Project 02 Summary

Tamique de Brito

Jul 2019

Abstract

This is a description of my second project in machine learning and data science. My goal in this project was to learn more about the data manipulation/visualization side, as opposed to my first project, whose purpose was to allow me to practice utilizing the algorithms I implemented (though I still did use the KNN algorithm in this project). The project was an exploration of the data set "Travel Review Ratings" obtained from the UCI Machine Learning Repository. I used the Pandas library for some of the data manipulation and Seaborn/matplotlib for visualization. As with the paper describing my first project, the purpose of writing this is to practice paper-writing, briefly reflect on what I did, and communicate my level of experience in this type of work.

1 The Dataset

The "Travel Review Ratings" dataset is a collection of Google reviews by various users of 24 attractions across Europe. The ratings are on a 1-5 scale, and are the average ratings for the user over that category (so that the values are decimals, not integers).

1.1 Data Loading and Cleaning

There wasn't much to do with the data. Only one data point had a missing value; that point was removed. Other than that, it was minimal manipulation to get it all into (feature vector, label) format. The Pandas library was used for some of the data manipulation (mostly when loading and doing a bit of formatting). Most of the time, the data was simply formatted as a list of (numpy vector, label) pairs.

1.2 Data Summary

Various information about the data was displayed, e.g. the histograms for individual features, the correlation heatmap, the highest correlations, histogram of correlations (highest correlation between any two features was about 0.62), and the number of users rating each category (rating of "0" means no reviews). Images generated are shown in figures 1, 2, 3, and 4.

