AI 534 Machine Learning

Name: Nirmit Patel
Email: patenirm@oregonstate.edu

1 Hands-on Exploration of the Income Dataset

Q: What percentage of the training data has a positive label (>50K)? (This is known as the positive %). What about the dev set? Does it make sense given your knowledge of the average US per capita income?

Training Data positive label (>50K): 25.02%
Dev Set positive label (>50K): 23.60%

Q: What are the youngest and oldest ages in the training set? What are the least and most amounts of hours per week do people in this set work?

Youngest: 17; least hours = 5; most hours=48
Oldest: 90; least hours = 35; most hours = 60
Least Hours = 1
Most Hours = 99

Q: Why do we need to binarize all categorical fields?

Binarizing converts the categorical fields to numerical form so that machine learning models can effectively work with them and extract meaningful information (e.g. calculating distance).

Q: Why we do not want to binarize the two numerical fields, age and hours?

Not binarizing numerical fields like age and hours is preferred because:

- It provides more valuable information by using differences as distances.
- It keeps the dimensionality of the feature map lower (92 dimensions) compared to binarization (230 dimensions).
- It allows for accurate representation of distances between ages and hours.

Q: How should we deal with the two numerical fields? Just as is, or normalize them (say, age / 100)?

Normalization is recommended for numerical fields like "age" and "hours" It scales them to a common range, making them directly comparable to categorical fields, which often have a different scale due to their finite set of values. This ensures that all data features have a similar scaling for more effective machine learning analysis.

Q: How many features do you have in total (i.e., the dimensionality)?

We have 92 features in total.

2 Data Preprocessing and Feature Extraction I: Naive Binarization

Question: Although pandas.get_dummies() is very handy for one-hot encoding, it's absolutely impossible to be used in machine learning. Why?

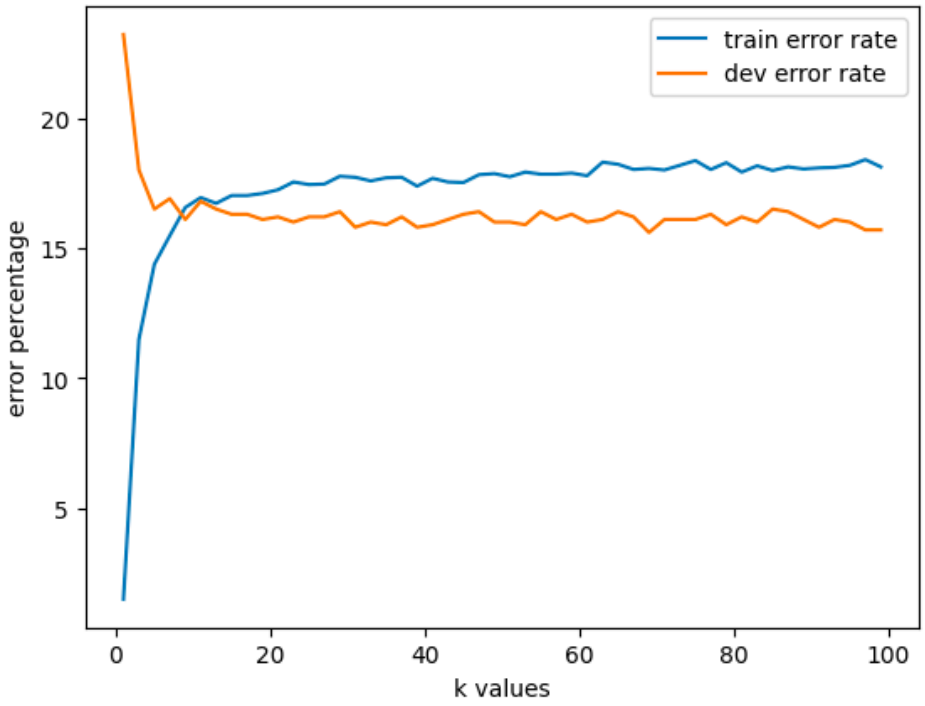
> It's not suitable for machine learning because it doesn't handle unseen data. Machine learning models must be able to accommodate new data but get_dummies encodes features based on the values present in the training data. This can lead to inconsistencies if the number of unique values in the testing data differs from the training data, potentially causing issues in model performance.

