p4

September 19, 2024

1 4b

implementing a function to draw M samples from distribution D

[9]:
```
import numpy as np

# set np random seed
np.random.seed(480)

# map x value to matrix A dimension
def map_xtoA(x):
    if x < 1/3:
        return 0
    elif x < 2/3:
        return 1
    else:
        return 2

# draw M samples from distribution D
def m_samples(m):
    A = np.array([[.1, .2, .2],[.2, .4, .8],[.2, .8, .9]])
    X = []
    Y = []
    for index in range(m):
        X.append(np.random.uniform(0.,1., 2))
        i = map_xtoA(X[index][0])
        j = map_xtoA(X[index][1])
        Y.append(np.random.binomial(1, A[i][j]))
    return np.array(X), np.array(Y)

# test set of 10,000 samples
X_test, Y_test = m_samples(10_000)
```

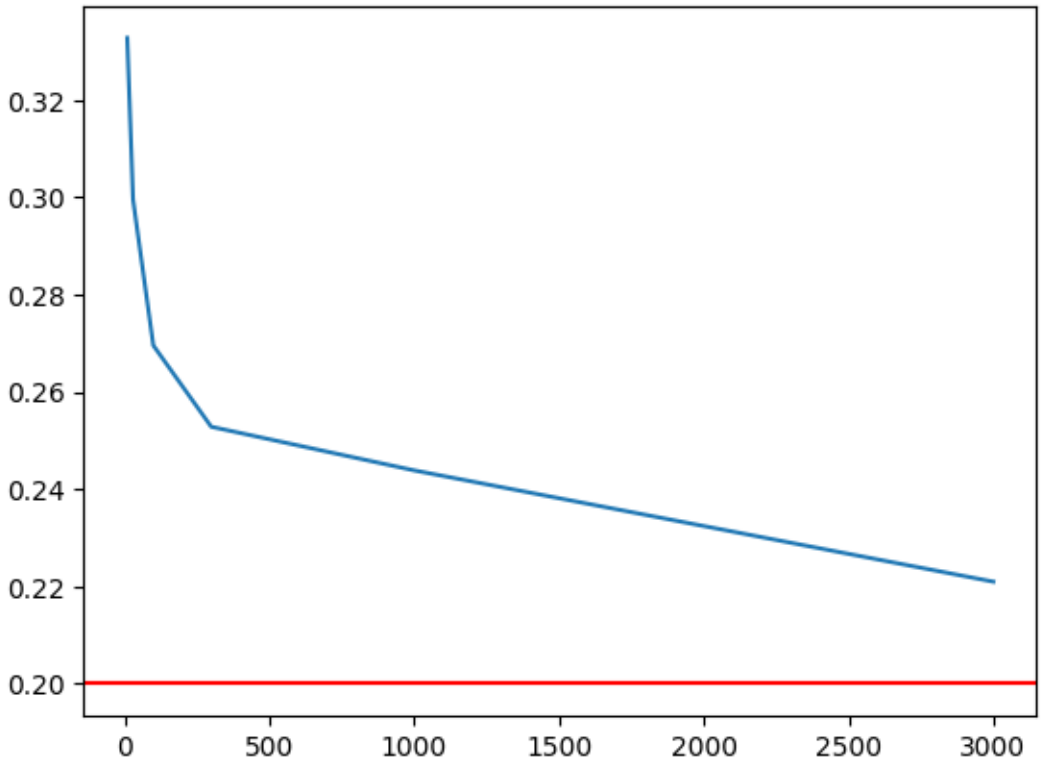
2 4c

plotting the learning curve of k-NN

[34]:
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt

np.random.seed(480)
M = [10, 30, 100, 300, 1000, 3000]
S_x, S_y = m_samples(3000)
avg_error = []
knn = KNeighborsClassifier(n_neighbors=4)

for m in M:
    index = [np.random.choice(S_x.shape[0], m, replace=False) for i in range(5)]
    x_m = [S_x[i] for i in index]
    y_m = [S_y[i] for i in index]
    error = []
    for i in range(5):
        knn.fit(x_m[i], y_m[i])
        y_pred = knn.predict(X_test)
        acc = accuracy_score(Y_test, y_pred)
