Dylan Parker
9/12/23
HW1 Answers
CS334-2

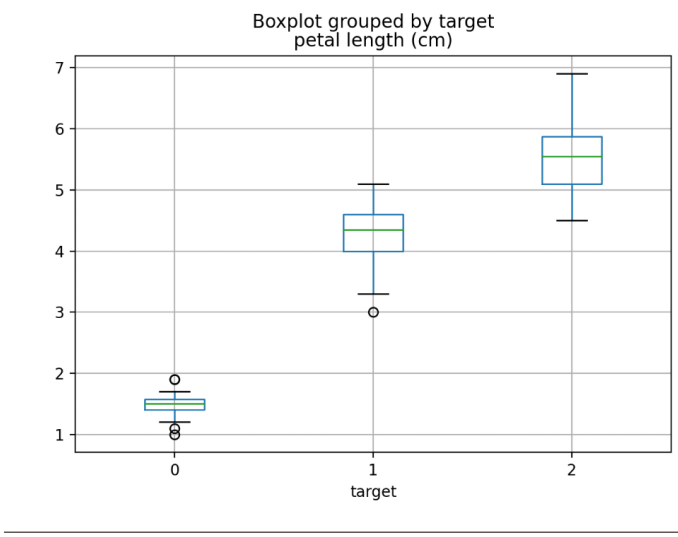
1d) The vectorized approach took 0.08% of the time it took for the nonvectorized approach to calculate the sum of squares.

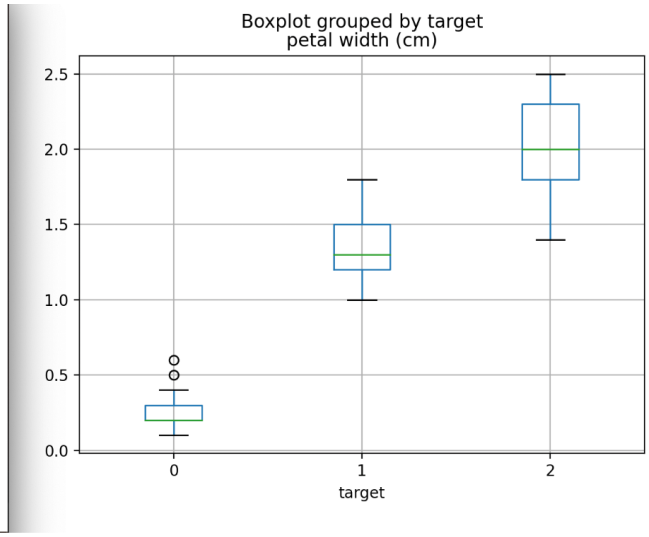
```
Time [sec] (for loop): 1.2458089579595253
Time [sec] (np loop): 0.0010030419798567891
```

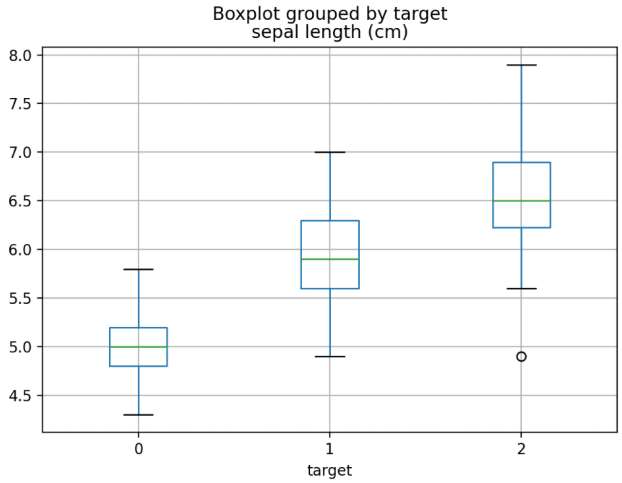
2d) I would only seriously consider petal length and width when classifying the species type, because the boxplots show little overlap between the target species when petal length and width are compared, however, the boxplots show lots of overlap when sepal length and width are compared. Additionally, the scatterplot between petal length and width indicates a strong positive linear correlation, however, the scatterplot between sepal length and width is a more random distribution between class 1 and 2. You can classify 0 based on the scatter plot though, but I would still prefer only using petal width and length to predict the class. Because of these signs, I would not be able to trust sepal length or width to predict the species type.

