HW1

1. DATA PRE-PROCESSING (FEATURE MAP):

1) Training set - Positive % of data – 1251/5000 = 25.02%
   Dev set - Positive % of data – 236/1000 = 23.59%
2) Training set – Youngest age – 17 Oldest age - 90
   Training set – Minimum number of hours – 1 Maximum number of hours = 99
3) Fields or their values have no meaning for the model unless they're all values of a given field are encoded using a standard scheme for representation purpose. Hence, all categorical fields are encoded to represent the spectrum of values assumed in rows of the dataset.
   In other words, age, sector, education etc. are fields having varied values which needs to be encoded using a standard scheme for model to understand. That is, age=30 or age=56 should be represented as (0,'30') and (0,'56') respectively. Similarly, sector=Private and sector='State-gov' is represented as (1,'Private') and (1,'State-gov') respectively.
   In this way, the model realized that features encountered with first index as 0 represent age, 1 represents sector, 2 represents education, 3 represents marital-status, 4 represents occupation, 5 represents race, 6 represents gender, 7 represents number-of-hours-per-week and 8 represents country.
4) **If age and hours are not considered, then maximum possible Euclidean and Manhattan distances are obtained as follows:**
   i. Maximum Euclidean distance – $\sqrt{14}$
   ii. Maximum Manhattan distance – 14
5) **Why do we not want to binarize numerical fields age and hours?** - All the fields are disintegrated into features (for both numerical and categorical) to let kNN algorithm to calculate similarities based on those features between different data points. This is done specially for categorical fields to infer similarity between otherwise what were just values. Whereas for numerical fields, similarity is already inferred based on their values and hence they're already features in their own sense.
   Another reason for not binarizing numerical fields is that it might result in very high number of features which might not necessarily be useful for our model which is being 'trained' with just 5000 data-points.
6) **Total number of features** – 92 (including age and number of hours per week)

Field	Field Index	Number of features
Age	0	67
Sector	1	7
Education	2	16
Marital-Status	3	7

Field	Field Index	Number of features
Occupation	4	14
Race	5	5
Gender	6	2
Hours-per-week	7	73
Country	8	39

7) Number of features if we binarize all the fields – 230

2. K-NN CLASSIFICATION

1. Implement k-NN classifier

1) **Is there any work in training after finishing the feature map?** No
2) **What's the time complexity of k-NN to test one example (dimensionality d, size of training set |D|)?** – For each new test example, we calculate distance with respect to all the features (dimensionality d) of a single training example. This is in turn calculated for all the training examples [D]. Essentially, this leads to d*|D| time complexity. Additionally, to select top k elements, we can sort the distances of given test example from all the training examples which adds the cost of |D|*log(|D|). Finally, selecting top k from this sorted list of distances adds cost of k. Thus total time complexity is O(d*|D| + |D|*log(|D|) + k)
3) **Do you really need to sort the distances first and then choose the top k?** Hint: there is a faster way to choose top k without sorting – No. We can use partial sorting using np.argpartition to get the indices of distance of top k elements selected from the unordered list of distances. One can also use quick-select algorithm to select the kth smallest elements of an unordered list to get the k nearest neighbors.

2. **Why does k in k-NN has to be an odd number?** – To get majority votes in situation of equal number of weights amongst the neighbors, picking odd number of neighbors would be helpful.
3. Error rate and predicted positive rates for different values of k on dev dataset are as mentioned below:

K	Error Rate	Predicted Positive Rate
1	23.9	26.7
3	20.0	24.6
5	23.8	23.8
7	17.4	24.0

K	Error Rate	Predicted Positive Rate
9	17.0	21.6
31	16.7	18.9
32	16.1	18.9
33	16.4	18.4
34	15.8	18.6
35	15.9	19.1
99	16.7	16.7
999	20.4	5.8
9999	20.4	5.8

Best error-rate on dev set is 15.8%
Best error-rate on dev set is obtained for k = 34

4. Training and Testing error rates:

K	Training Error Rate	Predicted Positive Rate
1	10.1	125.4
3	58.8	117.7
5	70.2	118.3
7	73.9	116.8
9	78.3	114.6
31	85.4	100.3
32	86.4	99.5
33	85.6	98.3

K	Training Error Rate	Predicted Positive Rate
34	84.8	97.3
35	85.2	97.9
99	89.1	91.6
999	110.9	26.6
9999	125.1	0.0

Training error is not 0 when k=1.

5.

1) What trends (train and dev error rates and positive ratios, and running speed) do you observe with increasing k? Do they relate to underfitting and overfitting? – With increasing values of k, train and dev error rates are observed to increase which indicate overfitting. This is also evident from the trend of strictly decreasing positive rates with increasing value of k for both dev dataset and training dataset.
2) What does k = ∞ do? Is it extreme overfitting or underfitting? What about k = 1? – When k approaches infinity, the we see that error rate overshoots indicating extremely high overfitting leading to predictions which be as good as random predictions. For k=1, we see that error-rate is high indicating underfitting because the model has basically failed to learn from its neighbors.