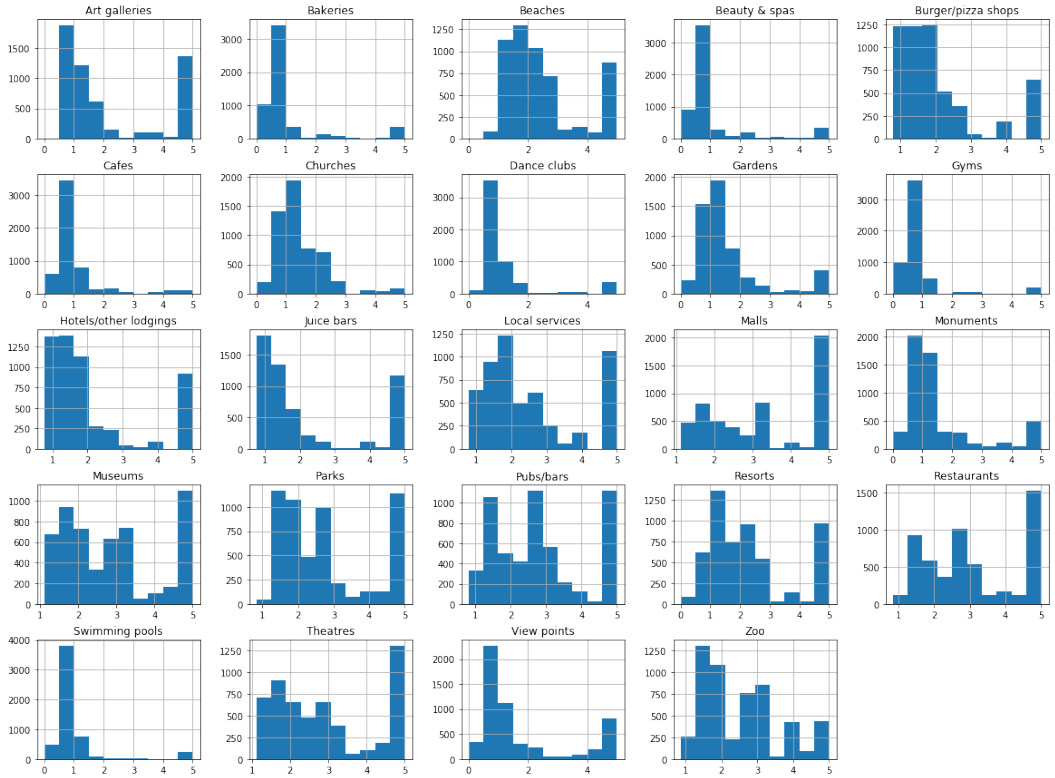

Figure 1: Feature Histograms

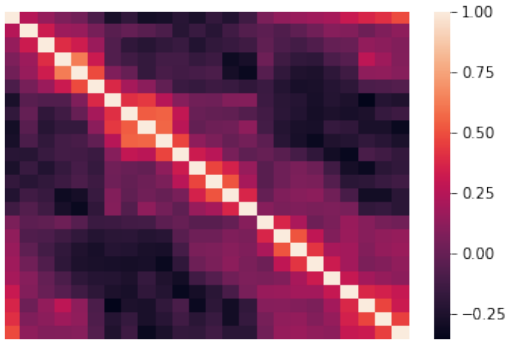

Figure 2: Correlation Heatmap

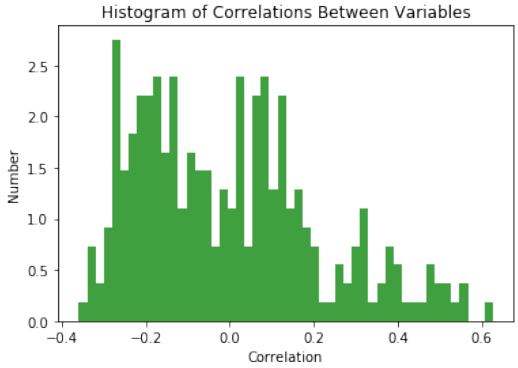

Figure 3: Correlations Histogram

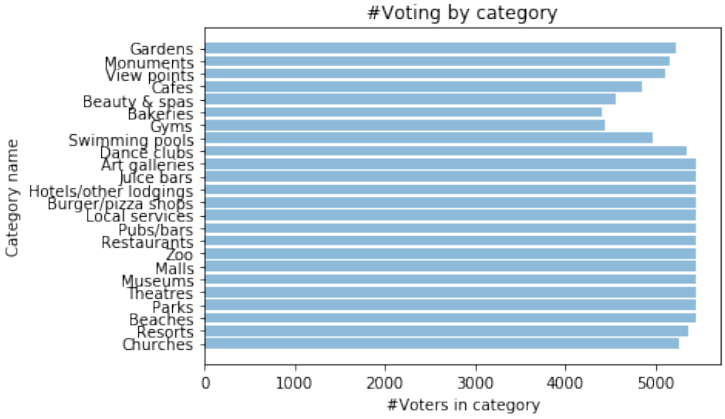

Figure 4: Number of Users Submitting Ratings in Each Category

2 Predicting Ratings of Reviewers on Given Category Based on Ratings in Other Categories with KNN

Here, the KNN algorithm was used to predict the rating a user will give a specified destination based on their rating of a select set of other destinations. The loss function used was average square difference, i.e. $\frac{1}{N}\sum(\hat{y_i} - y_i)^2$, where $\hat{y_i}$ are predicted ratings and y_i are the actual ratings. This means that the absolute average difference between predicted ratings and true ratings ($\frac{1}{N}\sum|\hat{y_i} - y_i|$) will be approximately the square root of the loss computed with the average-square loss, so that the absolute differences will be a bit higher than the given losses (since they are all less than 1).

2.1 Selecting Features to Use for Prediction

The features to use for prediction were selected by their correlation to the specified feature. The features with *lower* correlation were chosen to predict the value of the specified feature. More specifically, the features with correlation under a certain specified cutoff were selected. Of course, as would be expected, the attributes with higher correlation were better for prediction, but I wanted to see what would happen if the lower correlation ones were used.

2.2 Prediction Results

Figure 5 shows, for each feature, the loss of KNN over a range of K values and correlation cutoff values. These values were obtained, for each K and cutoff pair, by taking a specified number of samples of specified size from the data, computing the loss on those samples (after splitting each into a training and test set—8:2 ratio) and averaging them. It can be seen that the loss tends to decrease as more high-correlation attributes are used for prediction, although the benefit varies over different attributes (and of course similar information could be used to pick a minimum set of features that are "good enough" for prediction, in order to improve efficiency).

The loss also varies over K, decreasing initially, then rising back up. This would likely be because of two main reasons. First, at a low K, there is higher variance in the K nearest neighbors, so a given point is likely to get a neighbor average that is farther from its true value. Alternatively, selecting fewer points gives a higher weight to outliers, so that each errant point may cause multiple others to be misclassified. Second, when K gets too large, points from other clusters may start to be incorporated, pulling the predicted value towards their labels.

The lowest loss was about 0.5, for the destination "Churches" and the highest loss was about 3.0 for the destination "Art Galleries".