Question: After implementing the naive binarization to the real training set (NOT the toy set), what is the feature dimension? Does it match with the result from Part 1 Q5?

> The feature dimension, after applying naive binarization to the actual training set is 230. This does not align with the result from Part 1, Q5.

(a) Evaluate k-NN on both the training and dev sets and report the error rates and predicted positive rates for k from 1 to 100 (odd numbers only, for tie-breaking)
Q: what's your best error rate on dev? Which k achieves this best error rate?

> The best error rate on dev set is 15.6% for k = 69



(b) Q: When k = 1, is training error 0%? Why or why not? Look at the training data to confirm your answer.

> No, when k = 1, the training error is not 0%. This is because, with k = 1, the model is overly flexible and essentially memorizes the training data (overfitting). Consequently, the training error

appears low, but the model is unlikely to generalize well to new, unseen data, leading to high test error.

(c) Q: What trends (train and dev error rates and positive ratios, and running speed) do you observe with increasing k?

- The training and development error rates initially decrease as k increases, indicating that the model is improving in terms of fitting the training data and generalizing to the development data. However, after a certain point, both the training and development errors start to increase, suggesting overfitting.

- The positive ratios, related to the model's accuracy in classifying positive cases, show a similar trend. They initially improve with increasing k but start to degrade after a certain threshold.

- The running speed does not exhibit a consistent trend with increasing k. Instead, we see a decrease in running speed. This can be due to the number of comparisons that the model has to do to predict the value of y.

(d) Q: What does k = ∞ actually do? Is it extreme overfitting or underfitting? What about k = 1?

Setting k to an extremely large value (k = ∞) results in underfitting, where the model relies on the entire dataset for predictions, exhibiting high bias and low variance. It makes decisions based on the dataset's overall characteristics.

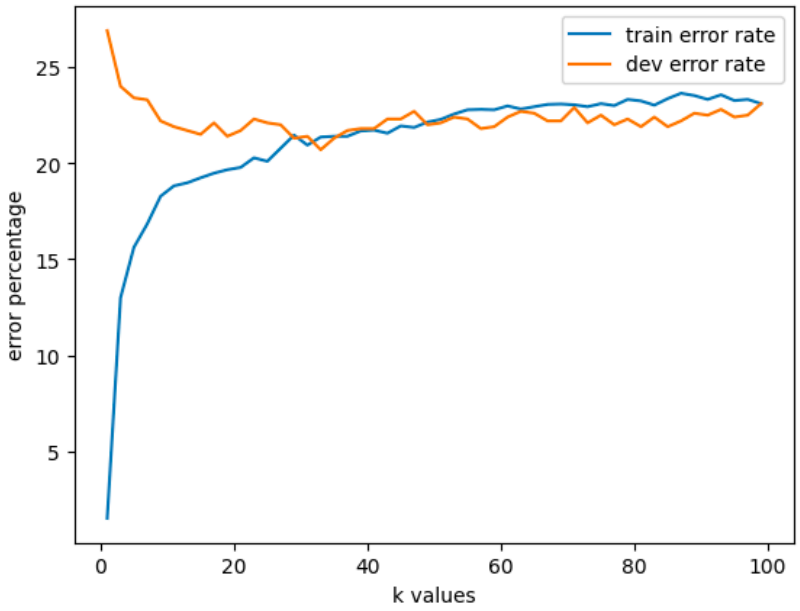
Conversely, when k is set to 1, it represents extreme overfitting. In this case, only the nearest single neighbor is considered for predictions, resulting in low bias but high variance. This sensitivity to individual data points can lead to overfitting, where the model may not generalize well to the underlying data patterns, particularly when there is noise or outliers in the data.

3 Data Preprocessing and Feature Extraction II: Smart Binarization

```
num_processor = 'passthrough' # i.e., no transformation
cat_processor = OneHotEncoder(sparse=False, handle_unknown='ignore')
```

Question: Re-execute all experiments with varying values of k (Part 2, Q 4a) and report the new results. Do you notice any performance improvements compared to the initial results? If so, why? If not, why do you think that is?