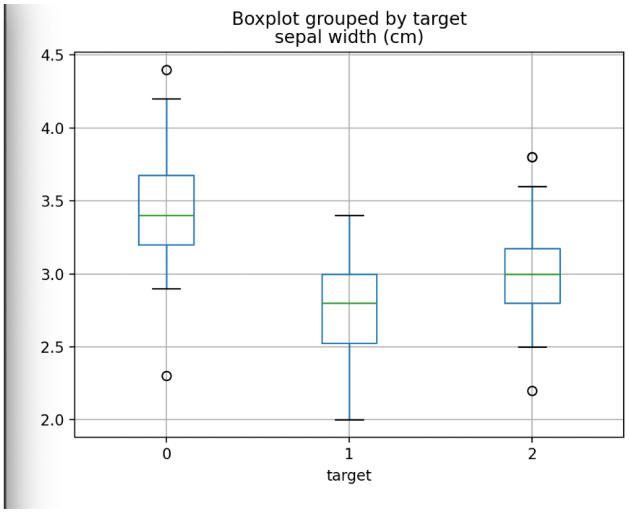
Here are some rules:

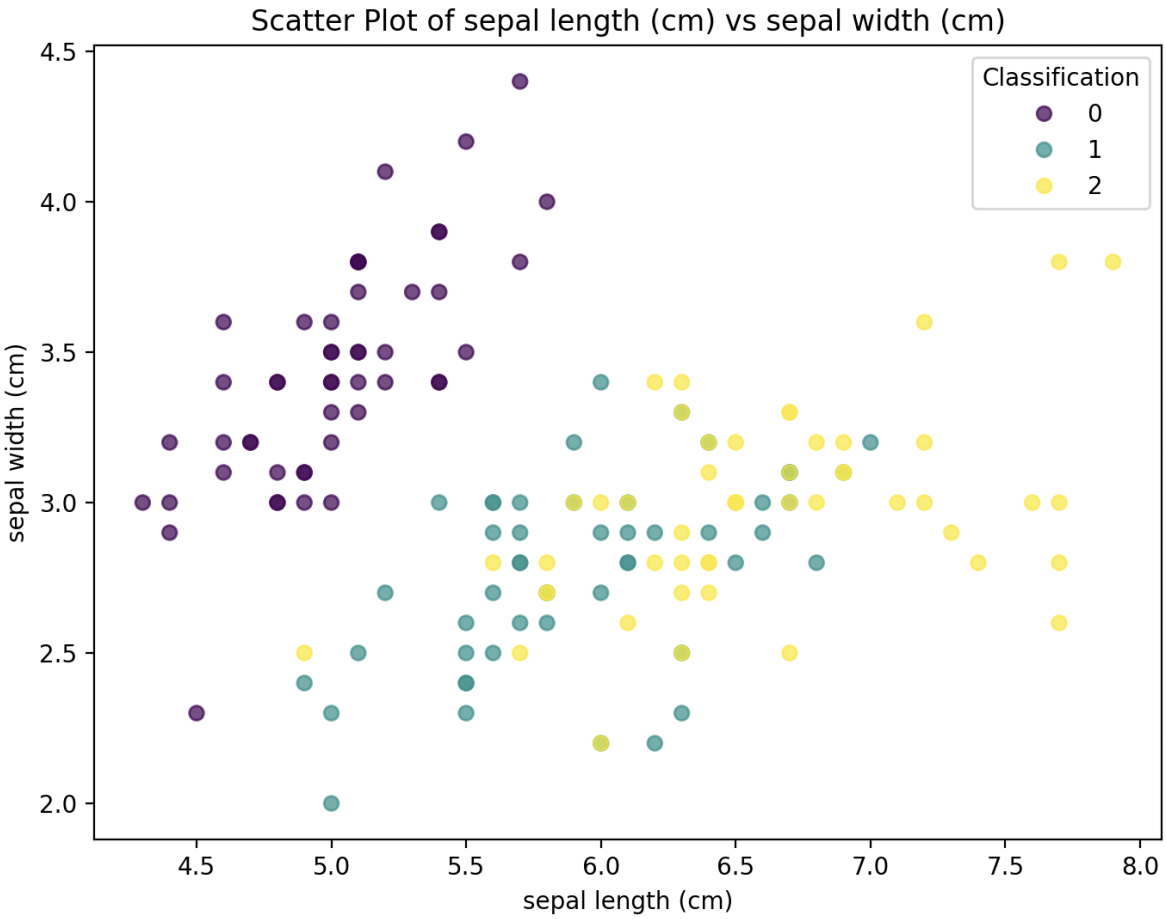
1. If the petal length is between 0cm and 2cm and its width is between 0cm and 0.75cm, then it is classified as 0.
2. If the petal length is between 3cm and 4.6cm and its width is between 1cm and 1.6cm, then it is classified as 1.
3. If the petal length is between 4.8cm and 7cm and its width is between 1.75cm and 2.5cm, then it is classified as 2.