A possible modification would be to use all features for prediction, but weight their importance by the absolute value of the correlation of the features.

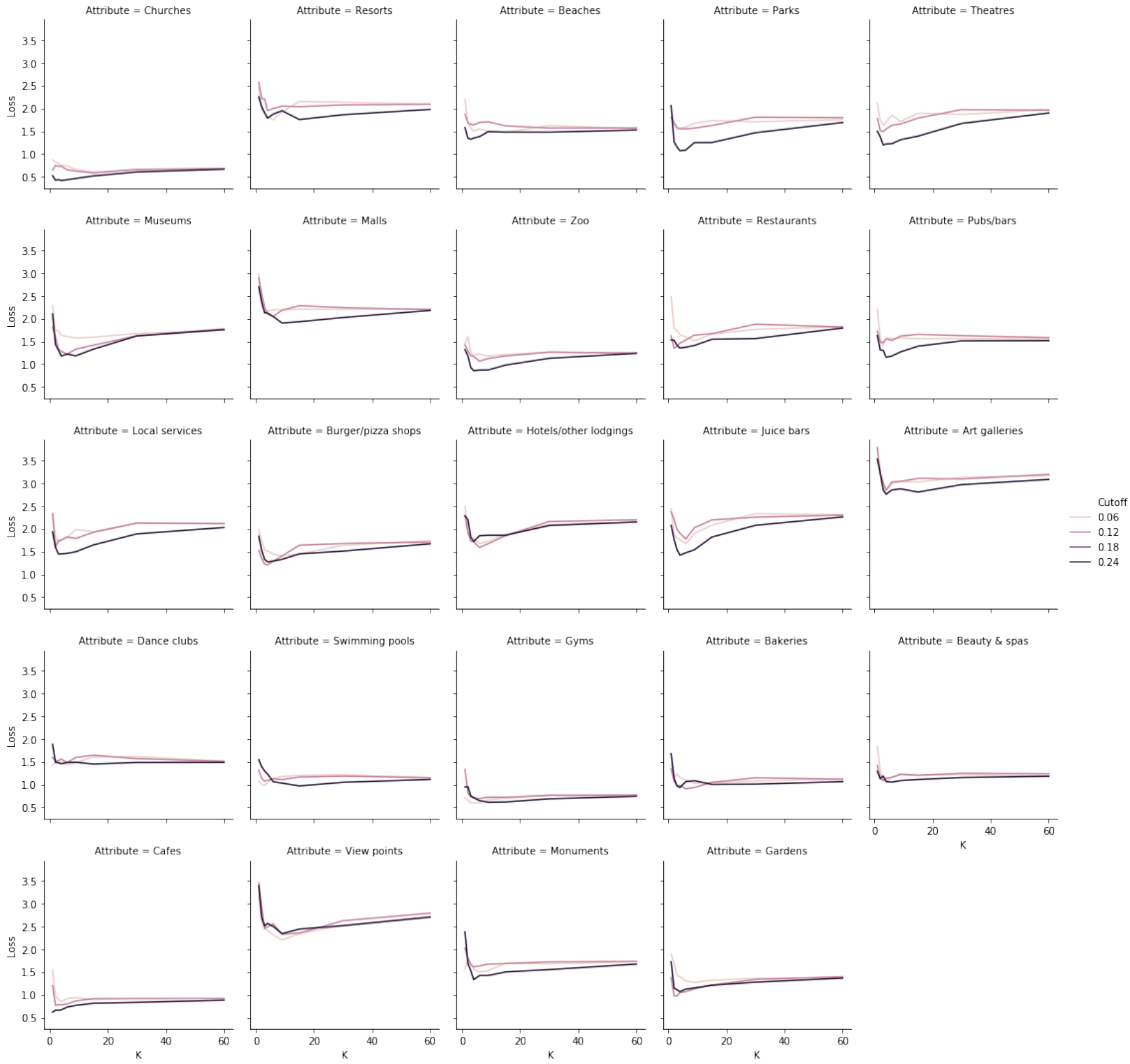

Figure 5: Average Square Loss v.s. K and correlation cutoff

3 Misc

These are a couple of other things I did.

3.1 More Accurate Graphs

Based on the data computed for loss v.s. K and correlation cutoffs, selected the attributes whose graphs varied the most over different values (those with the greatest variance for the loss of each attribute over all K and cutoff values). For these few attributes, more accurate graphs were computed (larger sample sizes and more samples in the procedure described in the first paragraph of "Prediction Results"). The result is shown in figure 6.

