

Part 1

1. The percentage of the training data that has a positive label is 25%. The dev set percentage is 23%. I think that it is valid given that when I search the average income on google it is around 50k and I would assume that the average income is impacted by people who make a lot more than 50k.
2. Training sets
 - a. Youngest
 - i. 17
 - b. Oldest
 - i. 90
 - c. Least amount of hours
 - i. 1
 - d. Most amount of hours
 - i. 99
3. Questions
 - a. We binarize categorical fields because they represent fixed data that is not ordered in a specific way making it easier to represent them with a binary value.
 - b. We do not binarize numerical fields like age and hours per week because they are continuous and changing and the relationships between the data have meaning. An example is how working 40 hours is different than only working 30 hours. Maintaining their values better represents the data.
4. I believe that it would be beneficial to normalize because it would ensure that both numerical fields contribute equally without one overpowering the other during the learning process
5. I got 90 when I did it on the terminal and 92 when I did it using pandas.

Part 2

1. n/a
2. It is impossible to use `pandas.get_dummies()` in machine learning due to its inconsistency. It does not ensure consistent representation across the pipeline.
3. The feature dimension is 230. It does not match the result from part 1 Q % as I got 90.
4.
 - a. My best dev error rate is 15.6 and $k = 69$ achieves this error rate. I got a 1-NN of 23.2.


- i. Feature dimension: 230
- ii. 230
- iii. k=1 train_err 1.5% (+: 0.3%) dev_err 23.2% (+: 24.8%)
- iv. k=3 train_err 11.5% (+: 0.2%) dev_err 18.0% (+: 22.2%)
- v. k=5 train_err 14.4% (+: 0.2%) dev_err 16.5% (+: 19.7%)
- vi. k=7 train_err 15.5% (+: 0.2%) dev_err 16.9% (+: 20.9%)
- vii. k=9 train_err 16.6% (+: 0.2%) dev_err 16.1% (+: 19.9%)
- viii. k=11 train_err 16.9% (+: 0.2%) dev_err 16.8% (+: 18.8%)
- ix. k=13 train_err 16.7% (+: 0.2%) dev_err 16.5% (+: 18.1%)
- x. k=15 train_err 17.0% (+: 0.2%) dev_err 16.3% (+: 18.5%)
- xi. k=17 train_err 17.0% (+: 0.2%) dev_err 16.3% (+: 18.7%)
- xii. k=19 train_err 17.1% (+: 0.2%) dev_err 16.1% (+: 18.5%)
- xiii. k=21 train_err 17.2% (+: 0.2%) dev_err 16.2% (+: 18.8%)
- xiv. k=23 train_err 17.5% (+: 0.2%) dev_err 16.0% (+: 18.2%)
- xv. k=25 train_err 17.4% (+: 0.2%) dev_err 16.2% (+: 17.6%)
- xvi. k=27 train_err 17.5% (+: 0.2%) dev_err 16.2% (+: 17.6%)
- xvii. k=29 train_err 17.8% (+: 0.2%) dev_err 16.4% (+: 17.8%)
- xviii. k=31 train_err 17.7% (+: 0.2%) dev_err 15.8% (+: 17.8%)
- xix. k=33 train_err 17.6% (+: 0.2%) dev_err 16.0% (+: 18.6%)
- xx. k=35 train_err 17.7% (+: 0.2%) dev_err 15.9% (+: 17.9%)
- xxi. k=37 train_err 17.7% (+: 0.2%) dev_err 16.2% (+: 17.8%)
- xxii. k=39 train_err 17.4% (+: 0.2%) dev_err 15.8% (+: 18.4%)
- xxiii. k=41 train_err 17.7% (+: 0.2%) dev_err 15.9% (+: 18.5%)
- xxiv. k=43 train_err 17.5% (+: 0.2%) dev_err 16.1% (+: 18.3%)
- xxv. k=45 train_err 17.5% (+: 0.2%) dev_err 16.3% (+: 18.1%)
- xxvi. k=47 train_err 17.8% (+: 0.2%) dev_err 16.4% (+: 18.0%)
- xxvii. k=49 train_err 17.9% (+: 0.2%) dev_err 16.0% (+: 17.6%)
- xxviii. k=51 train_err 17.7% (+: 0.2%) dev_err 16.0% (+: 18.2%)
- xxix. k=53 train_err 17.9% (+: 0.2%) dev_err 15.9% (+: 17.7%)
- xxx. k=55 train_err 17.8% (+: 0.2%) dev_err 16.4% (+: 17.8%)
- xxxi. k=57 train_err 17.8% (+: 0.2%) dev_err 16.1% (+: 17.7%)
- xxxii. k=59 train_err 17.9% (+: 0.2%) dev_err 16.3% (+: 18.1%)
- xxxiii. k=61 train_err 17.8% (+: 0.2%) dev_err 16.0% (+: 17.8%)
- xxxiv. k=63 train_err 18.3% (+: 0.2%) dev_err 16.1% (+: 17.9%)
- xxxv. k=65 train_err 18.2% (+: 0.2%) dev_err 16.4% (+: 17.6%)
- xxxvi. k=67 train_err 18.0% (+: 0.2%) dev_err 16.2% (+: 17.8%)
- xxxvii. k=69 train_err 18.1% (+: 0.2%) dev_err 15.6% (+: 17.6%)
- xxxviii. k=71 train_err 18.0% (+: 0.2%) dev_err 16.1% (+: 17.1%)

