Data analysis Project 3:

Applying machine learning methods to movie ratings data

Doma Ghale

Center for Data Science

DSGA 1001: Introduction to Data Science

Due date: December 23th 2021, 11:59pm

The aim of this project is to use Principal Component Analysis, kMeans clustering, logistic regression and Convolution Neural Network to examine movie ratings and behaviour questions. For my analysis, I used python programming language and some packages such as *pandas, sklearn* and *torch*.

1) I performed PCA on columns 421-474 of 'movieReplicationSet.csv'. First of all, I filled the missing data with the median value of that column. Then, I normalized the data. The next step was to compute the covariance matrix of that normalized data and compute their eigenvalues and eigenvectors (loadings). I used *sklearn* library's *PCA()* function, which does the necessary steps for us . To select the number of factors, I used the Kaiser criterion by keeping all factors with an eigenvalue > 1. As shown in the scree plot (Figure 1.1) we have 10 eigenvalues that satisfies Kaiser criterion. Those ten factors account for 55.49% of the variance. For each of the 10 principal components, I checked couple of the maximum absolute value of its loadings and the corresponding questions, and gave them the following names: "lively", "emotional", "reticent", "self-conceit", "creativity", "indolent", "temperamental", "absentminded","artistic", "ingenious" (Figure 1.2).

2) I used the 10 loadings to transform the original data from columns 421-474 in the new coordinate system. Figure 2 shows the plot, where each dot represents a person and the axes represent a pair of factors I found.

3) I used kMeans clustering to identify clusters in this new space. Using the silhouette method I found 2 as an optimal number of clusters between the first two principal components. Figure 3.1 shows the graph of silhouette scores for 2 to 10 clusters and figure 3.2 shows the graph of the original data in the new coordinate system with two colors– one for each cluster.

4) Next, I used the transformed original data (columns 421-474) with the 10 loading as predictors for the 400 movie ratings. To do so, I built logistic regression models. First, I filled the missing values with the median value of that movie. Then, I re-coded the movie ratings as high (1) or low (0) using the median of that movie as a threshold. I used 10 fold cross validation to avoid overfitting. I specifically used stratified cross validation so that the ratio between two classes (high vs low) is preserved. Doing so the average accuracy for the 400 movie ratings was 0.90 and the average auc score was 0.61. I looked at the breakdown of the scores and found that 62% of the movies have great accuracy (>0.90) and only 10% of the movies have good auc-roc (>0.70). Figure 4 shows the histogram of the average accuracy and auc-roc scores for all 400 movies.

5) Lastly, I created a one dimensional Convolutional Neural Network (CNN) to predict movie ratings (400), using information from personal questions (77). Note that the columns were not transformed using pca. I ran several iterations of different combinations of models. For my first model, I filled missing values of all of the columns with the median value of that column. Next, I

ran a CNN model using 77 columns to predict each of the 400 movie ratings using 3 epochs and learning rate of 1e-2. I made sure to set a random seed such that my results are reproducible. For this model I was not able to use stratification sampling to split the dataset into train and test because for some movies not all 10 ratings were present.

I noticed the average model performance was around 35.84%, which is not good at all. So, I looked at the model performance for individual movies and noticed some movies had great accuracy (>90%) and some movies had poor accuracy (<10%). I investigated why some movies were performing so poorly and found that, mostly the movies that had a very high number of missing data were replaced with the median value of that movie. That was one of the reasons why my model did not perform well.

Next, I reran the model after standardizing the predictors and the average accuracy for all of the 400 movies was still at 35.84%. I wondered if predicting the actual ratings was too granular because the difference between two ratings might have very subtle differences. To test that I selected the first movie "The Life of David Gale (2003)," and ran different models based on how I cleaned and selected the movie ratings and predictors for three types of outcomes: (I) actual 10 ratings, (II) binary outcome by assigning ratings >= median as 1 and 0 otherwise, and (III) three outcomes by assigning ratings > median as 2, ratings = median as 1 and 0 otherwise. For those three cases I did the following four types of data cleaning scenarios:

1. fill all missing data with median
2. fill all missing data with median and standardize predictors
3. fill only predictors with median and remove missing ratings
4. fill only predictors with median, standardize them and remove missing ratings

I used 3 epochs and a range of learning rates (1e-2, 1e-3, 1e-4).

