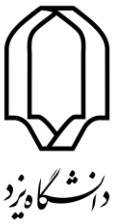
In the name of Allah



Computer Engineering Faculty of Yazd University
Artificial Intelligence

Machine Learning Course

Homework 4

(Instance-based Learning & Ensemble Learning)

Instructor: Dr. Yazdian

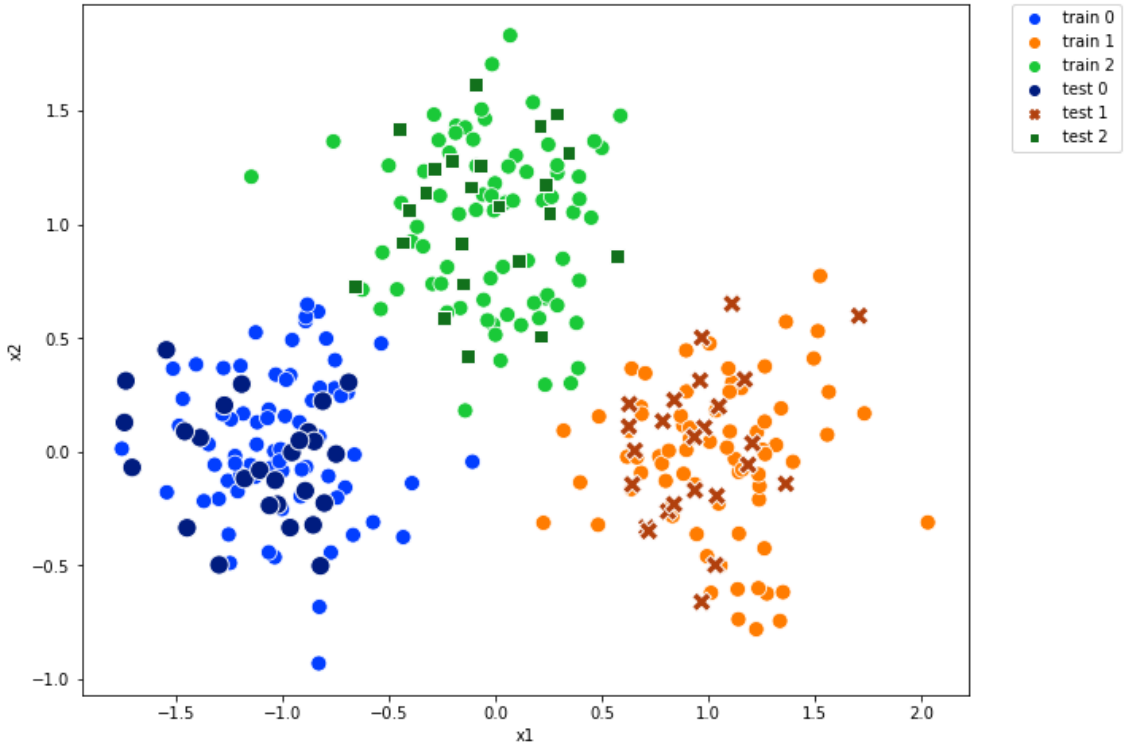
Shiva Zeymaran

Student ID: 40109434

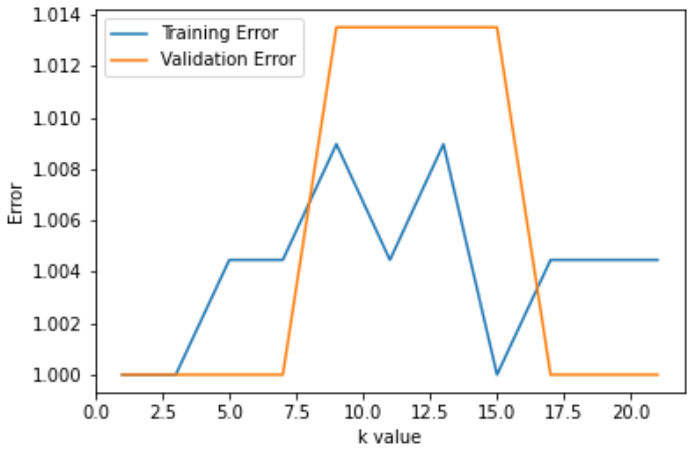
Winter 2023

Part A – KNN Algorithm for Classification

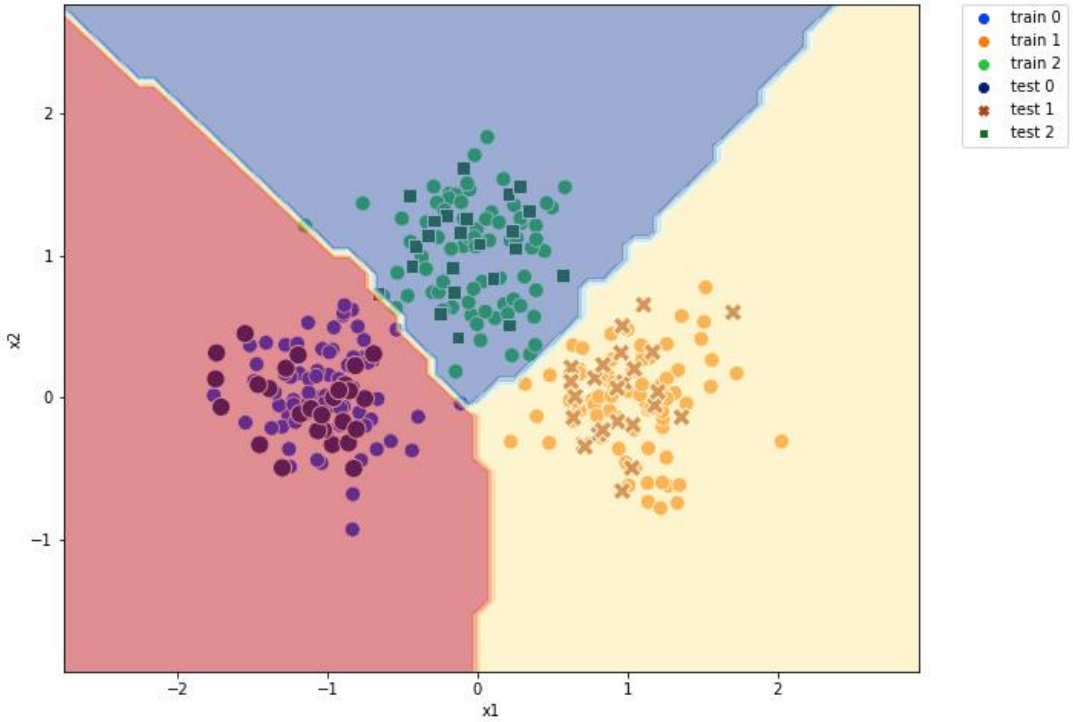
1. First, we should generate a dataset using given information and then split it to train and test sets. Finally, we visualize the data as follows; which shows samples of classes 0, 1, and 2. Also, we show test and train set samples in different shapes.



2. We classify the dataset using KNN algorithm with different values for *k*. Here, we used *1/accuracy* to find the error of the classifier. Then we calculated the train and validation errors using different *k* values. Training and test error curves vs. *k* values are as follows.
   The plot shows that by increasing the value of *k*, the training error **generally** will increase because the model will underfit. For validation error, by increasing the value of *k*, we have no different or even getting better results up to some value for *k*; but after that, the validation error will increase, too.
   As a result, the best value for *k* is when we have small *k* until the validation error not increased. Here, the best value with the last conditions happens when *k* is 7.

