During the EDA process, I found touristy cities, like Las Vegas in this case, contain higher review counts with moderate population. This affects the exploratory result of the relations between review counts and population. The 5,965 records that in Las Vegas are removed to improve the accuracy of the mode at the beginning. The Pearson correlation coefficient was about 0.099 and now 0.12. I randomly selected 250 records from rest of the dataset, which has a size of 27,324.

Since a food businesses is good or not is subjective, there's no definite answer to this model. This model shows if a food business is highly possible to be good compared to others in the area. Because the nature of the model, I decided to use clustering, and make the conclusion based on the relations between clusters. I applied K-Means clustering with divisive hierarchical clustering approach, first by population and next by review count. find_best_k function was created to help find the best number of cluster that generate the best model. The range of k to test on is from 2 to 8. Silhouette score is used to help identify the best clustering model. I also tried Agglomerative clustering. However, the dendrogram was visually difficult to observe the relations. Moreover, I only need two hierarchies. Therefore, I did not proceed with this approach.

After couple runs of the codes, I found the best number of clusters is either 5 or 7 depends on the 250 random samples. The second clustering mostly fall on 2 sometimes 3 or 4. In order to get more clusters to observe, if the best number is 2, I would use 3 instead. From the skewness score we can see the review count data is highly skewed to the right, which indicates most of the review counts are at lower numbers. This also indicates most good food places would fall in the small review count range, thus, I should choose the cluster groups with the small review count range as the reference for good businesses. I organized statistic results into a dataframe to compared. In addition, I ranked population range and review count range of each cluster from small to big. As the statistics show, the smaller the review count range is, the more estimated good food businesses it contains.

		population_rank	review_cnt_rank	estimated_good_cnt
cluster_label	**cluster_label_2**			
0	**0.0**	2	1	4.0
	1.0	2	3	0.0
	2.0	2	2	1.0
1	**0.0**	7	2	1.0
	1.0	7	1	8.0
	2.0	7	3	1.0

2	**0.0**	6	1	9.0
	1.0	6	2	2.0
	2.0	6	3	0.0
3	**0.0**	1	2	5.0
	1.0	1	1	32.0
	2.0	1	3	1.0
4	**0.0**	4	2	3.0
	1.0	4	3	1.0
	2.0	4	1	3.0
5	**0.0**	5	2	2.0
	1.0	5	3	0.0
	2.0	5	1	3.0
6	**0.0**	3	2	4.0
	1.0	3	1	13.0
	2.0	3	3	2.0

Exploring more on how likely the clustering could determine the good businesses, I decided to sample another 250 records. I divided the dataset without Las Vegas into five groups using percentile at 20, 40, 60, 80 and 100. 50 records are randomly selected from each group. The initial assumption of good food businesses are defined as four stars and above in certain review count range based on population of the area. To find that review count range, I first applied logarithm to reduce the skewness of the review count data. Next, I computed the upper bound and lower bound by adding or subtracting one standard deviation from the mean, then "converted" the range back by applying exponential. The food businesses with four stars and above in this review count range are labeled as good. This process was repeated on the 50 samples in all five groups. The 50 samples from each group are combined after labeling as the second sample dataset, which has the same size of 250 as the first sample dataset used in clustering. I used the same clustering approach on the second sample dataset, and created a similar dataframe as above. From the dataframe, we can see statistically labeled good businesses are less than the ones from clustering. However, some actually has the same counts in both columns.

		population_rank	review_cnt_rank	good_count	estimated_good_cnt
cluster_label	**cluster_label_2**				
0	**0.0**	8	1	6.0	11.0
	1.0	8	3	0.0	1.0
	2.0	8	2	0.0	7.0
1	**0.0**	3	1	3.0	4.0
	1.0	3	3	0.0	3.0
	2.0	3	2	3.0	4.0
2	**0.0**	7	1	5.0	6.0
	1.0	7	2	1.0	4.0
	2.0	7	3	0.0	0.0
3	**0.0**	1	1	27.0	32.0
	1.0	1	2	1.0	6.0
	2.0	1	3	0.0	2.0
4	**0.0**	6	2	0.0	3.0
	1.0	6	3	0.0	1.0
	2.0	6	1	5.0	7.0
5	**0.0**	5	1	0.0	0.0
	1.0	5	3	0.0	1.0

	2.0	5	2	0.0	2.0
6	**0.0**	4	1	3.0	5.0
	1.0	4	3	0.0	0.0
	2.0	4	2	3.0	3.0
7	**0.0**	2	1	5.0	5.0
	1.0	2	3	0.0	1.0
	2.0	2	2	0.0	1.0