6. Evaluation using Manhattan distance:

1) Error-rates and positive percentage on dev-set using Manhattan distance are close to those observed using Euclidean distance.

K	Error Rate	Predicted Positive Rate
1	23.9	26.7
3	20.7	25.1
5	19.0	24.0
7	17.3	23.1
9	17.4	21.8
31	16.5	19.1
32	16.6	19.0

K	Error Rate	Predicted Positive Rate
33	16.4	19.0
34	16.2	19.0
35	16.1	19.1
99	16.8	16.8
999	20.9	5.1
9999	20.9	5.1

2) Similarly, error-rates and positive percentage on training data-set using Manhattan distance are also close to those observed using Euclidean distance.

K	Training Error Rate	Predicted Positive Rate
1	10.1	125.4
3	58.9	117.6
5	70.7	116.0
7	74.3	117.0
9	77.8	114.3
31	85.3	98.8
32	85.6	97.1
33	86.0	97.1
34	85.6	97.5
35	86.2	97.5
99	89.3	91.8

K	Training Error Rate	Predicted Positive Rate
999	111.6	25.1
9999	125.1	0.0

7. Evaluation using all binarized features (using Euclidean distance):
    1) Error-rates and positive percentage on dev-set using Euclidean distance are close to those observed by not including age and number of hours worked per week. However, it can be seen that error-rate by using all the features including age and number of hours worked per week oscillates instead of strictly increasing monotonically after dropping to minimum at k=15.3.

    Predicted positive rate is also seen to be decreasing monotonically as seen with Euclidean distances by using all the binarized features.

K	Error Rate	Predicted Positive Rate
1	23.6	26.8
3	19.3	25.7
5	17.6	24.8
7	16.2	24.0
9	15.8	22.4
31	15.5	20.9
32	16.0	21.0
33	15.4	20.4
34	15.7	20.5
35	15.3	20.5
99	15.7	19.3
999	17.9	11.1
9999	18.0	11.0

2) Training error using all binarized features including age and number of hours per week:

K	Training Error Rate	Predicted Positive Rate
1	7.6	125.3
3	58.0	120.3
5	69.5	21.6
7	72.6	120.9
9	77.1	120.6
31	85.0	107.7
32	84.5	106.4
33	84.5	105.0
34	85.1	104.8
35	84.9	104.4
99	89.2	97.1
999	101.1	52.2
9999	125.1	0.0

3. DEPLOYMENT:

1. **At which k and with which distance did you achieve the best dev results?** – For k=34, best dev results were achieved.
2. **What's your best dev error rates and the corresponding positive ratios?** Best error rate is 15.3 by using all binarized features whereas best error rate is 15.8 by encoding all categorical features with k = 34. Positive ratio is k/corresponding positive rate = 34/18.6 = 1.83
3. **What's the positive ratio on test?** – k/corresponding positive rate = 34/

4. OBSERVATIONS:

1. Summarize the drawbacks of k-NN

a. High computation cost
b. Slow in training

c. Needs lot of memory
d. Prediction is slow

2. **Do you observe in this HW that best-performing models tend to exaggerate the existing bias in the training data? Is it due to overfitting or underfitting? Is this a potentially social issue?**
3. **What numpy tricks did you use to speed up your program so that it can be fast enough to print the training error? Hint: (a) broadcasting (such as matrix - vector); (b) np.linalg.norm(..., axis=1); (c) np.argsort() or np.argpartition(); (d) slicing.** – Broadcasting was used to calculate the distance of test example with respect to all the training examples in np.linalg.norm. The same method was used to efficiently calculate the Euclidean and Manhattan distances. Slicing was used throughout the code to extract portions of dataset for further usage. Method np.argpartition was also used to get indices of k smallest distances of a given test example with respect to all points in the training dataset.
4. **How many seconds does it take to print the training and dev errors for k = 99 on ENGR servers? Hint: use time python ... and report the user time instead of the real time.** – 40s on local machine
5. **What is a Voronoi diagram (shown in k-NN slides)? How does it relate to k-NN?** - A Voronoi diagram is a partitioning of a plane into regions based on distance to points in a specific subset of the plane. That set of points (called seeds, sites, or generators) is specified beforehand, and for each seed there is a corresponding region consisting of all points closer to that seed than to any other. These regions are called Voronoi cells.

5. DEBRIEFING:

1. Approximately how many hours did you spend on this assignment? – 36 hours
2. Would you rate it as easy, moderate, or difficult? - Moderate
3. Did you work on it mostly alone, or mostly with other people? – Mostly alone
4. How deeply do you feel you understand the material it covers (0%–100%)? – 70%
5. Any other comments?