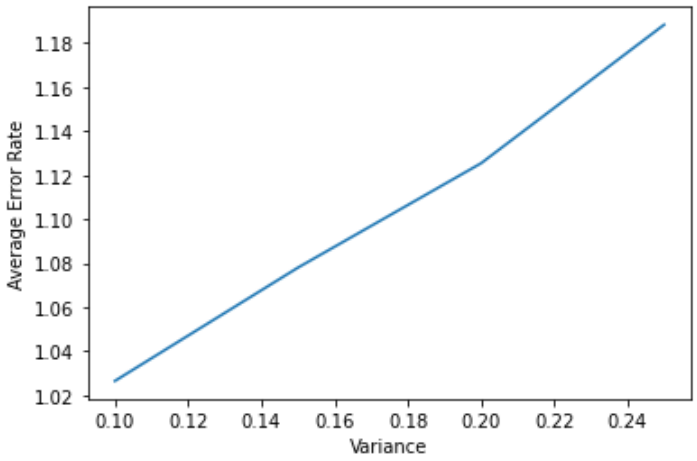


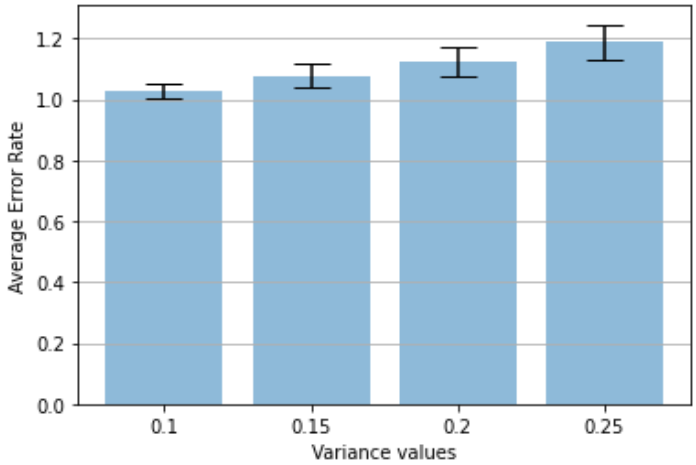
3. Plot dataset and the estimated decision boundary using $k = 7$



It is clear from this plot that the KNN algorithm works correctly for our generated dataset using the best calculated value of *k*.

4. In this part we calculate the average test error of classification over 50 generated datasets. Here, we use different variances to generate datasets; means that each 50 generated datasets have the same variance and it is different to another 50 datasets.
   We use 4 different values for variance and finally we will have 4 values for average error rate and standard deviation. We plot these values vs. *sigma* using error bars as follows.