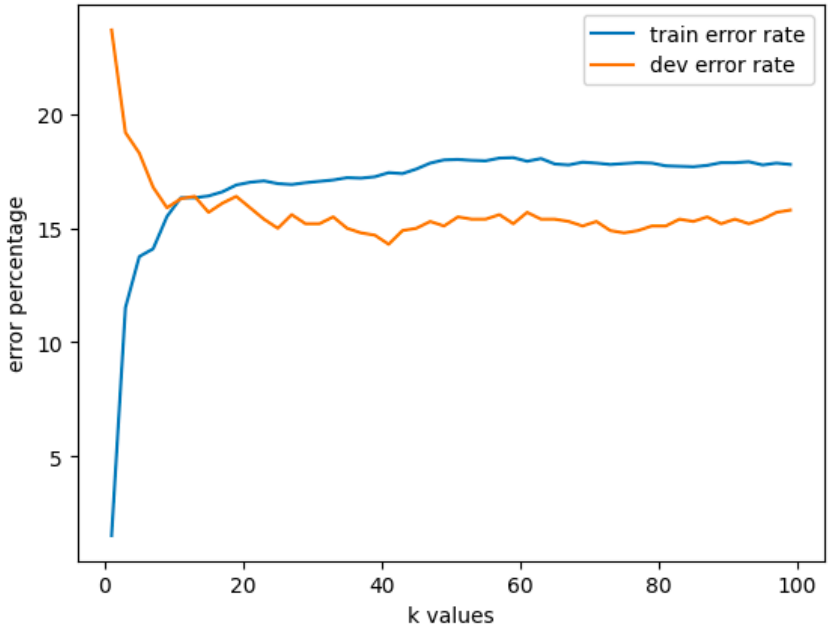
No, the new results indicate that naive encoding outperforms smart encoding. In naive encoding, not only were the categorical fields binarized, but the continuous variables "age" and "hours" were also transformed. This comprehensive binarization of both categorical and continuous features in naive encoding likely reduced errors, resulting in a better error rate compared to the smart encoding, where the numerical fields were not transformed ("passthrough"). Therefore, naive encoding appears to be a more effective approach in this context.



num_processor = MinMaxScaler(feature_range=(0, 2))
cat_processor = OneHotEncoder(sparse=False, handle_unknown='ignore')

Questions: Again, rerun all experiments with varying values of k and report the results (Part 2, Question 4a). Do you notice any performance improvements? If so, why? If not, why do you think that is?

Yes, in the Smart-Scaled encoding, there is a notable improvement in performance compared to the Naive encoding. The key difference is that in the Smart-Scaled encoding, numerical fields are not binarized but rather scaled using the MinMaxScaler. This scaling likely contributes to the improved performance, making the model better at handling numerical features, resulting in lower error rates.



4 Implement your own k-Nearest Neighbor Classifiers

Question: Before implementing your k-NN classifier, try to verify the distances from your implementation with those from the sklearn implementation. What are the (Euclidean and Manhattan) distances between the query person above (the first in dev set) and the top-3 people listed above? Report results from both sklearn and your own implementation.

Sklearn:

Euclidean: 4872: distance = 0.33441929
4787: distance = 1.41527469
2591: distance = 1.41674697

Manhattan: 4872: distance = 0. 38999161
4787: distance = 2. 05479452
1084: distance = 2. 10204082

My implementation:

Euclidean: 4872: distance = 0.3344192869821078
4787: distance = 1.4152746869361013
2591: distance = 1.4167469717499104

Manhattan: 4872: distance = 0.38999161308358965
4787: distance = 2.054794520547945
1084: distance = 2.1020408163265305

(a) Q: Is there any work in training after the feature map (i.e., after all fields become features)?

There is no additional work in the training phase after completing the feature map.

(b) Q: What's the time complexity of k-NN to test one example (dimensionality d, size of training set |D|)?

The time complexity of k-NN for testing one example with a dimensionality of d and a training set size of |D| is O(|D| * d).

