Software Project Final Report

Surya Dutta

Final Report

The final report involves testing the kthperclass classifier that you implemented and then comparing performance with the kNN classifier.

First, the kthperclass classifier is tested for **k=1** and **k=2**, using all 150 datapoints as training data. The test vectors to be used are given in Table D.

Table D. Iris dataset test vectors and results for kthperclass classification.

Test case	Petal length	Petal width	True Class	Pred Class k=1	Pred Class k=2
1	2.0	0.8	Setosa		
2	4.0	0.8	Versicolor		
3	6.5	2.5	Virginica		
4	4.5	1.7	Virginica		
5	4.8	1.8	Virginica		
6	5.0	1.8	Versicolor		
7	5.0	1.5	Virginica		

Provide the following for the test results.

1. Complete Table D with the kthperclass classifier results for k=1 and k=2.
2. For both k=1 and k=2, provide confusion matrices.
3. For both k=1 and k=2, fill in Table E, providing the overall probability of classification error (# errors/ number of test) as well as conditional classification error probabilities, conditioned on the true class.

Table E. Overall & conditional classification error probabilities for kthperclass classification trained with the irisf34 dataset.

kthperclass k value	Overall Error	Pe | setosa	Pe | versicolor	Pe | virginica
1				
2				

Second, compare the classification error performance for the kNN and kthperclass classifiers. You will need to write two functions.

1. **Splitdata**: it will take the original data set and split it into a training set and a test set. For the Iris dataset, this will involve taking the 50 vectors of each class and using the first Nt for training and the remaining 50-Nt for test. Warning: because of the order of the vectors and labels in the dataset, one cannot simply keep the first 90 vectors as this would give 50 setosa vectors and 40 versicolor vectors.

2. **Kthperclass**: it will take a training set (features and labels) and a test set (feature vectors) and produce a set of detected (predicted) labels.

You will also need to write an overall script or set of scripts, using these and other functions, to evaluate the probability of classification error as a function of k. Use this script with Nt = 30 (30 training vectors per class, 20 test vectors per class) to compare

1. kNN classification, k=1 through 17, and
2. kthperclass classification, k=1 through 17

Results for Nt = 35 are shown in Fig. C**.**

Question: Explain why the probability of classification error appears to take on only certain values.