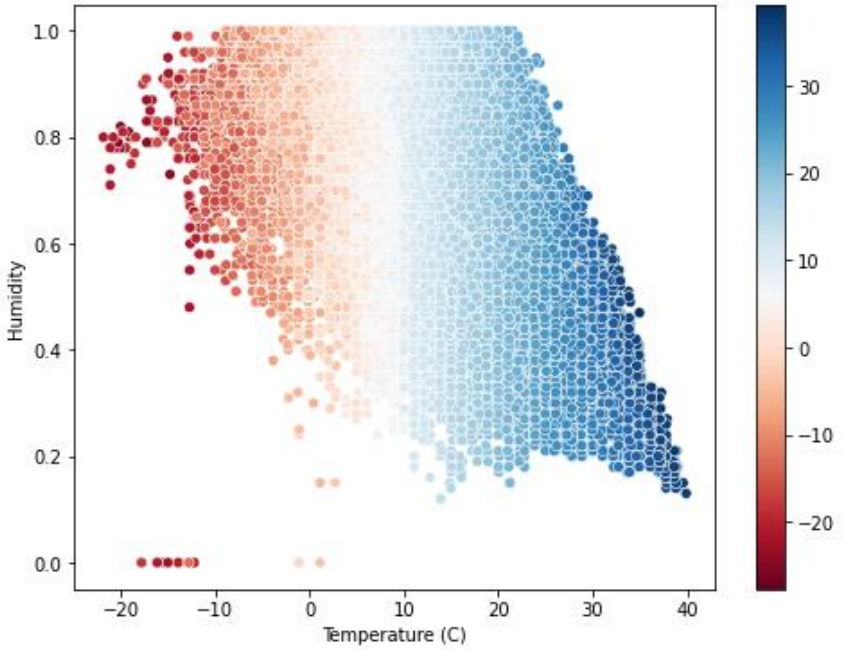




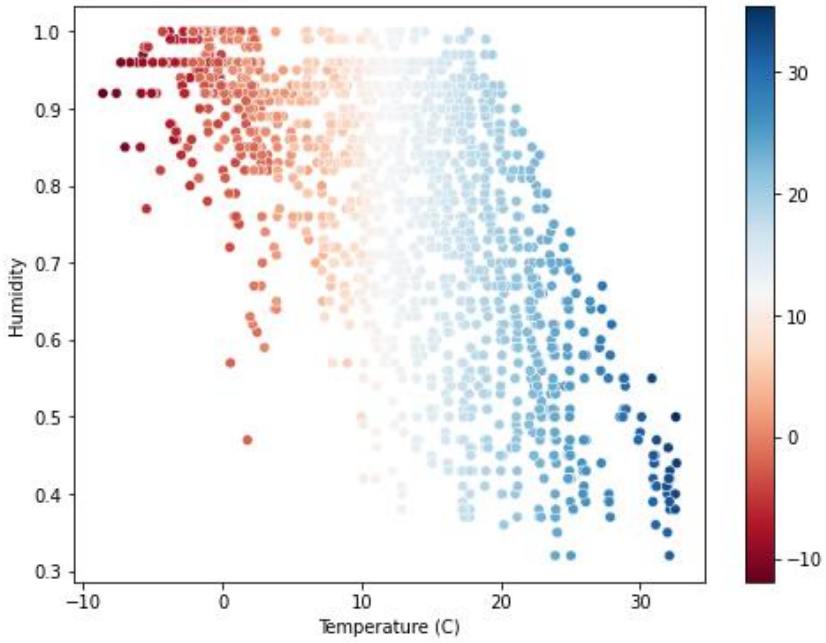
We find that by increasing the value of *sigma*, the average error rate and variance will increase, too. The reason is that increasing the *sigma* parameter means to generate a dataset with higher variance. Samples of this dataset are varying more and they are scattered. Therefore, the KNN algorithm works weaker in these cases and the error will increase. Similarly, we can explain the reason for the standard deviation increasing.

Part B – KNN Algorithm for Regression

1. Visualizing all dataset

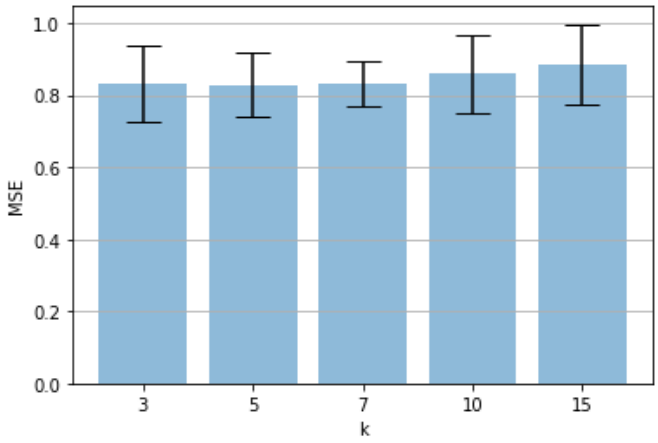


2. Visualizing the first 2000 samples of dataset



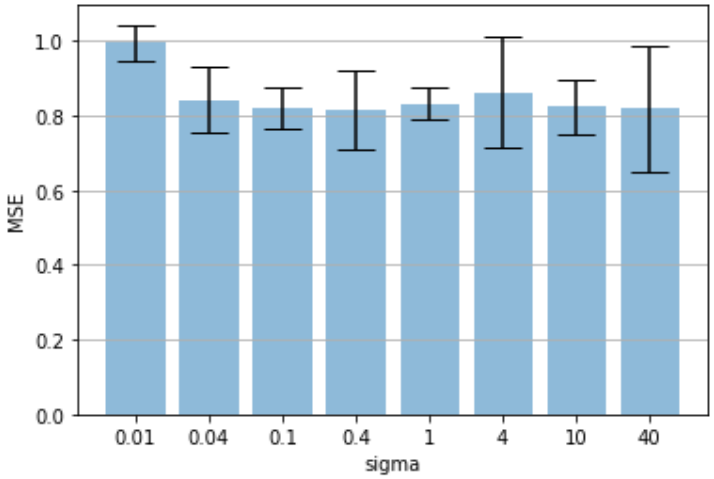
3. We try to predict the apparent temperature using KNN algorithm with $k = 1$ and 5-folds. Finally, we reached 5 values of MSE which the average of mean square errors is 1.33 and the standard deviation is 0.15.