(c) Q: 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.

Sorting the distances and choosing the top k has a time complexity of O(|D| * log |D|) with sorting. However, a faster way to select the top k without sorting is the "quick-select" algorithm, which has a time complexity of O(|D|).

(d) Q: What numpy tricks did you use to speed up your program so that it can be fast enough to print the training error? Hint: (i) broadcasting (such as matrix - vector); (ii) np.linalg.norm(..., axis=1); (iii) np.argsort() or np.argpartition(); (iv) slicing. The main idea is to do as much computation in the vector-matrix format as possible (i.e., the Matlab philosophy), and as little in Python as possible.

To accelerate the program and efficiently calculate k-NN, several numpy techniques were used, including broadcasting for operations on arrays without loops, np.linalg.norm(..., axis=1) for fast distance computation, and np.argpartition(...) for a partial sort to obtain the indices of the k-nearest neighbors. These techniques optimize performance by minimizing Python-level computations and following a vector-matrix approach.

(e) Q: 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.

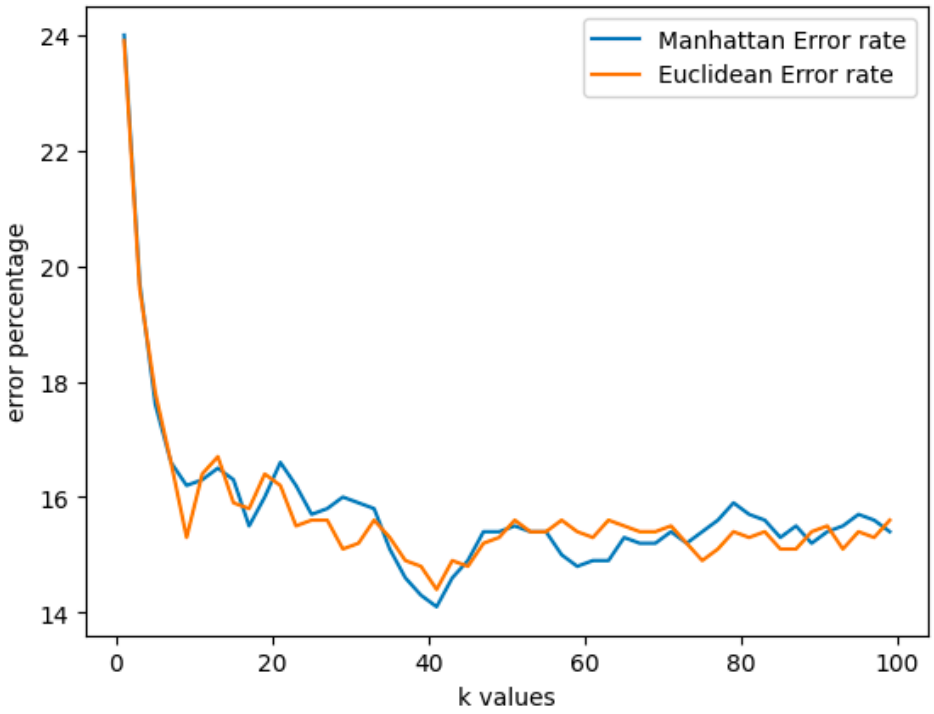
For me, on the ENGR servers, it takes approximately 13 seconds of user time to print the training and dev errors for k = 99.

```
[patenirm@flip1 ~/Fall2023/AI534$] time python knn.py
k = 99 train_err: 17.82% dev_err: 15.60%

real    0m22.472s
user    0m13.345s
sys     0m8.344s
```

Redo the evaluation using Manhattan distance (for k = 1 ~ 99). Better or worse?

Both Manhattan and Euclidean distance metrics show similar error rates on the dev set.



5 Deployment

Q: At which k and with which distance did you achieve the best dev results?

The best dev results were achieved at k = 41 for both Manhattan and Euclidean distance metrics. However, there is a slight difference in the accuracy: the best accuracy for Manhattan distance was 85.9%, while the best accuracy for Euclidean distance was 85.6%.

Q: What's your best dev error rates and the corresponding positive ratios?