        error.append(1-acc)
    avg_error.append(np.mean(error))
    error.clear()
print(avg_error)

plt.plot(M, avg_error)
plt.axhline(y=0.2, color = 'red')
```

```
[np.float64(0.33276000000000006), np.float64(0.29956),
np.float64(0.26955999999999997), np.float64(0.25284000000000006),
np.float64(0.24388), np.float64(0.221)]
```

[34]:
```
<matplotlib.lines.Line2D at 0x1690d09d0>
```



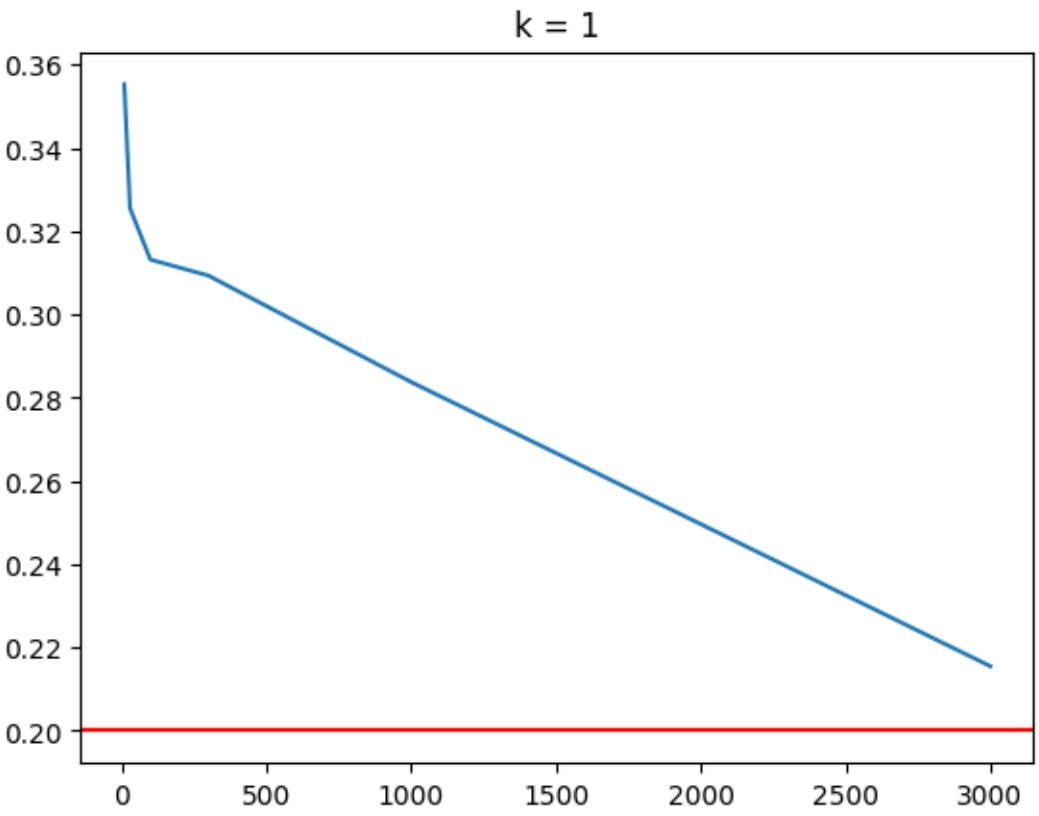
3 4d

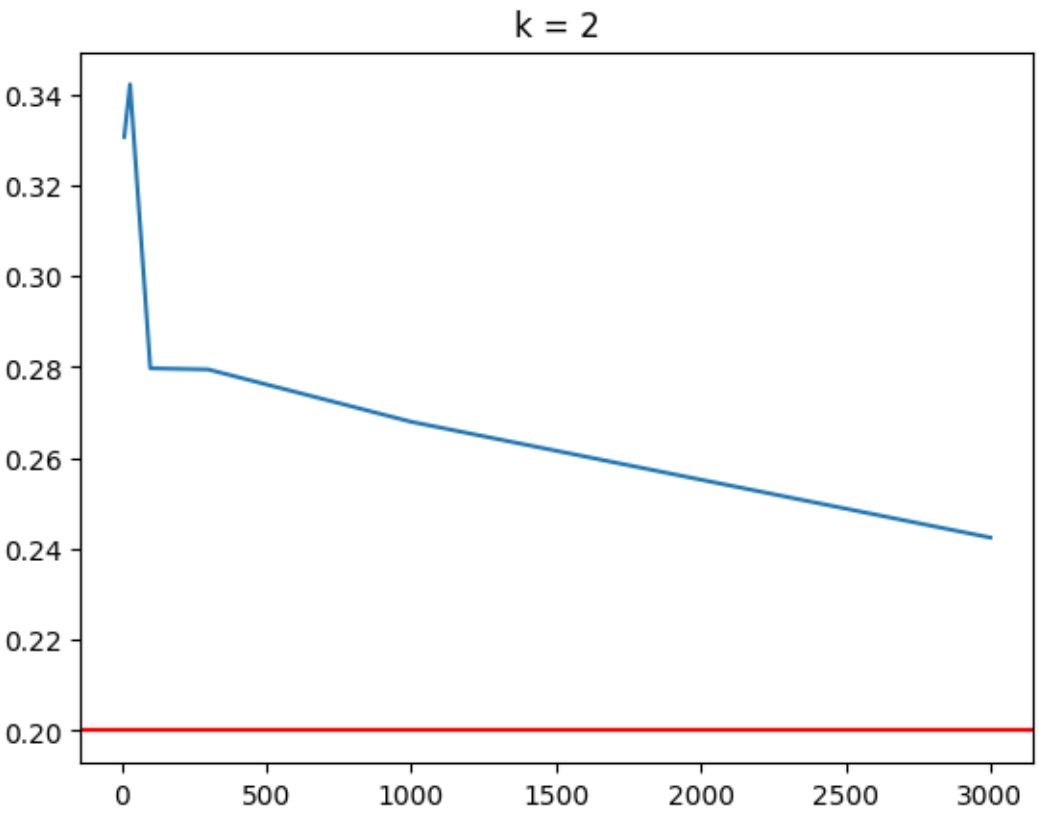
experimenting with hyperparameter k

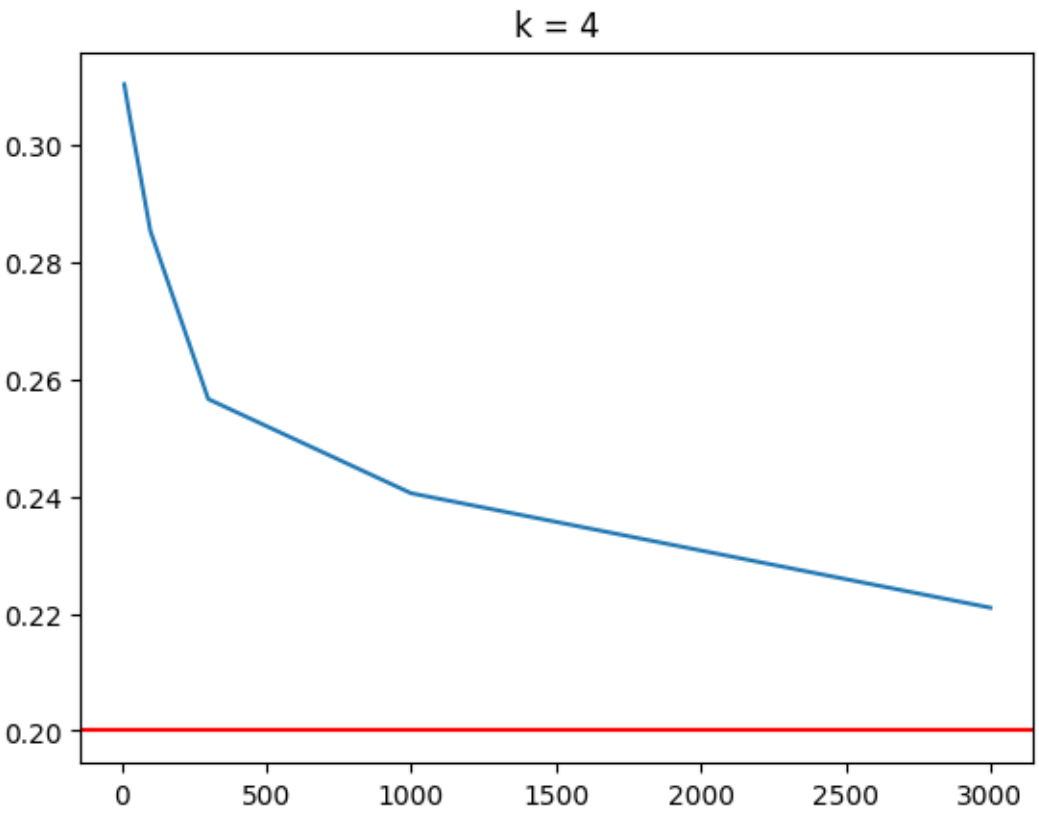
[38]:
```
K = [1, 2, 4, 8, 16, 32, 64]

np.random.seed(480)
