**Q3**

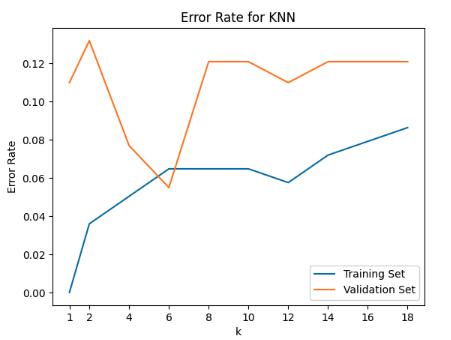
Print statements from code:

The validation error rate is 0.07692307692307693 in Problem **Set 1.1**

The validation error rate is 0.04395604395604396 in Problem Set **1.2 when using normalization**

The validation error rate is 0.04395604395604396 in Problem Set **1.2 when using minmax\_scaling**

The validation error rate is 0.04395604395604396 in Problem Set **1.3, which use cosine distance**



**In Problem Set 1.4,** we use the best k = 6 with the best validation error rate 0.054945054945054944

Using the best k, the final test error rate is 0.07100591715976332

**3.4 report:**

(1) Report and draw a curve based on the error rate of your model on the training set for each k. What do you observe? (2pts)

a. When k = 1, we observe the lowest error.

b. As k increases and finding the k-nearest neighbors (knn) becomes more complex, the error rate is higher compared to the first few iterations.

c. There is a drop at k = 12. For 6 ≤ k ≤ 20, k = 12 exhibits the lowest error.

d. The error line remains flat between k = 6 and k = 10.

e. As the value of k increases in KNN, the model may become less sensitive to local patterns and more influenced by global trends. This could impact the interpretability and generalization of the model.

g. When k=0, the error is the lowest equal to zero. However, this is not what we are looking for.

(2) Report and draw a curve based on the error rate of your model on the validation set for each k. What is your best k? (2pts) \n

a. The best k is 6 having the lowest Error Rate.

b. For 8 ≤ k ≤ 10 and 14 ≤ k ≤ 18 the error rate stays the same. Potentially, we can have an early stop after 10 if the error rate does not improve.

c. From going k=2 to k=6, the error rate improves

(3) What do you observe by comparing the difference between the two curves? (2pts) \n

a. Vladiation graph may provide better insight into the decision-making process leading to the conclusion than training graph.

b. The training curve tends to increase for 2<=k<=6 whereas the validation curve has has a drop.

d. A similar curve pattern has been observed for both training and validation curves when 8<=k<=14. However, training and validation and each k have a different error rate.

e. For k=6 (the best k - question 2) the validation error rate is lower than training error rate.

f. broadly speaking, as k increases in the training set, the error rate increrases or remains the same.

g. In the Validation graph, we have a convex function for k between 2 and 8. k=6 is the global minimum.

h. Both validation and training have a convex function for k between 10 and 14, with a local min of k=12

(4) What is the final test set error rate you get using your best-k? (1pt) \n

a. In Problem Set 1.4, we use the best k = 6 with the best validation error rate 0.054945054945054944

Using the best k, the final test error rate is 0.07100591715976332

(5) Comment on these results from the perspective of overfitting, generalization and hyper-parameter tuning. (3pts).\n

Overfitting:

When k=1, the error rate on the training set is low and higher on the validation set, this shows the model does not perform well on unseen data due to overfitting. \n

For small values of k, such as k=1 in KNN, the prediction depends solely on the nearest neighbor, which can lead to overfitting. The model becomes highly sensitive to noise in the training data, capturing local fluctuations rather than the true underlying structure of the data. \n

Generalization:

For k=6, the training and validation error rates are very close. This shows the model is generalized enough to perform the same for unseen data.

k=6 is less sensitive to a single point of data and still preserves the local pattern in the data.

k=6 provides a good balance between overfitting and underfitting in terms of generalization.

hyper-parameter tuning:

Using different Ks to find the best k is a hyper parameter tuning which we implemented in this problem. As a result, we found the best k is 6, which has the lowest error rate. \n

We learned a hyperparameter tuning is a necessary step for finding the best parameters in a classification problem.

It is crucial to have validation set in hyperparameter tuning to avoid overfitting and detect cases such as k=1, as mentioned above.