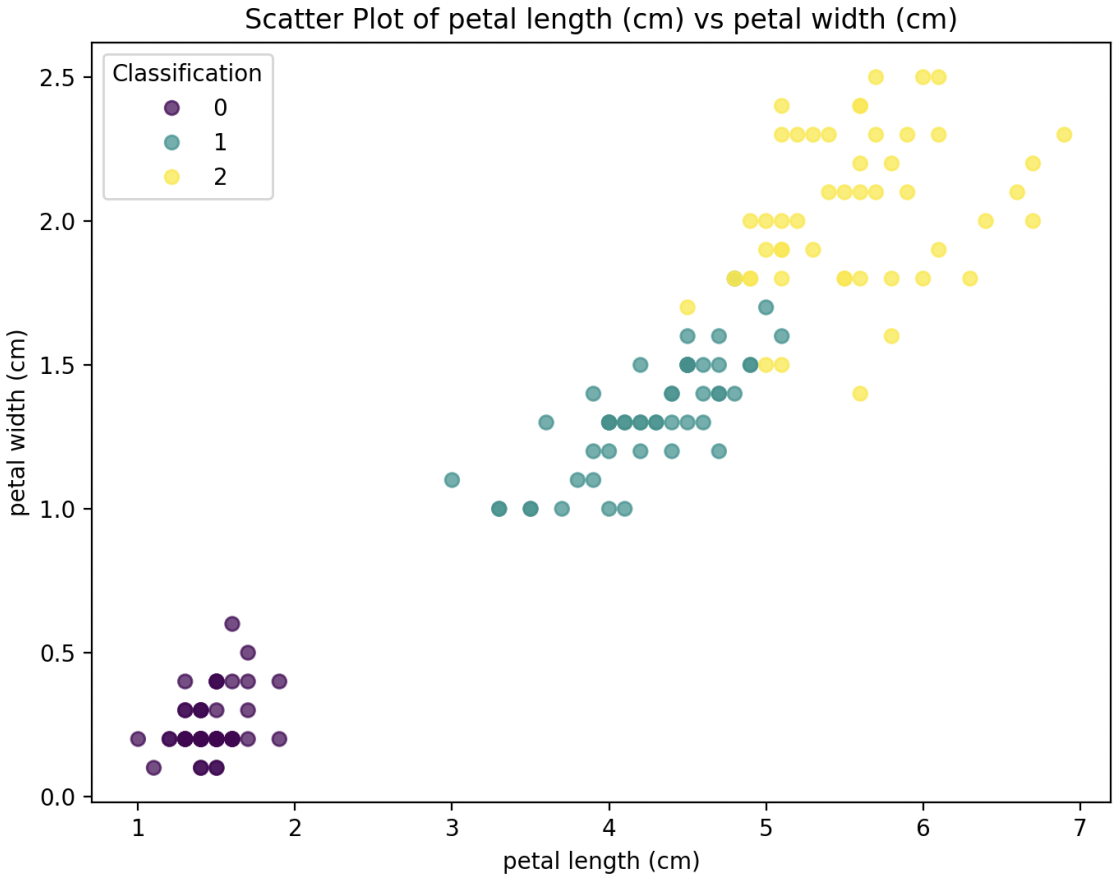

Boxplot grouped by target
petal length (cm)
7
6
5
4
3
2
1
0
1
2
target


Boxplot grouped by target
petal width (cm)
2.5
2.0
1.5
1.0
0.5
0.0
0
1
2
target


Boxplot grouped by target
sepal length (cm)
8.0
7.5
7.0
6.5
6.0
5.5
5.0
4.5
0
1
2
target


Boxplot grouped by target
sepal width (cm)
4.5
4.0
3.5
3.0
2.5
2.0
0
1
2
target


Scatter Plot of sepal length (cm) vs sepal width (cm)
Classification
0
1
2
sepal width (cm)
sepal length (cm)
4.5
4.0
3.5
3.0
2.5
2.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0