- xxxix. k=73 train_err 18.2% (+: 0.2%) dev_err 16.1% (+: 17.7%)
 - xl. k=75 train_err 18.4% (+: 0.2%) dev_err 16.1% (+: 17.7%)
 - xli. k=77 train_err 18.0% (+: 0.2%) dev_err 16.3% (+: 17.7%)
 - xlvi. k=79 train_err 18.3% (+: 0.2%) dev_err 15.9% (+: 17.3%)
 - xlvi. k=81 train_err 17.9% (+: 0.2%) dev_err 16.2% (+: 17.4%)
 - xlvi. k=83 train_err 18.2% (+: 0.2%) dev_err 16.0% (+: 17.2%)
 - xlvi. k=85 train_err 18.0% (+: 0.2%) dev_err 16.5% (+: 17.3%)
 - xlvi. k=87 train_err 18.1% (+: 0.2%) dev_err 16.4% (+: 17.4%)
 - xlvi. k=89 train_err 18.0% (+: 0.2%) dev_err 16.1% (+: 17.3%)
 - xlvi. k=91 train_err 18.1% (+: 0.2%) dev_err 15.8% (+: 17.4%)
 - xlvi. k=93 train_err 18.1% (+: 0.2%) dev_err 16.1% (+: 17.3%)
 - l. k=95 train_err 18.2% (+: 0.2%) dev_err 16.0% (+: 17.4%)
 - li. k=97 train_err 18.4% (+: 0.2%) dev_err 15.7% (+: 17.5%)
 - lii. k=99 train_err 18.1% (+: 0.2%) dev_err 15.7% (+: 17.3%)
 - liii. Best dev error rate: 15.6% achieved with k=69

- b. When k = 1 my training error is not 0%. My data produced a training error of 1.5%. After doing some research and looking on Ed, I learned that k could the training error could not be 0% for a few reasons. The reasons include, duplicate points with different labels, noise, tie-breaking k-NN, and more. I believe that there are duplicate data points, so I would assume that it is one of the reasons that is causing the training error to not be 0% in my code.
- c. The trends that I observed are that as k increases, the training error seemed to increase, and the dev error seemed to stay around the 15% and 16% range. The positive rate started out higher and then leveled out towards the end. The running speed stayed constant (from what I observed), but I assumed the larger k got the slower it would get.
- d. K infinity classifies every point and results in extreme underfitting due to the model ignoring the individual. K = 1 results in overfitting because each data point is classified based solely on the closest neighbor.

5114400

0.190819s



Your Best Entry!
Your most recent submission scored 0.190, which is the same as your previous score. Keep trying!

- e. My ranking on the public leaderboard is 51 and the error rate is 0.190. My positive rate is 17.1%.