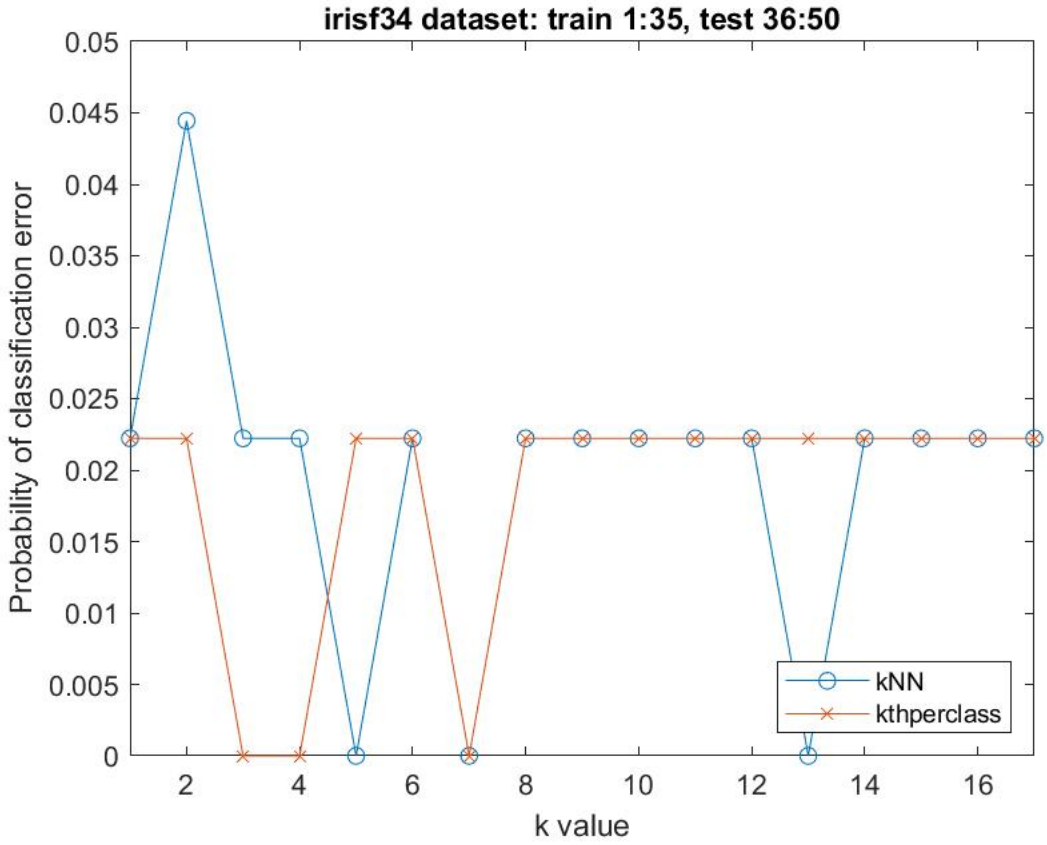

Figure D. Performance results for 35 training vectors per class, 15 test vectors per class.

Test Results for kthperclass classifier

1. Results for the kthperclass classifier for k=1 and k=2 are given in Table 1.
2. Confusion matrices for k=1 and k=2 are given in Figs. 1 and 2, respectively.

3. Overall and conditional probability of classification error values are given in Table 2.

Table 1. Iris dataset test vectors and results for kthperclass classification for final report.

Test case	Petal length	Petal width	True Class	Pred Class k=1	Pred Class k=2
1	2.0	0.8	Setosa	Setosa	Setosa
2	4.0	0.8	Versicolor	Versicolor	Versicolor
3	6.5	2.5	Virginica	Virginica	Virginica
4	4.5	1.7	Virginica	Virginica	Versicolor
5	4.8	1.8	Virginica	Versicolor	Virginica
6	5.0	1.8	Versicolor	Virginica	Virginica
7	5.0	1.5	Virginica	Virginica	Versicolor

Table 2. Overall & conditional classification error probabilities for kthperclass classification trained with the irisf34 dataset.

kthperclass k value	Overall Error	Pe | setosa	Pe | versicolor	Pe | virginica
1	0.14285	0.0	0.0	0.25
2	0.42857	0.0	0.5	0.5

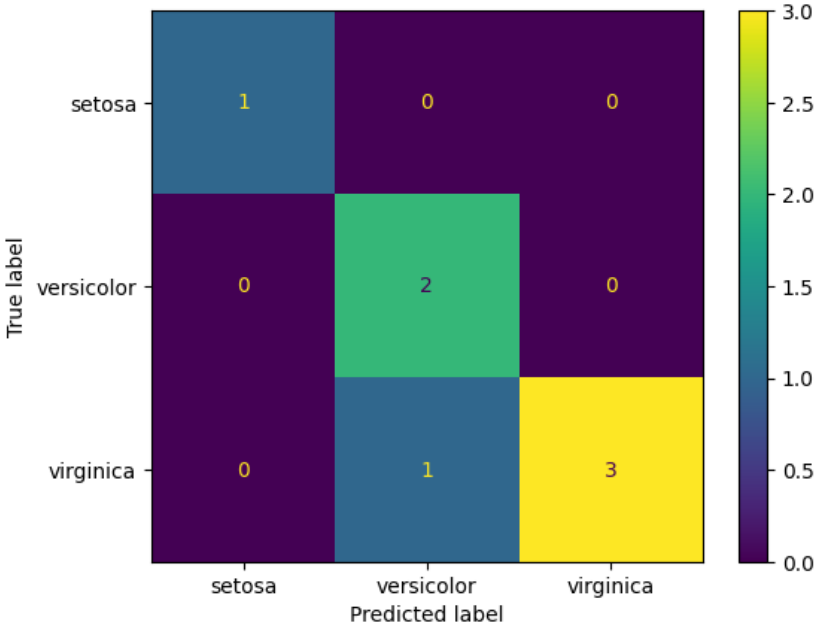

Figure 1. Confusion matrix for kthperclass classifier and k=1.