**Q4**

# **4.1**

F(w\_LS)= 217.48452613174004 on training data

F(w\_0)= 78885.82819617869 on training data

F(w\_LS)= 294.06836989399164 on testing data

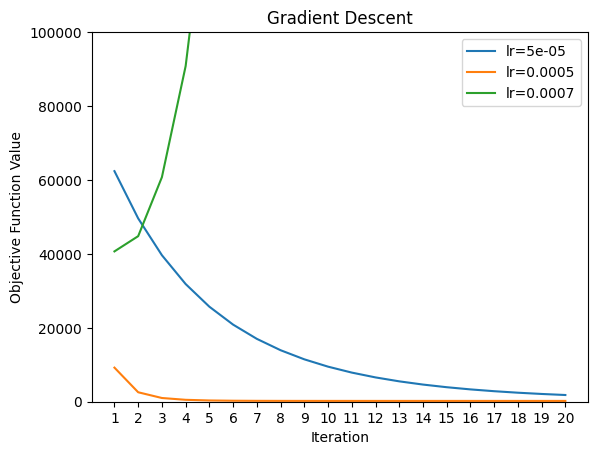
Gap: 294.068 - 217.485 = 76.583

The test objective function value with weights w\_LS performed much better than setting the weights w0 to a zero vector, showing the effectiveness of w\_LS in minimizing the residual sum of squares (RSS). However, there was still a gap between the training and test data, suggesting that the model with weights w\_LS may be slightly overfitting the training data.

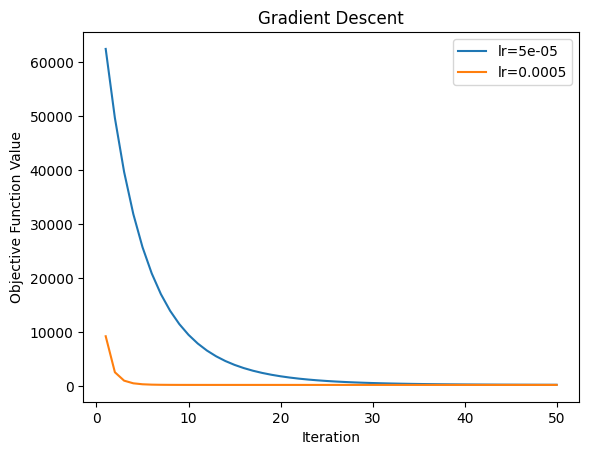
# **4.2**

A larger step size led to faster convergence in gradient descent. For instance, with a step size of 0.0005, convergence occurred in approximately seven iterations. In contrast, when the step size was reduced to 0.00005, it took over 30 iterations to approach convergence, and the objective function value remained significantly higher than that of 0.0005. However, surpassing a step size of 0.000582 resulted in a divergence of the objective function value after 100 iterations. In conclusion, the step size should be neither too small nor too large, and increasing the number of iterations does not guarantee a lower objective function value.

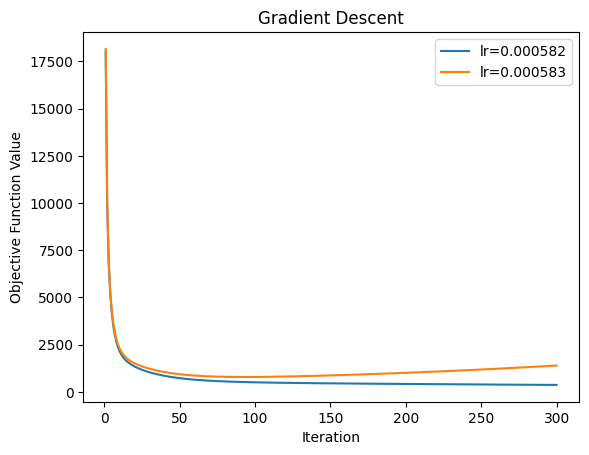
Best final objective value: 217.486, where step size=0.0005



Comparison of gradient descent convergence for various step sizes



The convergence comparison graph for step sizes of 0.00005 and 0.0005



The divergence boundary with 300 iterations

# **4.3**

Like gradient descent, stochastic gradient descent showed faster convergence with a larger step size, and the optimal step size should be neither too small nor too large. However, stochastic gradient descent had an erratic convergence behavior because only one random data point was used in each iteration instead of the entire data set. Also, we found that SGD required more iterations than GD to converge because randomly choosing a data point for each iteration resulted in noisy gradient estimates.

Best final objective value: 443.185, where step size = 0.005