1. I got a dev error rate of 20.7% achieved with a k of 33. The feature dimension is 92. I got a 1-NN of 26.7%. While my dev error rate is higher than what was achieved during part 2 Q 4, I did achieve my best dev error rate at a smaller k than I did in the previous question. Therefore, there was an improvement in performance when compared to part 2 question 4.

a. Feature dimension: 92

b. k=1 train_err 1.5% (+: 0.3%) dev_err 26.9% (+: 27.3%)

c. k=3 train_err 13.0% (+: 0.2%) dev_err 24.0% (+: 25.0%)

d. k=5 train_err 15.6% (+: 0.2%) dev_err 23.4% (+: 25.6%)

e. k=7 train_err 16.8% (+: 0.2%) dev_err 23.3% (+: 23.9%)

f. k=9 train_err 18.3% (+: 0.2%) dev_err 22.2% (+: 22.2%)

g. k=11 train_err 18.8% (+: 0.2%) dev_err 21.9% (+: 22.1%)

h. k=13 train_err 19.0% (+: 0.2%) dev_err 21.7% (+: 20.9%)

i. k=15 train_err 19.2% (+: 0.2%) dev_err 21.5% (+: 20.9%)

j. k=17 train_err 19.5% (+: 0.2%) dev_err 22.1% (+: 20.1%)

k. k=19 train_err 19.7% (+: 0.2%) dev_err 21.4% (+: 19.6%)

l. k=21 train_err 19.8% (+: 0.2%) dev_err 21.7% (+: 19.7%)

m. k=23 train_err 20.3% (+: 0.2%) dev_err 22.3% (+: 19.7%)

n. k=25 train_err 20.1% (+: 0.2%) dev_err 22.1% (+: 19.1%)

o. k=27 train_err 20.8% (+: 0.2%) dev_err 22.0% (+: 19.2%)

p. k=29 train_err 21.5% (+: 0.2%) dev_err 21.3% (+: 18.1%)

q. k=31 train_err 20.9% (+: 0.2%) dev_err 21.4% (+: 18.8%)

r. k=33 train_err 21.4% (+: 0.2%) dev_err 20.7% (+: 17.5%)

s. k=35 train_err 21.4% (+: 0.2%) dev_err 21.3% (+: 17.1%)

t. k=37 train_err 21.4% (+: 0.2%) dev_err 21.7% (+: 17.5%)

u. k=39 train_err 21.7% (+: 0.2%) dev_err 21.8% (+: 18.4%)

v. k=41 train_err 21.7% (+: 0.2%) dev_err 21.8% (+: 17.2%)

w. k=43 train_err 21.6% (+: 0.2%) dev_err 22.3% (+: 17.7%)

x. k=45 train_err 21.9% (+: 0.2%) dev_err 22.3% (+: 17.1%)

y. k=47 train_err 21.9% (+: 0.2%) dev_err 22.7% (+: 16.5%)

z. k=49 train_err 22.1% (+: 0.1%) dev_err 22.0% (+: 15.8%)

aa. k=51 train_err 22.3% (+: 0.1%) dev_err 22.1% (+: 15.7%)

bb. k=53 train_err 22.6% (+: 0.1%) dev_err 22.4% (+: 14.0%)

cc. k=55 train_err 22.8% (+: 0.1%) dev_err 22.3% (+: 14.5%)

dd. k=57 train_err 22.8% (+: 0.1%) dev_err 21.8% (+: 14.2%)

ee. k=59 train_err 22.8% (+: 0.1%) dev_err 21.9% (+: 13.9%)

ff. k=61 train_err 23.0% (+: 0.1%) dev_err 22.4% (+: 13.8%)

gg. k=63 train_err 22.8% (+: 0.1%) dev_err 22.7% (+: 13.7%)

