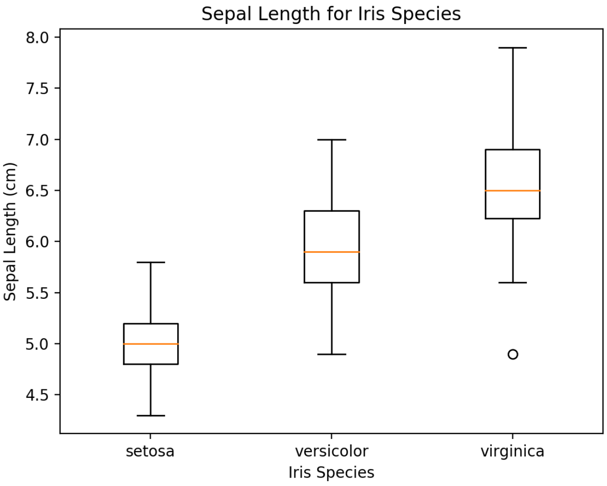
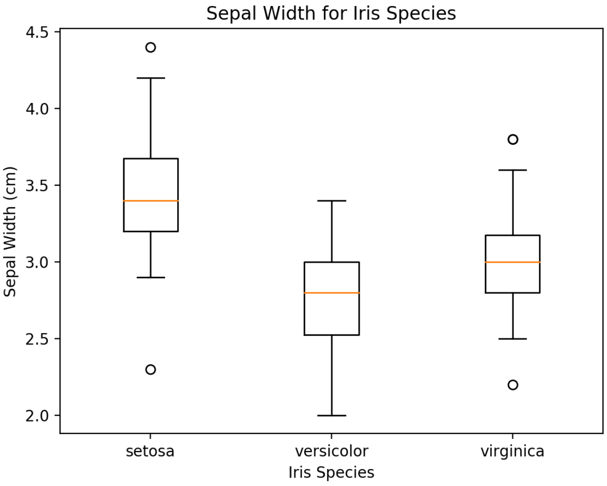
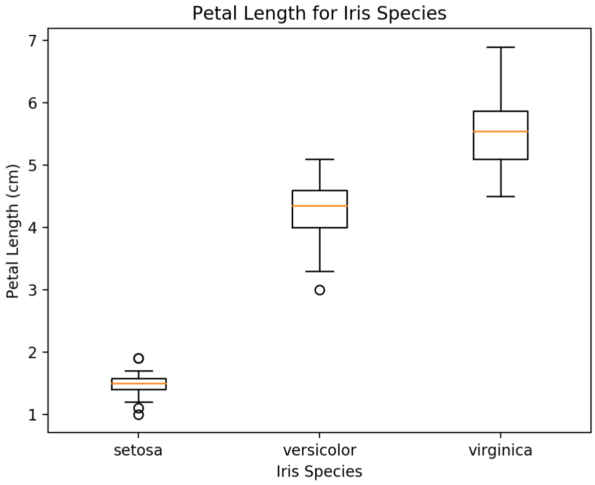
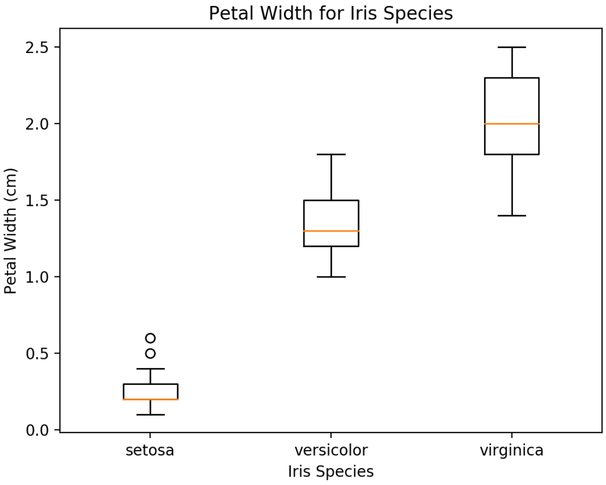
QUESTION 2B

