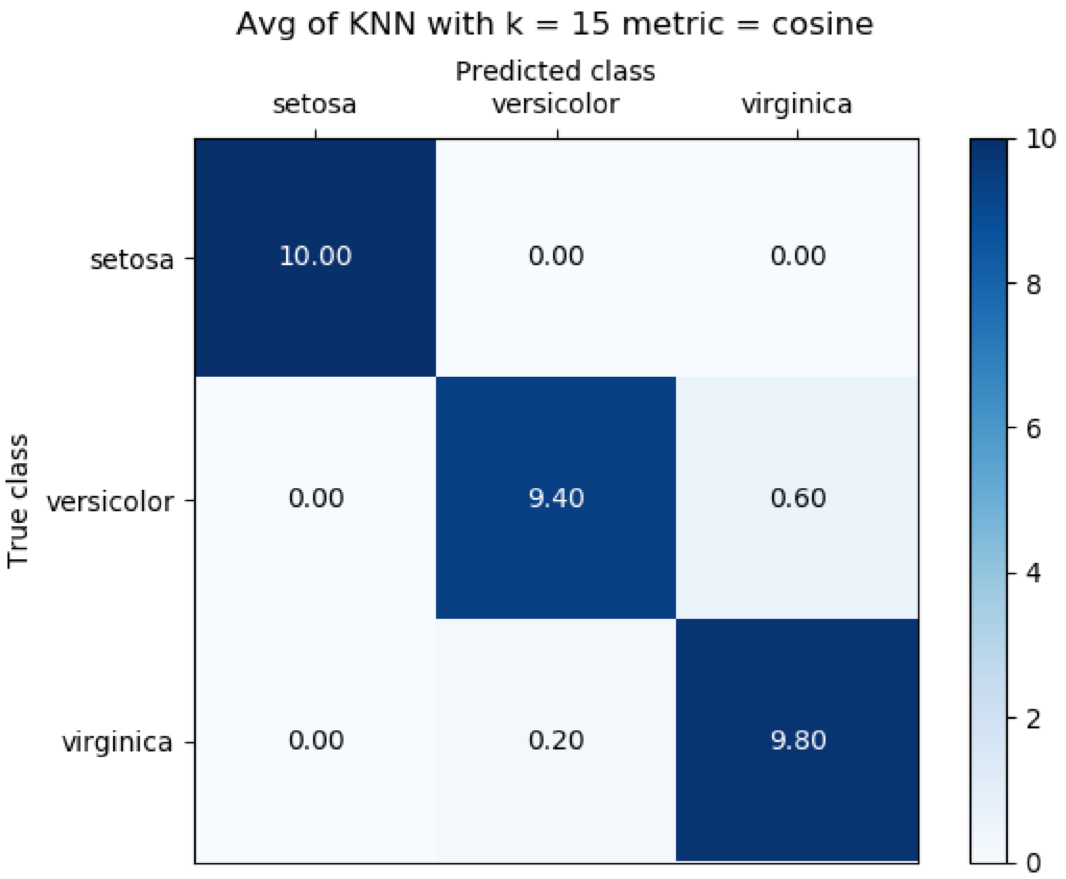
	K = 1	K = 5	K = 15
Dist = Euclidean	95.33	96.0	96.67
Dist = cosine	94.67	96.67	97.3

Best-performing metric (varied across multiple runs of KNN but overall produced the highest accuracy more often than the alternatives):