hh. k=65 train_err 22.9% (+: 0.1%) dev_err 22.6% (+: 14.0%)
 ii. k=67 train_err 23.1% (+: 0.1%) dev_err 22.2% (+: 13.6%)
 jj. k=69 train_err 23.1% (+: 0.1%) dev_err 22.2% (+: 13.0%)
 kk. k=71 train_err 23.0% (+: 0.1%) dev_err 22.9% (+: 12.1%)
 ll. k=73 train_err 22.9% (+: 0.1%) dev_err 22.1% (+: 12.1%)
 mm. k=75 train_err 23.1% (+: 0.1%) dev_err 22.5% (+: 11.9%)
 nn. k=77 train_err 23.0% (+: 0.1%) dev_err 22.0% (+: 11.2%)
 oo. k=79 train_err 23.3% (+: 0.1%) dev_err 22.3% (+: 11.7%)
 pp. k=81 train_err 23.2% (+: 0.1%) dev_err 21.9% (+: 11.5%)
 qq. k=83 train_err 23.0% (+: 0.1%) dev_err 22.4% (+: 11.0%)
 rr. k=85 train_err 23.4% (+: 0.1%) dev_err 21.9% (+: 11.1%)
 ss. k=87 train_err 23.6% (+: 0.1%) dev_err 22.2% (+: 11.0%)
 tt. k=89 train_err 23.5% (+: 0.1%) dev_err 22.6% (+: 10.8%)
 uu. k=91 train_err 23.3% (+: 0.1%) dev_err 22.5% (+: 10.1%)
 vv. k=93 train_err 23.6% (+: 0.1%) dev_err 22.8% (+: 10.4%)
 ww. k=95 train_err 23.3% (+: 0.1%) dev_err 22.4% (+: 11.0%)
 xx. k=97 train_err 23.3% (+: 0.1%) dev_err 22.5% (+: 10.9%)
 yy. k=99 train_err 23.1% (+: 0.1%) dev_err 23.1% (+: 10.9%)
 zz. Best dev error rate: 20.7% achieved with k=33
 aaa. Your file has passed the formatting test! Your positive rate is 11.8%.
 Your positive rate seems reasonable. This does not guarantee that your
 prediction accuracy will be good though.


2. My new 1-NN dev error is 23.9% and my best dev error is 14.4% which was achieved with a k of 41. For this implementation, my best dev error lowered, but it took me a little longer to get the desired dev error than it did for 3 part 1. It did take me a shorter amount of time than it did for 2 q 4. This marks an improvement in performance from what was accomplished in 2 q4. Part 3 question 2 incorporated the MinMaxScaler function, which I believe has helped with the improvement in performance. MinMaxScaler standardizes the features by scaling them to a given range allowing the performance to increase from what was achieved during part 2.
 - a. Feature dimension: 92
 - b. k=1 train_err 1.5% (+: 0.3%) dev_err 23.9% (+: 27.1%)
 - c. k=3 train_err 11.5% (+: 0.2%) dev_err 19.2% (+: 26.0%)
 - d. k=5 train_err 13.8% (+: 0.2%) dev_err 18.3% (+: 24.7%)
 - e. k=7 train_err 14.1% (+: 0.2%) dev_err 16.9% (+: 24.1%)
 - f. k=9 train_err 15.5% (+: 0.2%) dev_err 15.9% (+: 22.3%)
 - g. k=11 train_err 16.3% (+: 0.2%) dev_err 16.3% (+: 21.1%)
 - h. k=13 train_err 16.3% (+: 0.2%) dev_err 16.4% (+: 21.8%)