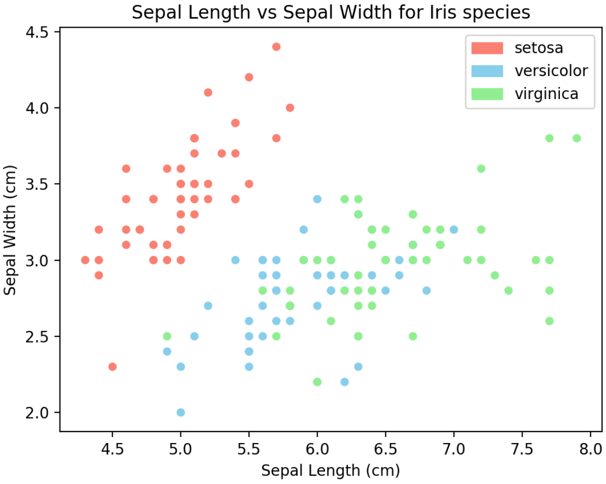

Sepal Length for Iris Species

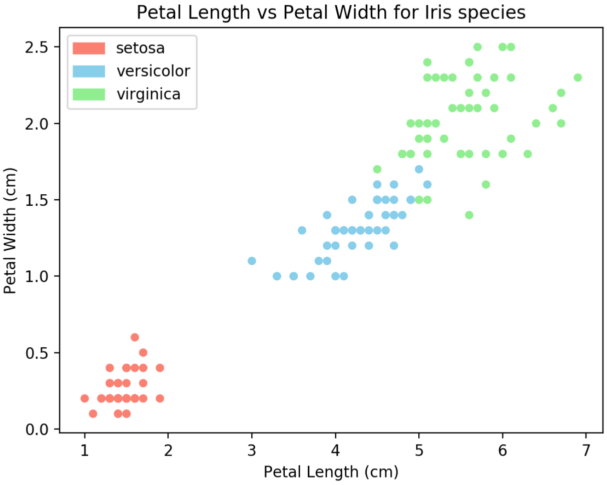

Sepal Width for Iris Species


Petal Length for Iris Species


Petal Width for Iris Species

QUESTION 2C

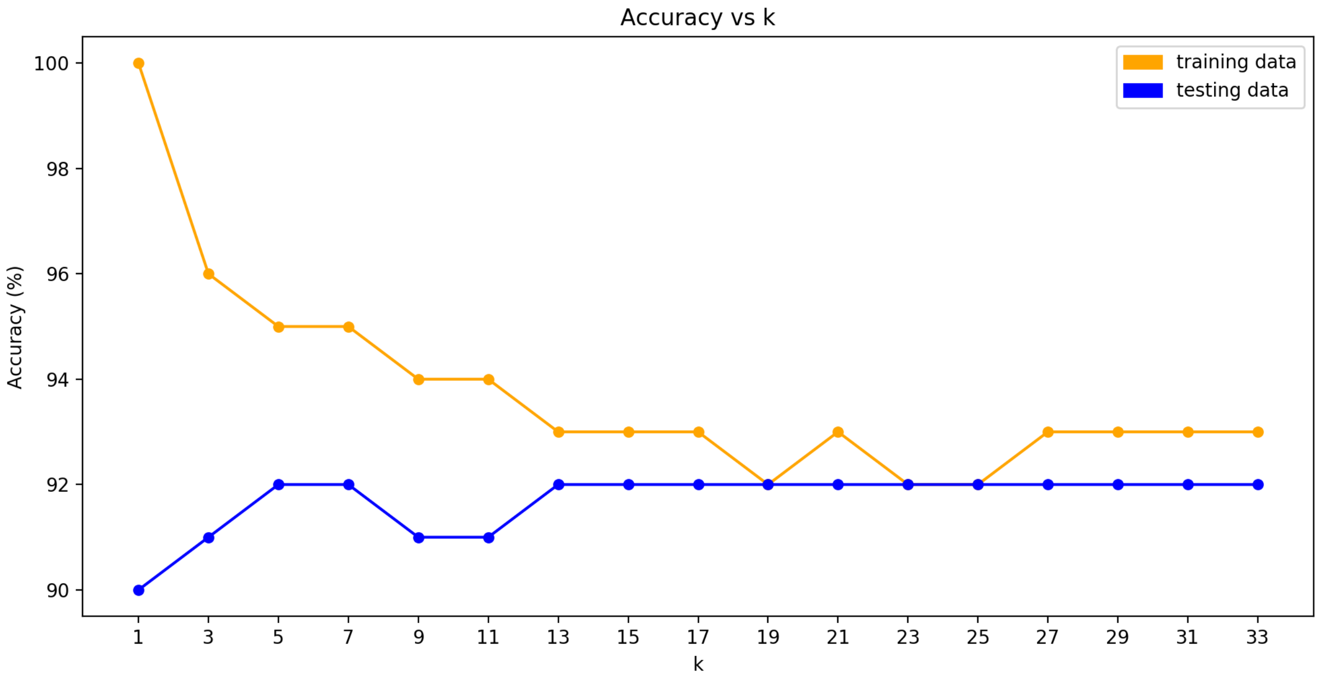

Sepal Length vs Sepal Width for Iris species


Petal Length vs Petal Width for Iris species

QUESTION 2D

Based on the plots, it is evident that the setosa species has a significantly smaller petal size as compared the alternative species. Additionally, the sepals of setosa are wider than they are longer while for the other species, sepals are longer than they are wide. It's a lot tougher to differentiate between versicolor and virginica. The boundaries between the two species aren't well defined from sepal dimensions, but looking at petal dimensions it is apparent that the petals of virginica are slightly larger than the petals of versicolor

QUESTION 3D



QUESTION 3E

- The function `predict` predicts the target value for all data points of the input feature set `xFeat`
- The function runs a prediction for each of the ***n*** data points in the input set
- Each prediction is broken up into three steps:
    1. Find the distance between the data point being classified and all pre-existing data points in the model
        - To find the distance between any two data points, the euclidian distance needs to be found, which is calculated by the function named `euclidian_distance`
        - To find the euclidian distance, one needs to find the square root of the sum of squares of differences between the feature values of the two data points