For Case I scenario 1, the accuracy for three different learning rates were [0.5, 0.5, 0.5]. Those accuracies are worse than random guesses. If we look at the predicted outputs of the test data (20% of original data = 220) and their actual values, we see that the only rating predicted by the model was "2" and there is only 1 label that is rating "2". Therefore, 1/220*100 = 0.5. Same results for scenario 2. For scenario 3, accuracy was [10.4, 10.4, 2.07]. First two learning rates are better than the learning rate of 1e-4, but no better than random guesses. The accuracy of each epoch tells us that increasing epoch doesn't always improve the accuracy: first learning rate [12.5, 6.2, 12.5], second learning rate [18.8, 6.2, 6.2], and third learning rate [0.0, 0.0, 6.2]]. For scenario 4, accuracy was [10.4, 16.7, 8.33]. Accuracy for the learning rate of 1e-3 is slightly better, but not good enough.

For Case II scenario 1, accuracy was [96.8 96.8 96.8]. The accuracy looks good, but in reality it is not. 1021 rows (93%) are missing so filling them with median meant that if we categorize the

ratings into high vs low using median as a threshold, a high proportion of the movies were grouped as 1 or 0 based on whether we use >= or > median. I had used >= median as 1 and otherwise 0. The test set had 213 cases of "1" and 7 cases of "0" and the model predicted all 220 cases as "1", so the accuracy came out to be 213/220 = 98.8%. Therefore, when assessing the model performance it is important to look beyond accuracy and see what the base rate is. I went ahead and did the rest of the three cases and I was not happy with my model performances. So, instead I decided to categorize the ratings into three categories: 0 if below median, 1 if at median and 2 if above median. This is a better split than the binary outcomes because of the highly uneven ratios, and better than 10 outcomes because that level of granularity might have very subtle differences. The model performances for 3 outcomes were better than 10 and 2 outcomes as I expected and shown in figure 5. Note that accuracy for both case II and case III was around 50% when I removed the missing ratings, but predicting 50% accurately for three outcomes is better than 50% accuracy for two outcomes. When case II and case III are compared, it is surprising to me that standardizing the predictors did worse than not standardizing them for scenario 3 and 4.

Based on the comparison from figure 5, case III and scenario 3 seems to be the best combination. But, I am also worried that for movies with a lot of missing data the number of data in the testing set would be very low and the accuracy would be affected by it. So, I decided to rerun the model for all 400 movies using Case III and both scenario 1 (filling all missing data with median) and 3 (filling missing data for predictors only with median, and removing rows with missing movie ratings) with learning rate 1e-2 and epoch 3. The final accuracy for 400 movies for scenario 1 was 79.81% and scenario 3 was 44.17%. The only difference between my very first model and scenario 1 case III is the number of outcomes (10 vs 3), and as I expected the accuracy jumped from 35.84% to 79.81%. Additionally, I tried running my model with epoch 5 and the accuracy did not improve (79.78%).

Based on my findings from running tests on CNN, I would go back and rerun the logistic regression model after assigning the ratings into three categories. I would also rerun the CNN model with the result from the principal component analysis done in part 1 of this project and see how they compare with logistic regression results.

Appendix

Figure 1.1: Scree plot the number of factors (principal components)