i. k=15 train_err 16.4% (+: 0.2%) dev_err 15.7% (+: 21.9%)
j. k=17 train_err 16.6% (+: 0.2%) dev_err 15.7% (+: 21.9%)
k. k=19 train_err 16.8% (+: 0.2%) dev_err 16.4% (+: 20.8%)
l. k=21 train_err 17.1% (+: 0.2%) dev_err 15.9% (+: 21.1%)
m. k=23 train_err 17.2% (+: 0.2%) dev_err 15.4% (+: 21.6%)
n. k=25 train_err 17.0% (+: 0.2%) dev_err 15.0% (+: 21.2%)
o. k=27 train_err 16.9% (+: 0.2%) dev_err 15.6% (+: 20.8%)
p. k=29 train_err 17.1% (+: 0.2%) dev_err 15.3% (+: 20.9%)
q. k=31 train_err 17.1% (+: 0.2%) dev_err 15.3% (+: 21.1%)
r. k=33 train_err 17.2% (+: 0.2%) dev_err 15.6% (+: 21.2%)
s. k=35 train_err 17.2% (+: 0.2%) dev_err 15.0% (+: 20.6%)
t. k=37 train_err 17.1% (+: 0.2%) dev_err 14.8% (+: 20.8%)
u. k=39 train_err 17.3% (+: 0.2%) dev_err 14.6% (+: 20.8%)
v. k=41 train_err 17.4% (+: 0.2%) dev_err 14.4% (+: 20.6%)
w. k=43 train_err 17.4% (+: 0.2%) dev_err 14.8% (+: 20.4%)
x. k=45 train_err 17.6% (+: 0.2%) dev_err 15.0% (+: 20.4%)
y. k=47 train_err 17.9% (+: 0.2%) dev_err 15.3% (+: 19.9%)
z. k=49 train_err 18.0% (+: 0.2%) dev_err 15.4% (+: 20.2%)
aa. k=51 train_err 18.0% (+: 0.2%) dev_err 15.7% (+: 19.9%)
bb. k=53 train_err 18.0% (+: 0.2%) dev_err 15.4% (+: 19.8%)
cc. k=55 train_err 18.0% (+: 0.2%) dev_err 15.4% (+: 20.2%)
dd. k=57 train_err 18.1% (+: 0.2%) dev_err 15.6% (+: 20.0%)
ee. k=59 train_err 18.1% (+: 0.2%) dev_err 15.3% (+: 19.9%)
ff. k=61 train_err 18.0% (+: 0.2%) dev_err 15.7% (+: 20.1%)
gg. k=63 train_err 18.0% (+: 0.2%) dev_err 15.4% (+: 19.8%)
hh. k=65 train_err 17.8% (+: 0.2%) dev_err 15.1% (+: 19.9%)
ii. k=67 train_err 17.9% (+: 0.2%) dev_err 15.5% (+: 19.9%)
jj. k=69 train_err 17.9% (+: 0.2%) dev_err 15.2% (+: 20.0%)
kk. k=71 train_err 17.8% (+: 0.2%) dev_err 15.3% (+: 19.9%)
ll. k=73 train_err 17.8% (+: 0.2%) dev_err 15.1% (+: 20.1%)
mm. k=75 train_err 17.8% (+: 0.2%) dev_err 14.8% (+: 19.8%)
nn. k=77 train_err 17.9% (+: 0.2%) dev_err 14.9% (+: 19.3%)
oo. k=79 train_err 17.9% (+: 0.2%) dev_err 15.1% (+: 19.3%)
pp. k=81 train_err 17.7% (+: 0.2%) dev_err 15.1% (+: 19.5%)
qq. k=83 train_err 17.7% (+: 0.2%) dev_err 15.3% (+: 19.5%)
rr. k=85 train_err 17.8% (+: 0.2%) dev_err 15.3% (+: 19.3%)
ss. k=87 train_err 17.8% (+: 0.2%) dev_err 15.6% (+: 19.6%)
tt. k=89 train_err 17.9% (+: 0.2%) dev_err 15.3% (+: 19.7%)

uu. k=91 train_err 17.9% (+: 0.2%) dev_err 15.2% (+: 19.6%)
 vv. k=93 train_err 17.9% (+: 0.2%) dev_err 15.1% (+: 19.3%)
 ww. k=95 train_err 17.8% (+: 0.2%) dev_err 15.5% (+: 19.3%)
 xx. k=97 train_err 17.9% (+: 0.2%) dev_err 15.5% (+: 19.5%)
 yy. k=99 train_err 17.8% (+: 0.2%) dev_err 15.7% (+: 19.3%)
 zz. Best dev error rate: 14.4% achieved with k=41

34


14400



0.176

9

19s



Your Best Entry!
 Your most recent submission scored 0.176, which is an improvement of your previous score of 0.190. Great job!

Tweet this

3. My new error rate is 17.6% and my ranking is 34. My positive rate is 19.4
- your file has passed the formatting test! Your positive rate is 19.4%. Your positive rate seems reasonable. This does not guarantee that your prediction accuracy will be good though.