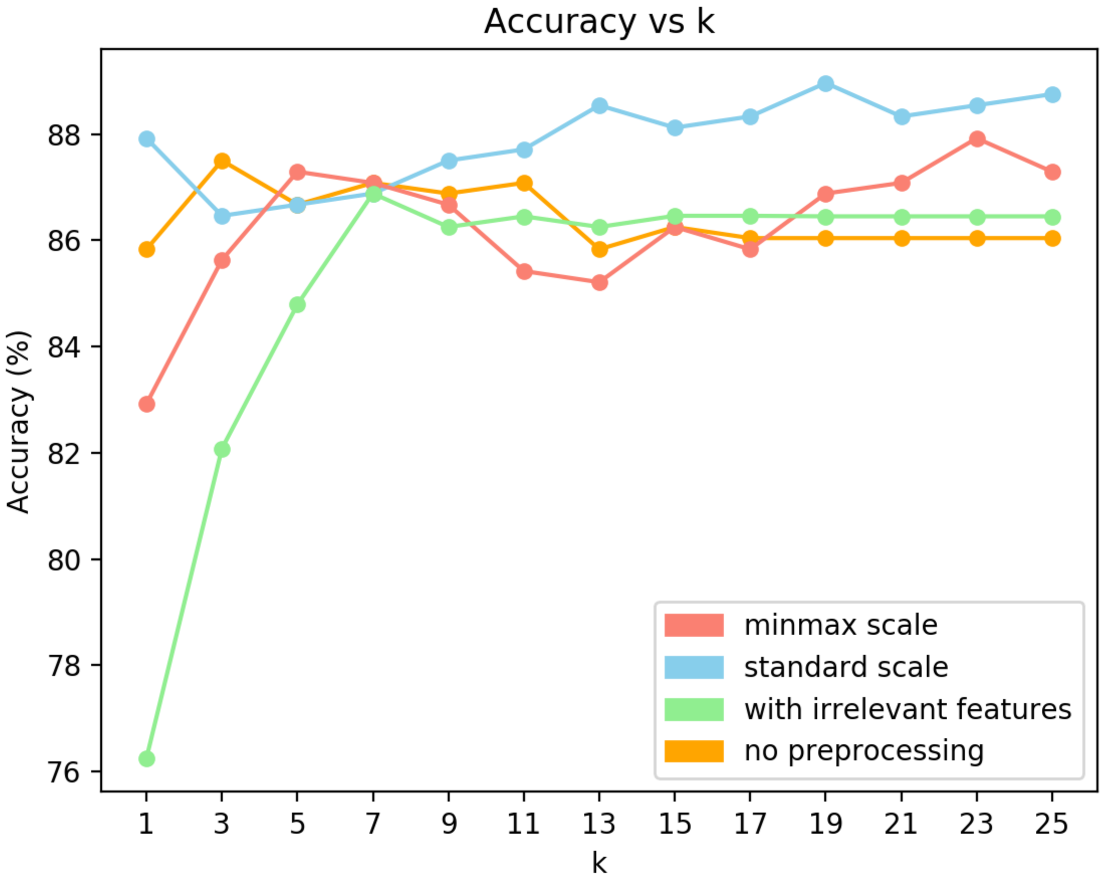
- Assuming both data points have ***d*** features, the complexity of finding the euclidian distance between the two data points is ***O(d)***
- Therefore, the complexity of finding the euclidian distance between the data point being classified and all ***n*** data points in the model is ***O(dn)***, which is stored in the list `distance`

2. Find the classes of the ***k*** data points with the shortest euclidian distance to the data point being predicted
   - The k closest data points are found using the function `find_k_min`
   - The function works by finding the index of the closest data point in the `distances` list which has a complexity of ***O(n)***
   - The function then changes its value to infinity to find the next closest data point which has a complexity of ***O(1)***
   - The function then retrieves the target value of this data point and stores it in a list named `closest`
   - The function continues finding the next closest data point until it has found the ***k*** closest data points
   - This means that all these steps are repeated k times, giving the entire function a time complexity of ***O(kn)***

3. Find the majority class from the closest ***k*** data points
   - This step requires the counts for each class to calculate the majority class
   - Since only ***k*** data points are taken into consideration the complexity of the counting is ***O(k)***

These steps run for each of the ***n*** data points needed to be classified and thus the overall time complexity of the `predict` function is ***$O((dn + kn + k) * n) = O(dn^2 + kn^2 + kn)$***

QUESTION 4E



Based on the data, the scaled data performs slightly better than the irrelevant and no-preprocessed data. Additionally, for the dataset with irrelevant features, for lower k, the irrelevant features create a lot of noise and therefore significantly decrease the accuracy since lower k is very sensitive to noise. However for higher k, it performs close to data without the irrelevant features.