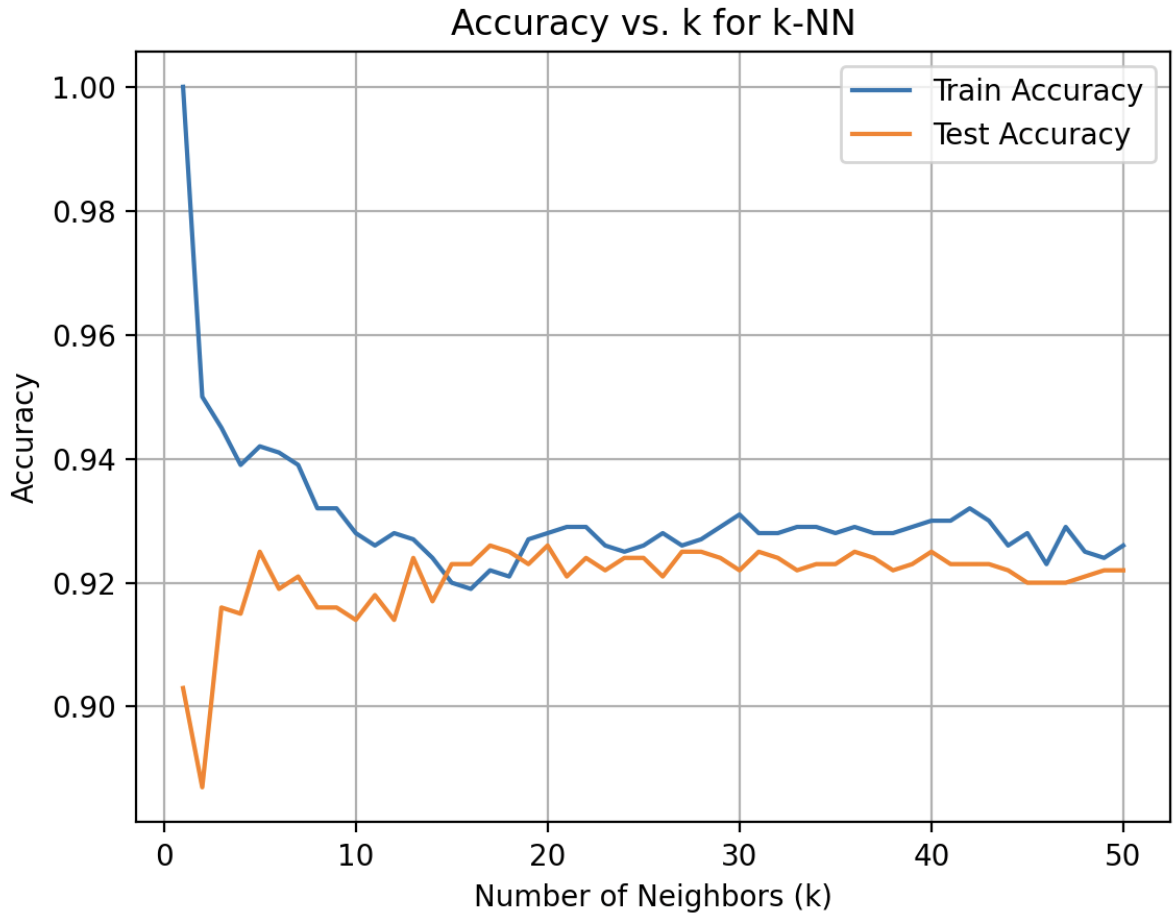


3d) I got a training accuracy of 92.6% and a testing accuracy of 92.2% at k=50. It should be noted that this accuracy would be basically the same at k=20 though.

```
(base) dylanethan@MacBook-Pro-284 hw1_template % python knn.py 50
Training Acc: 0.926
Test Acc: 0.922
```

```
(base) dylanethan@MacBook-Pro-284 hw1_template % python knn.py 20
Training Acc: 0.928
Test Acc: 0.926
```

For example, the graph shows that as k increases, the training accuracy and testing accuracy begin to converge at around 92% accuracy starting around k=15



3e)

- I use ‘np.linalg.norm’ to calculate the Euclidean distances between ‘x_i’ and ‘self.xFeat_train’ and store them in ‘dist’. This has a complexity of O(n * d), where ‘d’ is the number of features that are being compared for each ‘n’ sample.
  Current Complexity: O(n*d)
- I then use ‘np.argsort’ to sort ‘dist’ in increasing order. This has a complexity of O(n*log(n)) because it uses quicksort to go through the ‘n’ distance in ‘dist’.
  Past Complexity: O(n*d) Current Complexity: O(n*d + n*log(n))
- Lastly, there are ‘k’ nearest neighbors that we use the sorted array to get the (1) ‘nearest_labels’ and add the relevant indices to ‘nearest_labels’ which are (2) counted up and the (3) majority count is the appended ‘predicted_label’ to ‘yHat’. This has a complexity of O(3k + 1) = O(k), because it runs ‘k’ times for these three steps and append has a complexity of O(1).
  Past Complexity: O(n*d + n*log(n)) Current Complexity: O(n*d + n*log(n) + k)

This gives me a final complexity of: O(n*d + n*log(n) + k)

4d) When using pre-processing techniques such as standard scaling and min-max scaling, we see that the accuracy is slightly better than when no pre-processing is used. This is because scaling data can help balance the impact of each variable on the distance function and can help the performance of the prediction function. This is especially important when the data starts with different units of measurement.

Additionally, adding irrelevant features does have a small effect on the accuracy of the predictions. This is because the irrelevant features can sometimes 'fool' the prediction function and can lead to worse or better predictions as shown by the graph.