# M = [10, 30, 100, 300, 1000, 3000]
# S_x, S_y = m_samples(3000)
avg_error = []
# knn = KNeighborsClassifier(n_neighbors=4)
for k in K:
    for m in M:
        if k > m:
            knn = KNeighborsClassifier(n_neighbors=m)
        else:
            knn = KNeighborsClassifier(n_neighbors=k)
        index = [np.random.choice(S_x.shape[0], m, replace=False) for i in␣
 ↪range(5)]
        x_m = [S_x[i] for i in index]
        y_m = [S_y[i] for i in index]
```

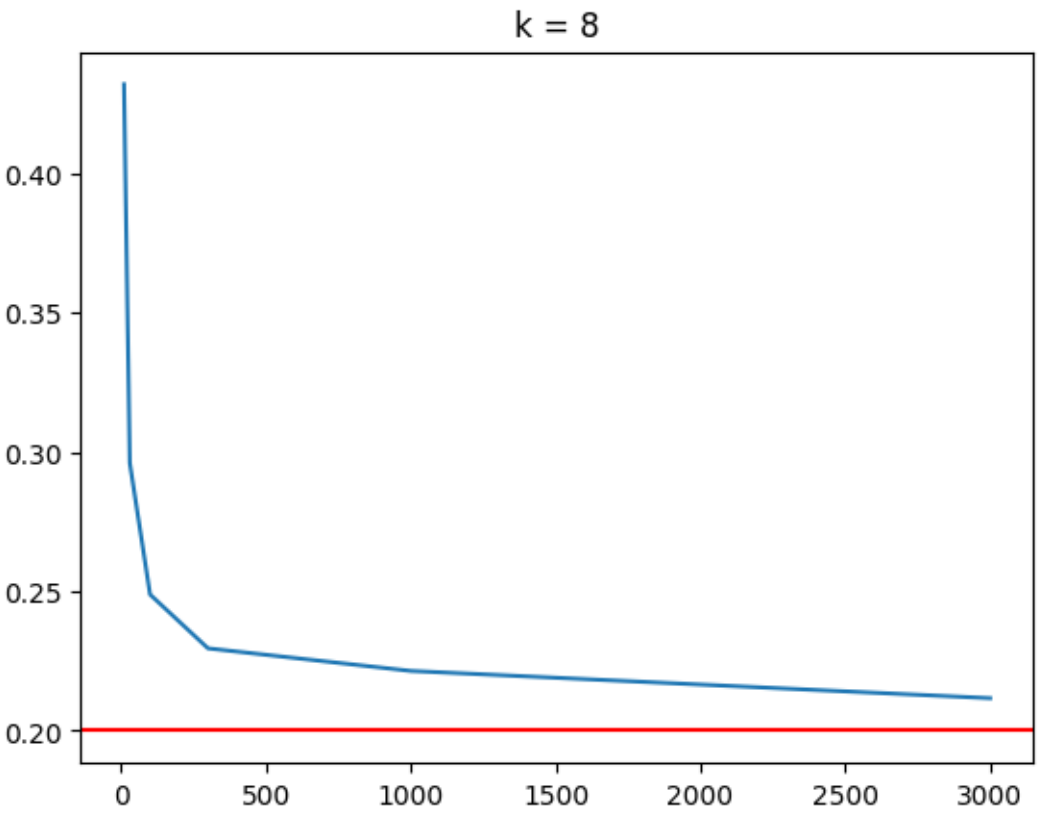
```
        error = []
        for i in range(5):
            knn.fit(x_m[i], y_m[i])
            y_pred = knn.predict(X_test)
            acc = accuracy_score(Y_test, y_pred)
            error.append(1-acc)