3.2 Approximated Loss Based on Sample Size

To see how the sample size affected the approximated loss, the losses were graphed for a few attributes over both K and sample size. Results shown in figure 7

3.3 Normalization

Attempted to reduce loss by normalizing feature vectors, so that users who gave similar relative reviews (rather than absolute ratings) contribute more to the predicted rating. This was done in the hope that users with similar relative reviews would be better predictors. This did not improve loss (didn't seem to make any difference).

3.4 Probability Integral Transform

Although I did not end up using it to transform the data and see if it improved prediction, I did write a function to compute an approximate probability integral transform (i.e. a function that transforms a distribution into a uniform distribution) of data given by a histogram.

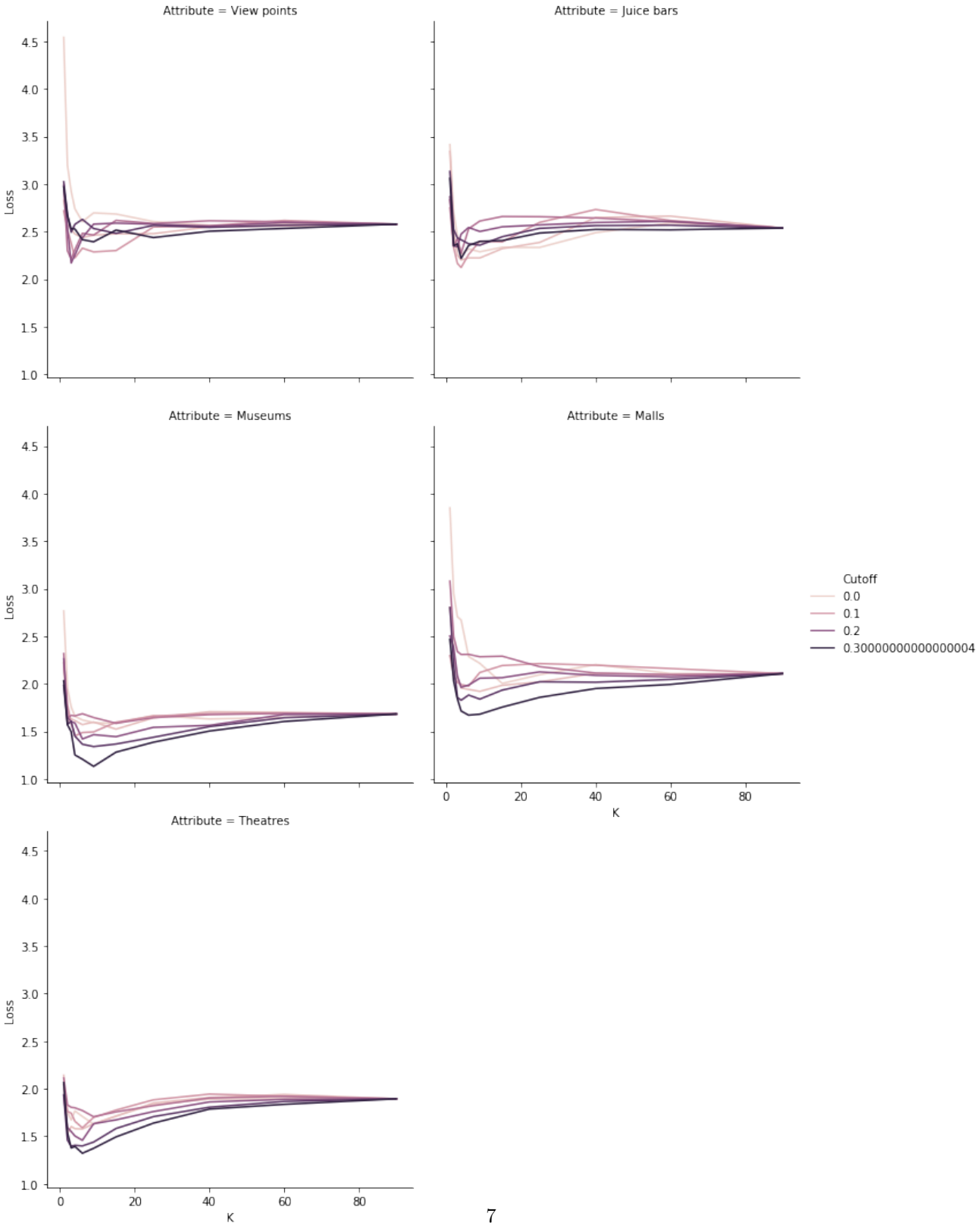

Figure 6: Average Square Loss v.s. K and correlation cutoff

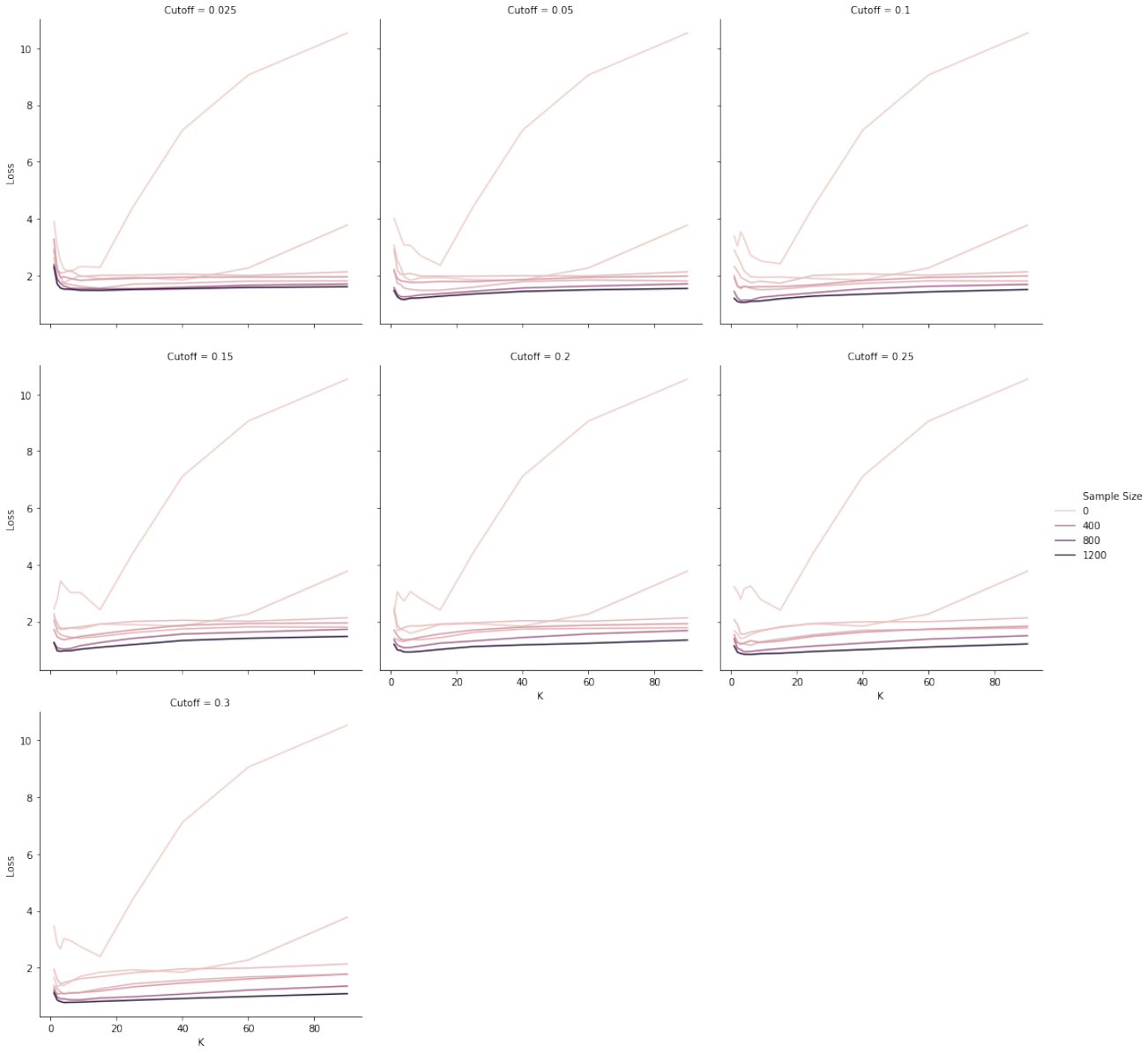

Figure 7: Average Square Loss v.s. K and sample size