Part 4

- The sklearn and self-implementation results reported these three results for the following indices. (sklearn results are in green and self-implementation results are in purple)

Index	Euclidean	Index	Manhattan
4872	0.334 / 0.334	4872	0.390 / 0.390
4787	1.415 / 1.415	4787	2.055 / 2.055
2591	1.417 / 1.417	1084	2.102 / 2.102

- R
 - No, there is no work in training after the feature map because knn is a lazy algorithm and therefore there is no “training” after the creation of the features. It is instance based, so it does not make a model from the training. It uses the observations it makes in the prediction process and assumes that similar things exist close to one another.
 - The time complexity for the KNN algorithm for predicting a single test is $O(nd)$, where n is the number of training examples and d is the number of features.
 - No, you do not really need to sort the distances first and then choose the top k . There are other things that can be used, like `np.argpartition`. `Np.argpartition` allows you to get the indices of the smallest distances without needing to completely sort.

d. I ended up using `np.linalg.norm`, `np.sum`, `np.argmax`, `np.bincount`, and `np.argmax` to speed up my program. `np.linalg.norm` returns one of the seven matrix norms, `np.sum` calculates the sum of the array elements, `np.bincount` counts the number of unique values in an array, and `np.argmax` finds the indices of the max values along a determined axis. (all of these definitions were obtained on google using digitalOcean, BMC Software, Analytis Vidhya, and geeksforgeeks). I used these in one version of my code. In another version I only used `np.sum` and `np.abs`.

- i. k=1 train_err 1.5% (+: 0.3%) dev_err 23.9% (+: 27.3%)
- ii. k=3 train_err 11.5% (+: 0.2%) dev_err 19.5% (+: 25.9%)
- iii. k=5 train_err 13.7% (+: 0.2%) dev_err 18.4% (+: 24.8%)
- iv. k=7 train_err 14.2% (+: 0.2%) dev_err 16.6% (+: 23.6%)
- v. k=9 train_err 15.5% (+: 0.2%) dev_err 15.8% (+: 22.2%)
- vi. k=11 train_err 16.3% (+: 0.2%) dev_err 16.3% (+: 21.1%)
- vii. k=13 train_err 16.5% (+: 0.2%) dev_err 16.5% (+: 21.9%)
- viii. k=15 train_err 16.5% (+: 0.2%) dev_err 15.8% (+: 21.6%)
- ix. k=17 train_err 16.7% (+: 0.2%) dev_err 16.0% (+: 21.6%)
- x. k=19 train_err 17.0% (+: 0.2%) dev_err 16.6% (+: 21.0%)
- xi. k=21 train_err 17.1% (+: 0.2%) dev_err 16.3% (+: 20.9%)
- xii. k=23 train_err 17.2% (+: 0.2%) dev_err 15.5% (+: 21.7%)
- xiii. k=25 train_err 17.0% (+: 0.2%) dev_err 15.4% (+: 21.4%)
- xiv. k=27 train_err 16.9% (+: 0.2%) dev_err 15.7% (+: 20.9%)
- xv. k=29 train_err 17.1% (+: 0.2%) dev_err 15.0% (+: 20.8%)
- xvi. k=31 train_err 17.0% (+: 0.2%) dev_err 15.1% (+: 21.3%)
- xvii. k=33 train_err 17.0% (+: 0.2%) dev_err 15.6% (+: 21.4%)
- xviii. k=35 train_err 17.2% (+: 0.2%) dev_err 15.0% (+: 20.8%)
- xix. k=37 train_err 17.2% (+: 0.2%) dev_err 14.8% (+: 20.8%)
- xx. k=39 train_err 17.2% (+: 0.2%) dev_err 14.7% (+: 20.9%)
- xxi. k=41 train_err 17.4% (+: 0.2%) dev_err 14.3% (+: 20.9%)
- xxii. k=43 train_err 17.5% (+: 0.2%) dev_err 14.8% (+: 20.4%)
- xxiii. k=45 train_err 17.7% (+: 0.2%) dev_err 15.0% (+: 20.4%)
- xxiv. k=47 train_err 17.8% (+: 0.2%) dev_err 15.1% (+: 20.1%)
- xxv. k=49 train_err 18.0% (+: 0.2%) dev_err 15.3% (+: 20.1%)
- xxvi. k=51 train_err 17.9% (+: 0.2%) dev_err 15.7% (+: 19.9%)
- xxvii. k=53 train_err 18.0% (+: 0.2%) dev_err 15.3% (+: 19.7%)
- xxviii. k=55 train_err 17.9% (+: 0.2%) dev_err 15.4% (+: 20.0%)
- xxix. k=57 train_err 18.0% (+: 0.2%) dev_err 15.6% (+: 19.8%)
- xxx. k=59 train_err 18.1% (+: 0.2%) dev_err 15.6% (+: 19.6%)