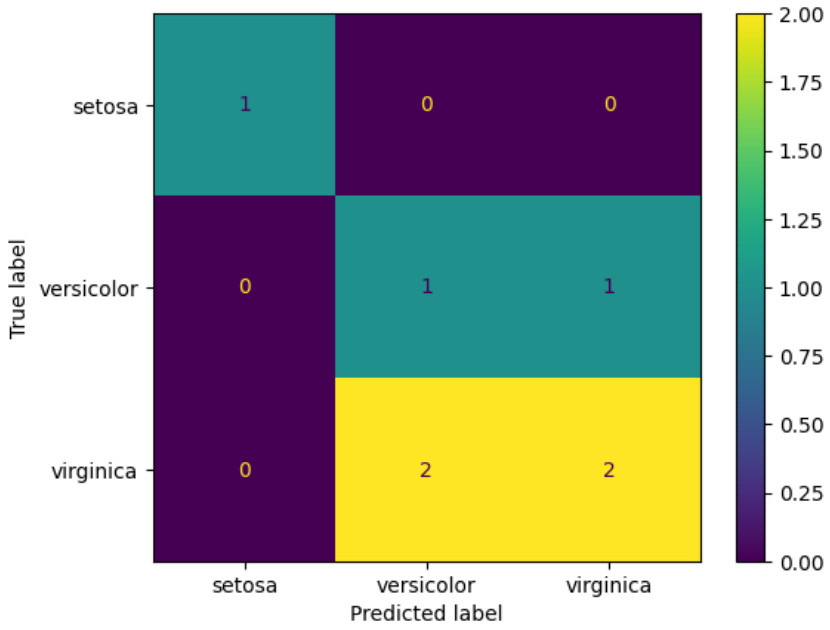

Figure 2. Confusion matrix for kthperclass classifier and k=2.

Results comparing kNN and kthperclass classifiers

Classification error as a function of k is shown for the two classifiers in Fig. 3.

Question: Explain why the probability of classification error appears to take on only certain values.

Answer: The Iris dataset's unique distribution results in the classification error assuming specific values. Notably, when features 3 and 4 are employed, the setosa class can be perfectly separated, but there are a few datapoints from the versicolor and virginica classes which cannot be separated by using these features. Consequently, the classifier's errors are confined only to these set of overlapping points, giving only certain probability values of the classification error.

Figure 3. Pe vs. k for the kNN and kthperclass classifiers using the iris dataset