        avg_error.append(np.mean(error))
        error.clear()
    print(avg_error)
    plt.figure()
    plt.plot(M, avg_error)
    plt.title('k = ' + str(k))
    plt.axhline(y=0.2, color = 'red')
    avg_error.clear()
```

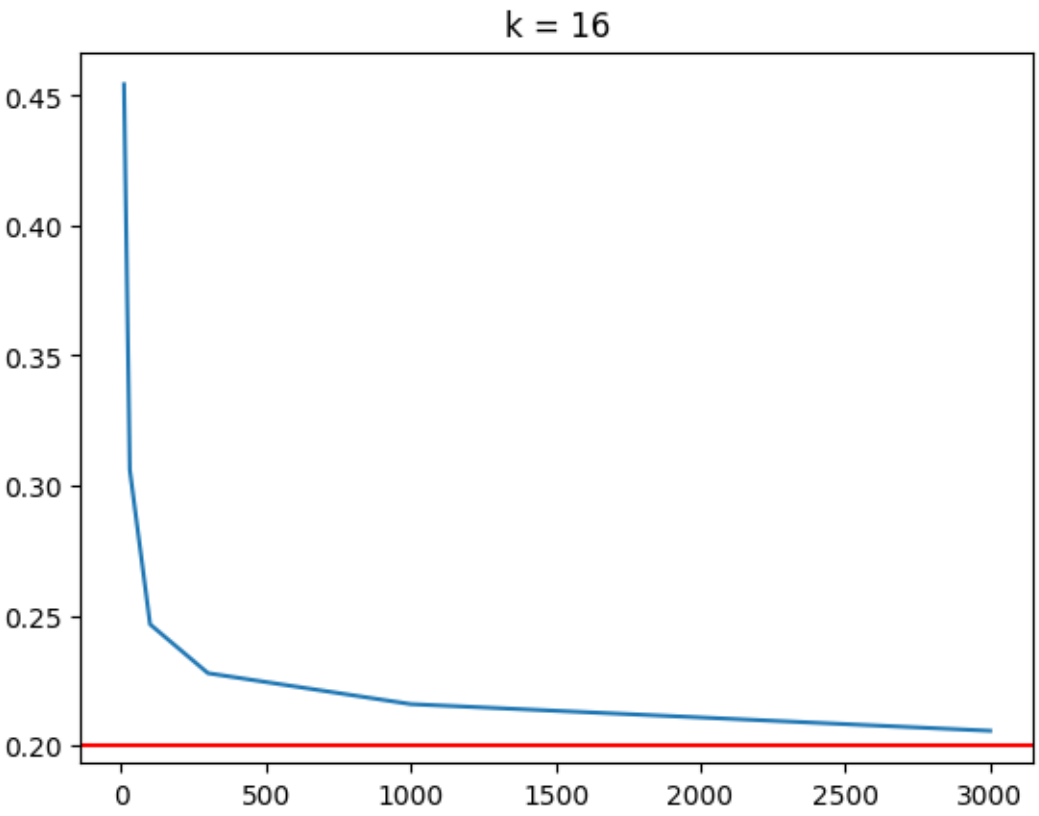
```
[np.float64(0.35527999999999993), np.float64(0.32548), np.float64(0.31318),
np.float64(0.30934), np.float64(0.28378), np.float64(0.21550000000000002)]
[np.float64(0.33064), np.float64(0.34221999999999997),
np.float64(0.27974000000000004), np.float64(0.27948000000000006),
np.float64(0.26798), np.float64(0.24250000000000008)]
[np.float64(0.31032000000000004), np.float64(0.30454), np.float64(0.28532),
