Name: Hetsvi Navnitlal
PA1

1.
Training

K	Value error
1	0.00
5	
9	
15	
3	

Validation

K	Value Error
1	
5	
9	
15	

The classifier performs best on the validation data because the error is the smallest. The test error for this classifier is

Error for 1 nn classifier is: 0.0011464968152866241

2.

Training

K	Value error
1	0.04
5	
9	
15	

Validation

K	Value Error
1	
5	
9	
15	

The classification is less accurate because of less data. The running time shortens. No it isn't because small time change is important while running it to see how expensive it would be. The run time shortens because now it only has to go through some data.

```
import pandas as pd
import numpy as np
import math
import statistics
from collections import Counter
import random


text = np.loadtxt('./pa1train.txt')
validation = np.loadtxt('./pa1validate.txt')
project = np.loadtxt('./projection.txt')
data = np.loadtxt('./pa1test.txt')


def neighbors(training, points, k):

    # Calculate the distance
    near = []
    for i in range(len(training)):
        bet = 0
        for j in range(len(training[i])):
            bet += math.pow((training[i][j] - points[j]), 2)
        near.append([training[i], math.sqrt(bet)])

    # Sorting the distance
    sort = sorted(near, key=lambda n: n[1])

    # K nearest
    sorted_near = []
    for q in range(k):
        sorted_near.append(sort[q])

    # Popular random label
    label = []
    for m in range(len(sorted_near)):
        lst = sorted_near[m][0]
        label.append(lst[len(lst)-1])

    popular = []
    counts = Counter(label)

    for i in counts.items():
        if i[1] == max(counts.values()):
            popular.append(i[0])
    return random.choice(popular)


def p(max_label, points, k):
    v = 0
    for n in range(len(points)):
            if points[n][-1] != max_label:
                v = v+1
    percent = v/ np.size(points)
    print("Error for", k, "nn classifier is: ", percent)
```

```
def projections(data, project):
    data_label = np.delete(data, len(data[0])-1, 1)
    d = np.matmul(data_label, project)
    bel = np.delete(data, [t for t in range(len(data[0])-1)], 1)
    return np.hstack((d, bel))


def main(train, vali, k):
    for e in range(len(vali)):
        print(e)
        a = neighbors(train, vali[e], k)
    p(a, vali, k)


n_text = projections(text, project)
n_validate = projections(validation, project)
n_other = projections(data, project)

main(text, text, 3)
main(text, text, 1)
main(text, text, 5)
main(text, text, 9)
main(text, text, 15)
main(text, validation, 1)
main(text, validation, 5)
main(text, validation, 9)
main(text, validation, 15)
main(n_text, n_text, 5)
main(n_text, n_text, 1)
main(n_text, n_text, 9)
main(n_text, n_text, 15)
main(n_text, n_validate, 1)
main(n_text, n_validate, 5)
main(n_text, n_validate, 9)
main(n_text, n_validate, 15)
```