My best dev error rate is 14.1%, which corresponds to the choice of k=41. The corresponding positive ratio for this configuration is 20.5%.

Q: What's the positive ratio on test?

The positive ratio on the blind test data, using k=41 and Manhattan distance, is 20.8%.

6 Observations

Q: Summarize the major drawbacks of k-NN that you observed by doing this HW.

The major drawbacks of k-NN observed in this homework are:

- Lack of Learning: k-NN doesn't extract meaningful information from the data or learn patterns as other classifiers do. It relies solely on raw data.
- Slow Testing: Testing in k-NN is slow, and its complexity grows linearly with the size of the training data, making it impractical for big data applications.
- Equal Importance of Dimensions: k-NN treats all dimensions equally, even when some may be more important than others. For example, age should carry more weight than some other features.
- Computationally Intensive: The algorithm is computationally intensive, particularly with large datasets, making it less efficient.
- Fixed k-Value: The choice of the k-value is static and doesn't adapt to changes in the dataset, potentially affecting performance.

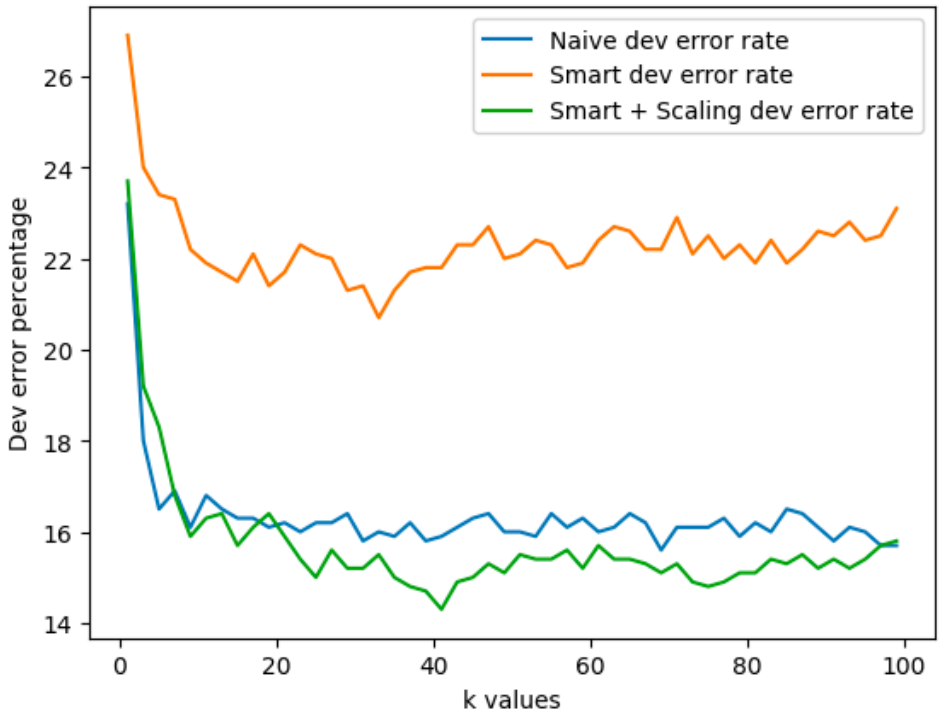
Q: Do you observe in this HW that best-performing models tend to exaggerate the existing bias in the training data? Is it due to overfitting or underfitting? Is this a potentially social issue?

In this HW, it's evident that the best-performing models tend to amplify the existing biases present in the training data. This phenomenon is primarily due to underfitting.

The issue here is that the computer programs we're creating often end up making biases in our society even worse. This is because these programs don't always understand the data they're given

very well. So, if they see something as a positive sign, they might make it even more positive than it really is, which can create problems. This is a big concern as we rely more and more on these programs in our daily lives.

7 Extra Credit Question



Debriefing

1. Approximately how many hours did you spend on this assignment?
   Approximately 40 hrs.

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%)?
   90%

5. Any other comments?
   It was a great time working on this assignment. I don't have proper experience in Machine Learning but the way this course is designed, I am able to get a clear understanding.