- xxxi. k=61 train_err 18.0% (+: 0.2%) dev_err 15.5% (+: 19.5%)
- xxxii. k=63 train_err 18.1% (+: 0.2%) dev_err 15.4% (+: 19.8%)
- xxxiii. k=65 train_err 18.0% (+: 0.2%) dev_err 15.5% (+: 19.9%)
- xxxiv. k=67 train_err 17.9% (+: 0.2%) dev_err 15.5% (+: 19.9%)
- xxxv. k=69 train_err 17.9% (+: 0.2%) dev_err 15.2% (+: 19.8%)
- xxxvi. k=71 train_err 18.0% (+: 0.2%) dev_err 15.6% (+: 19.6%)
- xxxvii. k=73 train_err 17.9% (+: 0.2%) dev_err 15.2% (+: 20.0%)
- xxxviii. k=75 train_err 17.9% (+: 0.2%) dev_err 14.9% (+: 19.7%)
- xxxix. k=77 train_err 17.9% (+: 0.2%) dev_err 15.0% (+: 19.4%)
- xl. k=79 train_err 17.8% (+: 0.2%) dev_err 15.1% (+: 19.5%)
- xli. k=81 train_err 17.8% (+: 0.2%) dev_err 15.3% (+: 19.3%)
- xl. k=83 train_err 17.7% (+: 0.2%) dev_err 15.5% (+: 19.3%)
- xl. k=85 train_err 17.8% (+: 0.2%) dev_err 15.5% (+: 19.1%)
- xliv. k=87 train_err 17.9% (+: 0.2%) dev_err 15.4% (+: 19.4%)
- xl. k=89 train_err 17.8% (+: 0.2%) dev_err 15.5% (+: 19.7%)
- xlvi. k=91 train_err 17.9% (+: 0.2%) dev_err 15.5% (+: 19.5%)
- xl. k=93 train_err 17.9% (+: 0.2%) dev_err 15.2% (+: 19.2%)
- xl. k=95 train_err 17.8% (+: 0.2%) dev_err 15.3% (+: 19.7%)
- xl. k=97 train_err 17.9% (+: 0.2%) dev_err 15.5% (+: 19.5%)
- l. k=99 train_err 17.8% (+: 0.2%) dev_err 15.8% (+: 19.2%)
- li. Best dev error rate: 14.3% achieved with k=41
- lii. Positive rate in test set: 20.90%

e. I got a time of about 20 seconds.

3. My Manhattan numbers are higher than the scores I got for the Euclidean equation but not by much. I assume the reason the Euclidean scores are lower is because it does not have as rigid a path as the Manhattan equation. The video explained that the Manhattan equation is like driving to a location (and you must follow the path of the road) and the Euclidean equation is like you are flying (you can just cut across). It would make sense for the Euclidean results to be lower than the Manhattan results.

Part 5

1. I obtained comparable results with both distances but my results for the Euclidean distance were better. The best results were achieved at k = 41.
2. The best dev-error for k = 41 was 14.3% and the positive rate was 20.90%.
3. My positive ratio is 20.9%