K = 15, distance metric = cosine

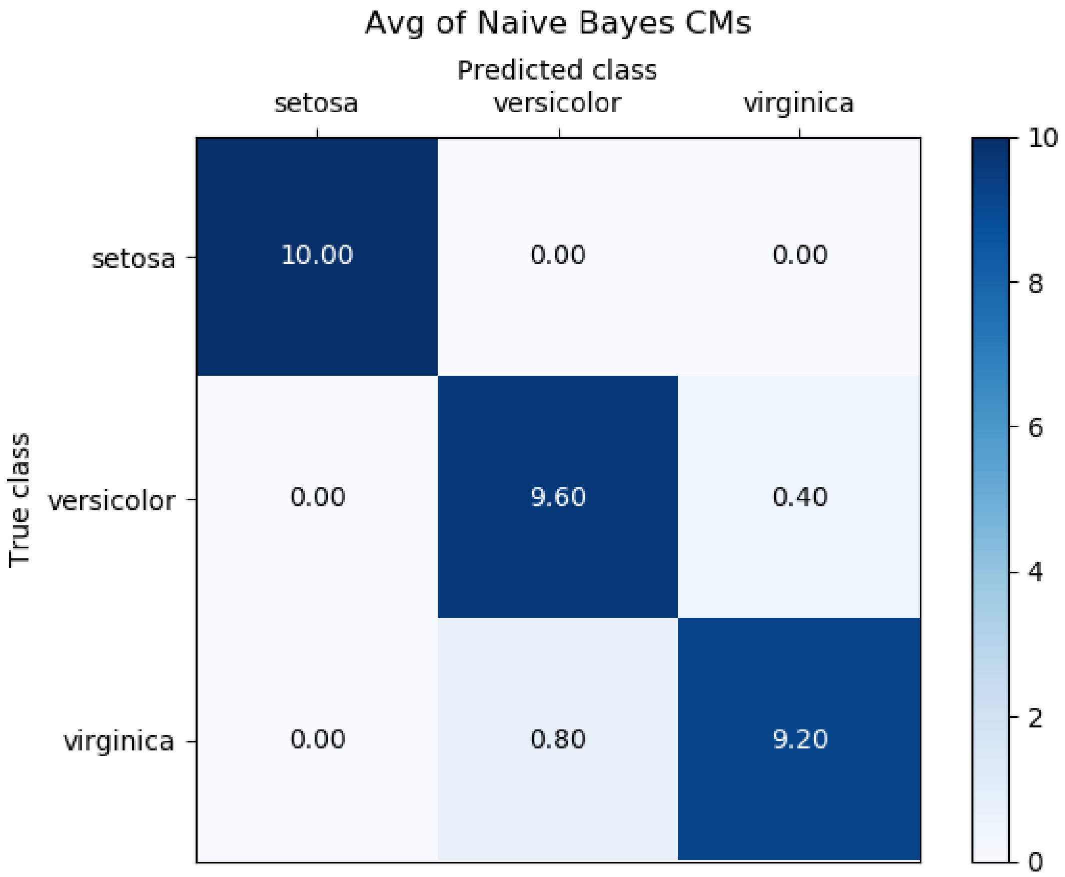
Average accuracy: 97.3

Average of each fold's confusion matrices:



Naïve Bayes

Average of each fold's confusion matrices:



Note: these can be verified using the confusion matrices I've generated and stored automatically for every condition and fold in the plots *folder.*

Question: Explain the difference between MLE and MAP procedures for Naïve Bayes. If you had to implement this using MAP, how would your code have been different? Identify one case in which MLE and MAP would produce the same/similar answers.

MLE is simply a specific case of MAP in which the prior probability is uniform or constant. That is, we expect the probability of any of the classes to be equally likely (or we have no information on the prior and have to assume so). In the case of Naïve Bayes, this would mean that, instead of using only the estimated parameters (mean and variance) in calculating the Gaussian PDF, we'd also need to multiply by the prior class probabilities (which we'd draw from the dataset by determining the frequencies of each class appearing) then choose the most likely of those. This would steer the predictions toward guesses that we'd deem make more sense based on the probability of belonging to the specified class in the observed data. If a class almost never occurs, MLE may still label a set of features as belonging to that class since it assumes equal distribution, but MAP can give a more informed prediction and instead gear the prediction more toward similar candidates whose classes appear more often in the data. As for cases in which they are similar, I'd imagine this happening for problems in which the problem you're dealing with involves data that follows a uniform distribution. For example, the distribution of numbers in the MNIST dataset (each digit is probably equally likely to appear in normal scenarios), or dice rolling (where, on a fair dice, each outcome [class] is equally likely).