4. In this part we repeat the last steps using 5 different values of *k*. Therefore, we reached 5 values for average mean square error and 5 values for standard deviation. Then we plot the MSE and standard deviation vs. the parameter *k* using error bars as follows.



We understand that by increasing the value of *k* parameter, MSE will decrease at first; but if the *k* value passes a threshold then the error will increase because we have underfitting. It seems that in this case, the best value for *k* is 5 because after that the MSE increases.

5. In this part we use Kernel/Weighted KNN with different values of *sigma*. Here we assume that *k* is 5 (the best value we find in the last part) and we compute the average of MSE and standard deviation after running 5-folds cross-validation.
Then we plot the MSE and standard deviation vs. the parameter *sigma* using error bars as follows.



We understand that by increasing the value of *sigma*, the error will decrease at first but then it will increase. Therefore, increasing the value of *sigma* into very big values can cause underfitting. Here, the best value of *sigma* is 0.4 with the minimum error value of 0.81.

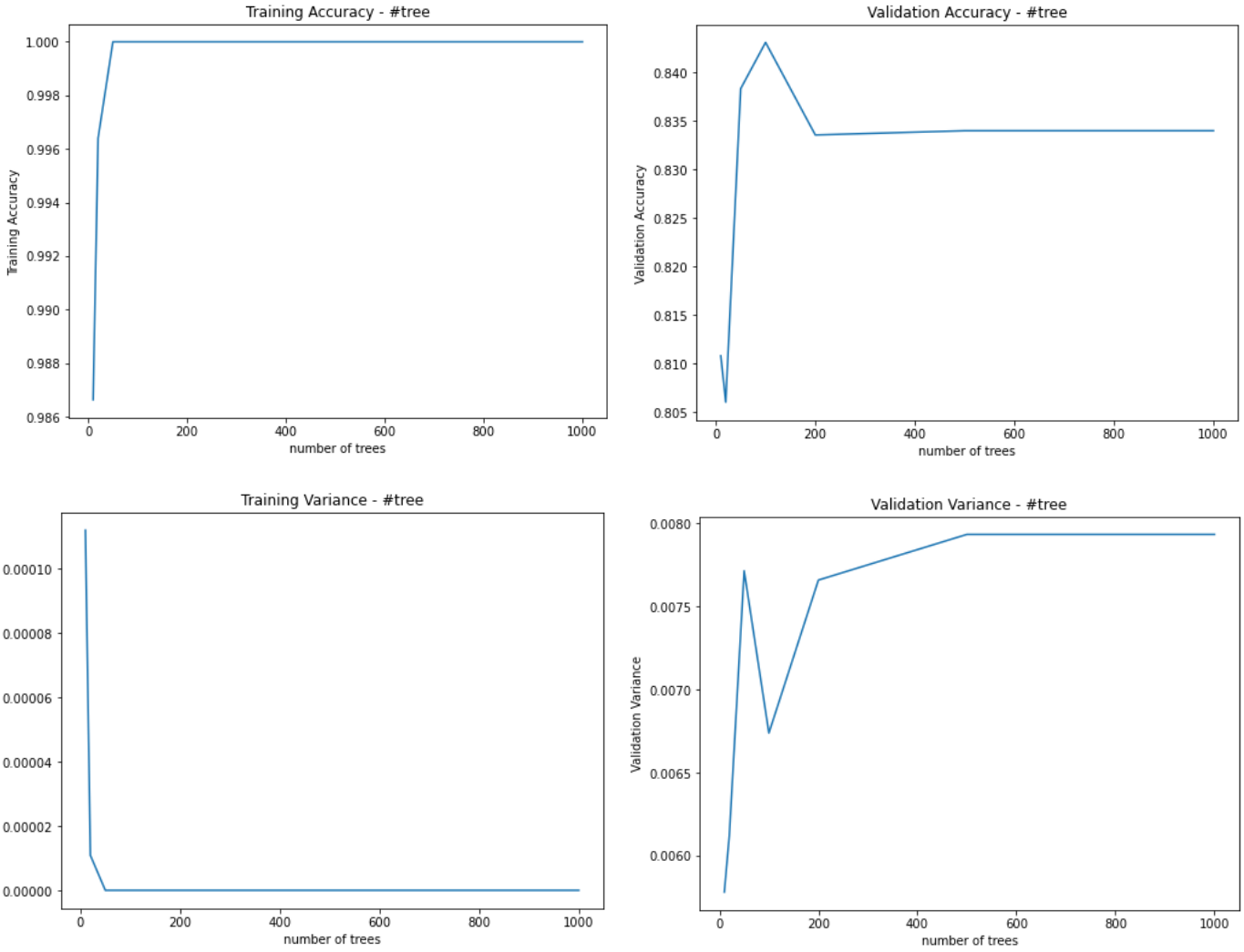
6. In comparison with KNN regression, we can understand that with fixed value of $k = 5$ the MSE was 0.829 in KNN regression but it decreases to 0.815 in Kernel KNN by using the best value of *sigma*. Therefore, the Kernel KNN algorithm performs better in this case.
   The reason is that in Kernel KNN we use the values of nearest samples with different weights (effects) rather than looking at them in the same way (not considering how much they are near to the new sample). This leads us to find the best value of new sample.