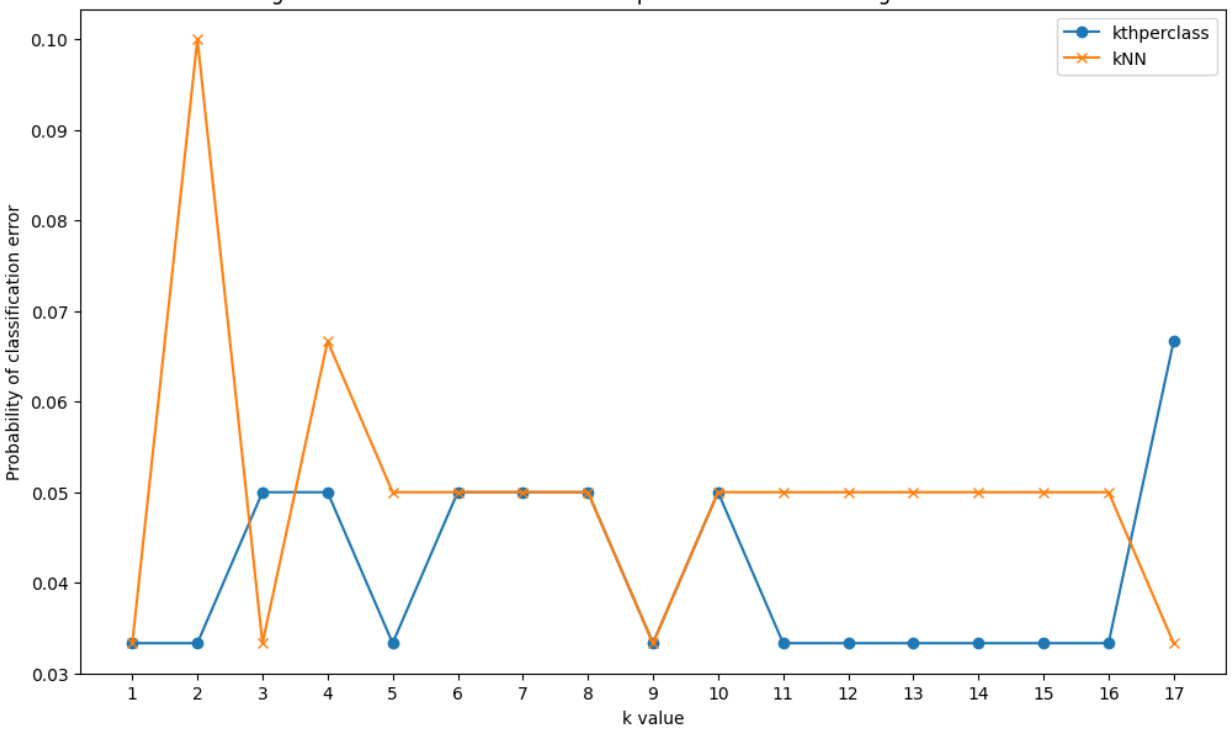

Figure 3. Pe vs. k for the kNN and kthperclass classifiers using the iris dataset (first 30 vectors per class for training and remaining 20 vectors per class for testing).

Appendix A: code listing of function to split the data

```
[40] def split_data(X, y, Nt):

        unique_classes = np.unique(y)
        X_train, X_test, y_train, y_test = [], [], [], []

        for cls in unique_classes:
            # Filter the data for the current class
            X_cls = X[y == cls]
            y_cls = y[y == cls]

            # Split the data for the current class
            X_train_cls, X_test_cls = X_cls[:Nt], X_cls[Nt:]
            y_train_cls, y_test_cls = y_cls[:Nt], y_cls[Nt:]

            # Append the split data to the overall training and testing sets
            X_train.append(X_train_cls)
            X_test.append(X_test_cls)
            y_train.append(y_train_cls)
            y_test.append(y_test_cls)
```

```
    # Concatenate the data from each class to form the final training and testing sets
    X_train = np.concatenate(X_train)
    X_test = np.concatenate(X_test)
    y_train = np.concatenate(y_train)
    y_test = np.concatenate(y_test)

    return X_train, X_test, y_train, y_test

# Using the split_data function to split the Iris dataset with Nt=30
Nt = 30  # Number of training vectors per class
X_train, X_test, y_train, y_test = split_data(X, y, Nt)

# Test split
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

✓ Connected to Python 3 Google Compute Engine backend

Appendix B: code listing of function to implement the kthperclass classifier

```
# Function to implement the kthperclass classifier
def kthperclass_classifier(X_train, y_train, X_test, k):
    predictions = []
    classes = np.unique(y_train)

    for test_point in X_test:
        fom_per_class = []

        for cls in classes:
            class_points = X_train[y_train == cls]
            distances = [cityblock(test_point, train_point) for train_point in class_points]
            distances.sort()
            kth_distance = distances[k-1] if k <= len(distances) else np.inf
            fom_per_class.append(kth_distance)

        min_fom = min(fom_per_class)
        detected_classes = [i for i, fom in enumerate(fom_per_class) if fom == min_fom]
        detected_class = min(detected_classes)
        predictions.append(detected_class)

    return np.array(predictions)
```