np.float64(0.25660000000000005), np.float64(0.24054000000000003),
np.float64(0.221)]
[np.float64(0.43200000000000005), np.float64(0.29612),
np.float64(0.24889999999999998), np.float64(0.22958000000000003),
np.float64(0.22148000000000004), np.float64(0.2117)]
[np.float64(0.45439999999999997), np.float64(0.30606),
np.float64(0.24656000000000003), np.float64(0.22776000000000002),
np.float64(0.21586), np.float64(0.2056)]
[np.float64(0.4848), np.float64(0.42400000000000004), np.float64(0.25736),
np.float64(0.2271), np.float64(0.21606), np.float64(0.20620000000000002)]
[np.float64(0.4544), np.float64(0.45439999999999997),
np.float64(0.30751999999999996), np.float64(0.23222000000000004),
np.float64(0.21494), np.float64(0.2085)]
```

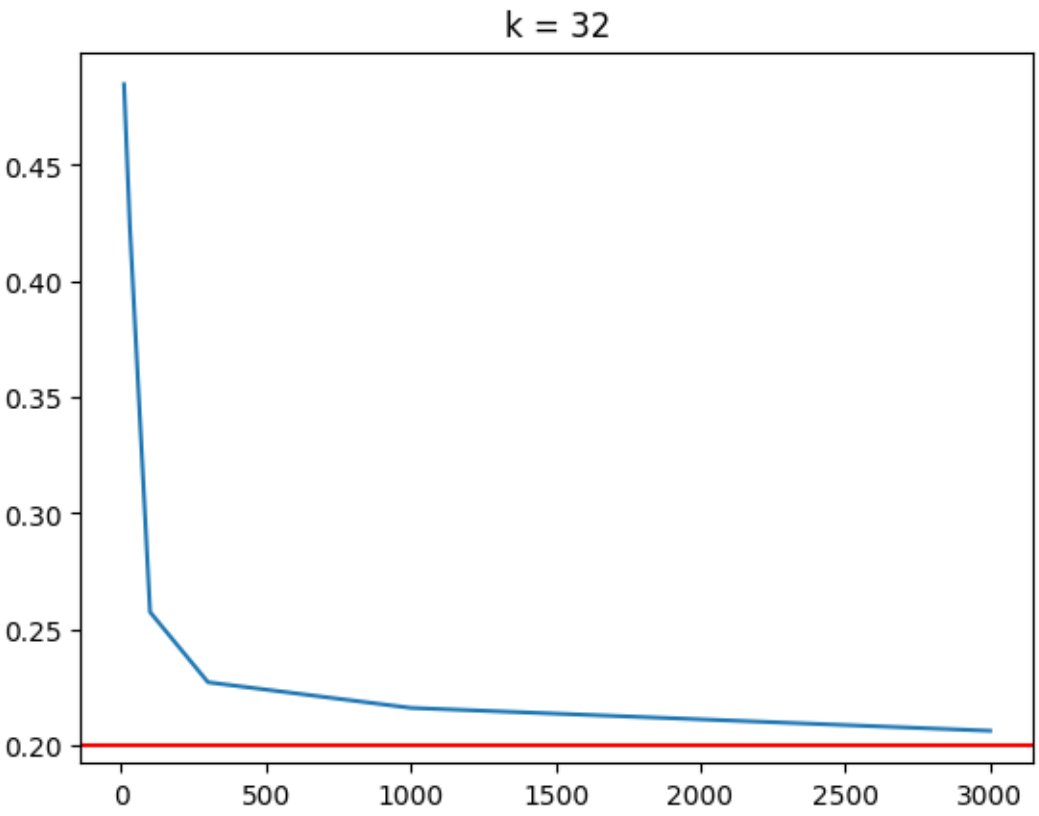

k = 1
0.36
0.34
0.32
0.30
0.28
0.26
0.24
0.22
0.20
0
500
1000
1500
2000
2500
3000

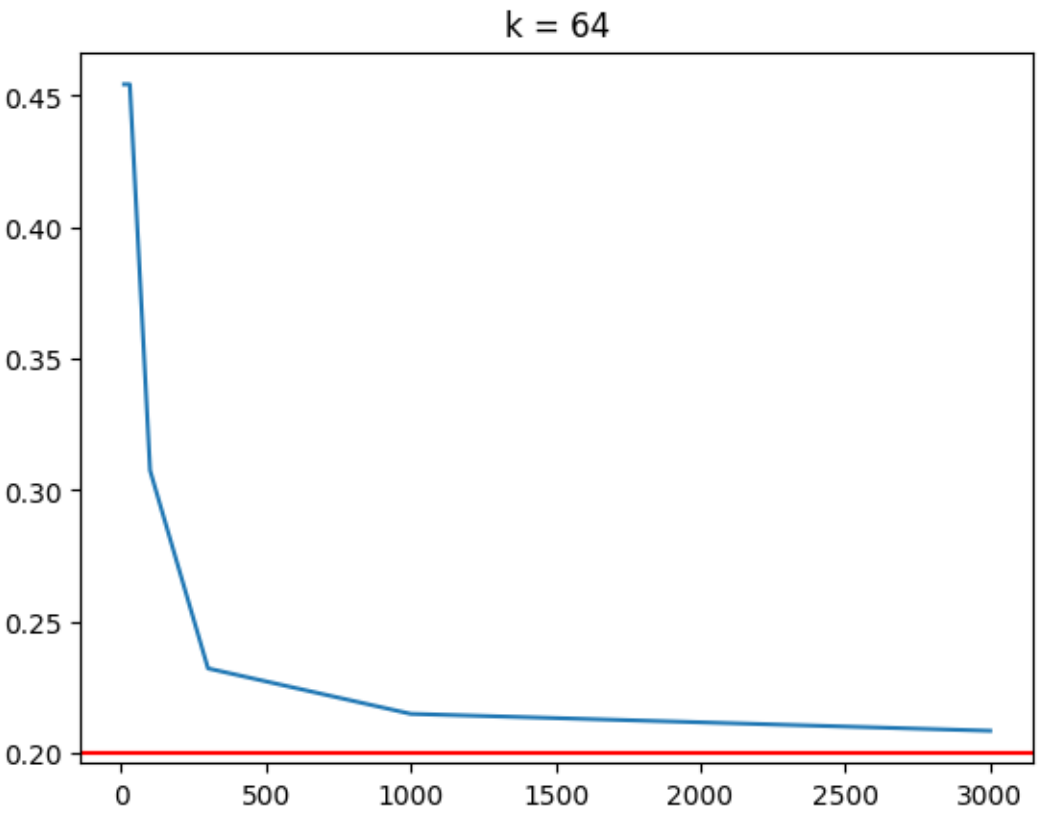

k = 2
0.34
0.32
0.30
0.28
0.26
0.24
0.22
0.20
0
500
1000
1500
2000
2500
3000


k = 4
0.30
0.28
0.26
0.24
0.22
0.20
0
500
1000
1500
2000
2500
3000


k = 8
0.40
0.35
0.30
0.25
0.20
0
500
1000
1500
2000
2500
3000


k = 16
0.45
0.40
0.35
0.30
0.25
0.20
0
500
1000
1500
2000
2500
3000


k = 32
0.45
0.40
0.35
0.30
0.25
0.20
0
500
1000
1500
2000
2500
3000


k = 64
0.45
0.40
0.35
0.30
0.25
0.20
0
500
1000
1500
2000
2500
3000