Conclusion

- In both KNN classification and KNN regression we have parameter *k* which shows the number of nearest neighbors to consider in our decision
- In both of them we should be careful about the value of *k* because increasing its value very much, can cause underfitting
- We can find the best value of *k* using cross-validation in both algorithms
- In Kernel KNN regression we also have another parameter, *sigma*, that we should set using cross-validation

Part C – Ensemble Learning

1. In this part we used Random Forest model on training data with different number of trees in the forest. We try to test 7 different values for n_estimators which are [10, 20, 50, 100, 200, 500, 1000].
   To test that which number of trees is the best, we used 10-fold cross-validation. Then we try to plot the test and validation accuracies and their corresponding variances as follows.



We can understand that by increasing the number of trees the training accuracy will increase and validation accuracy first increases and then decreased. This means that to find the best value for number of trees we should consider not to choose a very big value to keep the complexity as low as possible; and also considering the place that the highest validation accuracy reached. This happens when the number of trees is **100** in this example.

2. By using the best selected number of trees from previous part, now we can train the model using the whole training data.
   There are the results given from testing this model on our test set:

   Precision, Recall and f-measure

	precision	recall	f1-score	support
1	0.79	0.81	0.80	27
2	0.81	0.78	0.79	27
accuracy			0.80	54
macro avg	0.80	0.80	0.80	54
weighted avg	0.80	0.80	0.80	54

   Confusion matrix

```
[[22  5]
 [ 6 21]]
```

   *Accuracy = **0.79***

3. Now we learn a Decision Tree model on this training set and report the results of testing the model on the test set.

   Precision, Recall and f-measure

	precision	recall	f1-score	support
1	0.75	0.67	0.71	27
2	0.70	0.78	0.74	27
accuracy			0.72	54
macro avg	0.72	0.72	0.72	54
weighted avg	0.72	0.72	0.72	54

   Confusion matrix

```
[[18  9]
 [ 6 21]]
```

   *Accuracy = **0.72***

4. The accuracy of the Random Forest model with 100 trees is better than a single Decision tree. Also, we can understand this from reported confusion matrixes and other measures. Therefore, Random Forest works better than Decision tree.