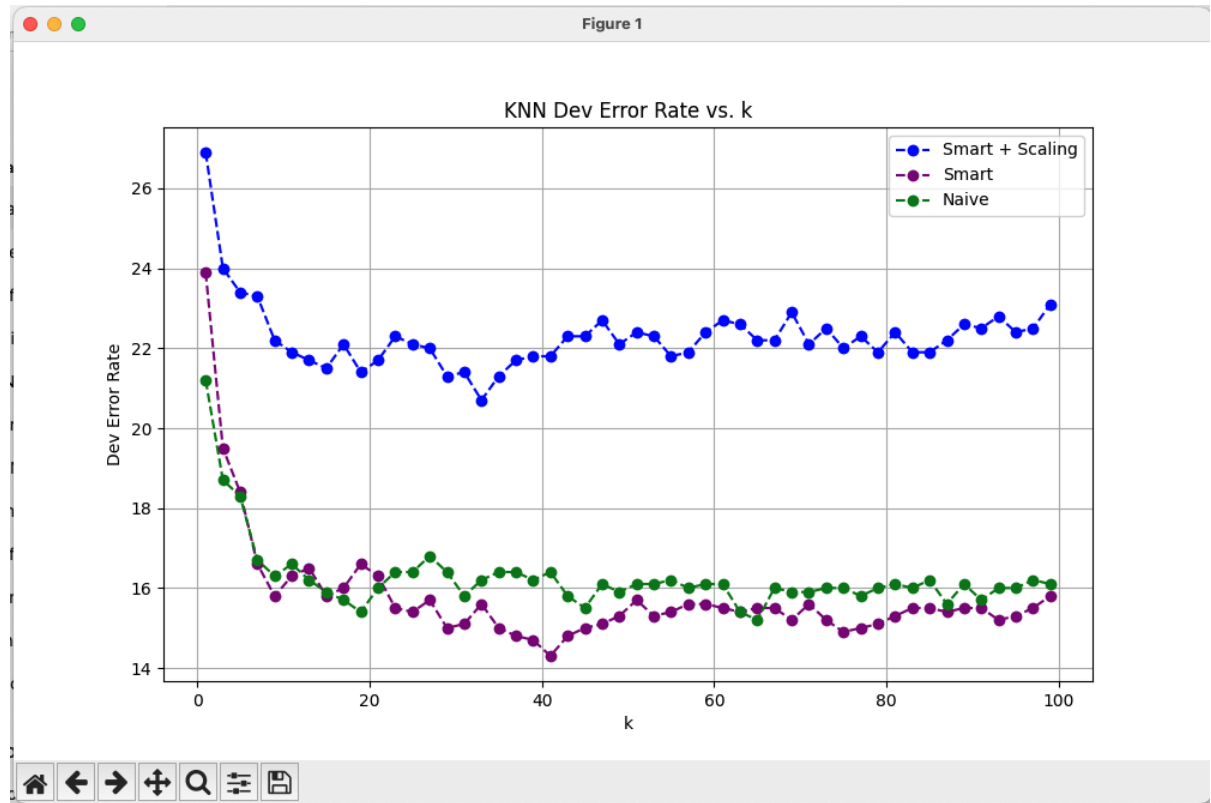
4. My best rank is 42 with a score of 0.168. It took me 22 tries.



Part 6

1. One of the biggest drawbacks is that every feature is treated equally when calculating distances leading to features of less relevance having the same importance as a feature of more relevance. Another drawback is that because every feature is treated the same, it can skew the results. The size of the dataset can also cause issues because knn calculates the distance between each data point in the training set. So, if the training set is exceptionally large, it can slow down the predictions. The k that is chosen can also impact the model (a smaller k can lead to overfitting and a larger k can lead to underfitting).
2. Yes, the best-performing models can exaggerate the existing bias because k-NN works by selecting data points that are closest to the test sample, based on the patterns in the training data. This means that any biases present in the training set can directly influence the model's predictions. The exaggeration of bias can result from both overfitting and underfitting, but overfitting is more common and more probable. Overfitting (low k) can cause the exaggeration of bias because the small k causes the model to become sensitive to individual data points. This can cause the data to create patterns that will exaggerate the bias present in the training data (as can be seen by the fact that my model shows a bias towards men). Underfitting (high k) can also cause bias because the model can reflect and amplify an already existing bias in the training data. The existing bias is caused by overfitting and it is a social issue. In reviewing the data, I noticed that there are higher percentages of women with low income compared to men. This can suggest that the training data is biased and reflects the larger problem of societal disparities between the economic standings of men and women. Additionally, I noticed that there were more men than women in the dataset overall. The imbalance can create a bias (overfitting) where the model disproportionately predicts based from male patterns causing men to be over-represented and women to be under-represented. So, yes, there is bias and it is most likely caused by overfitting.

Part 7 (extra credit)



1.

Debriefing

1. I think I spent about 40 to 50 hours on the assignment over the three weeks.
2. I would rate the assignment of moderate difficulty because while most of the assignments were straightforward, there were some questions that caused me quite a bit of trouble.
3. My assignment was a healthy balance of working alone and working with others (I often attended office hours as well).
4. I think that I understand about 50-70% of the material deeply, but I am working on that understanding and hope to further my understanding more as the term progresses.
5. I do not have any other comments.