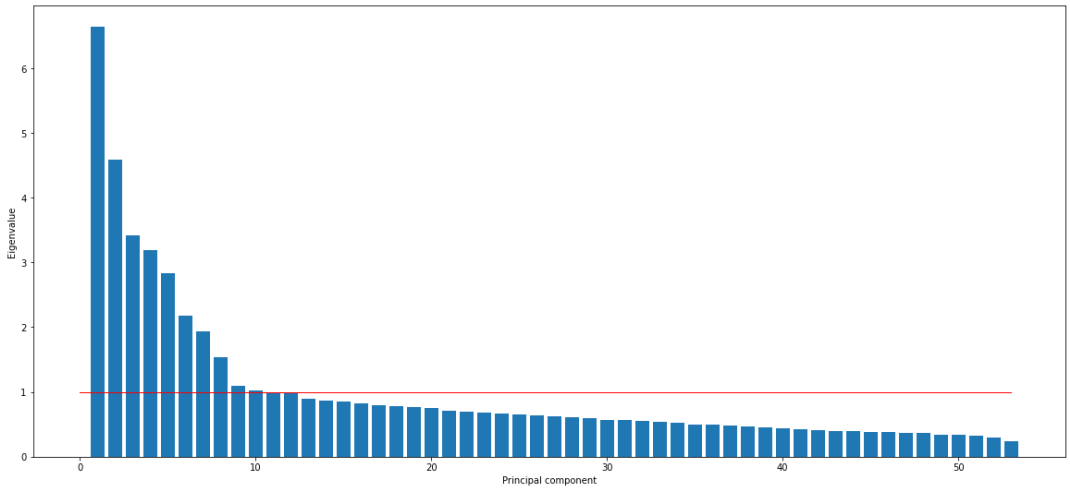


Figure 1.2: Loadings for the first 10 principal components

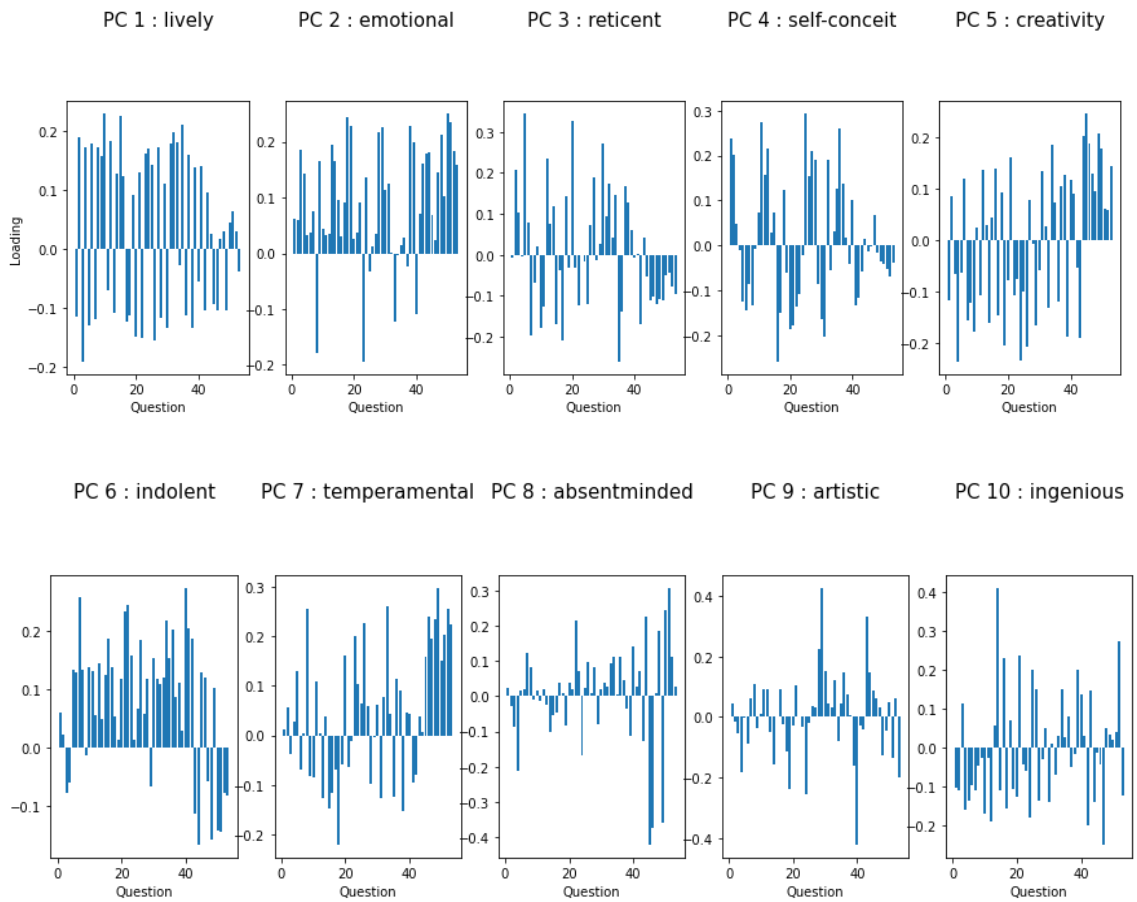


Figure 2: Plot the data from columns 421-474 in the new coordinate system

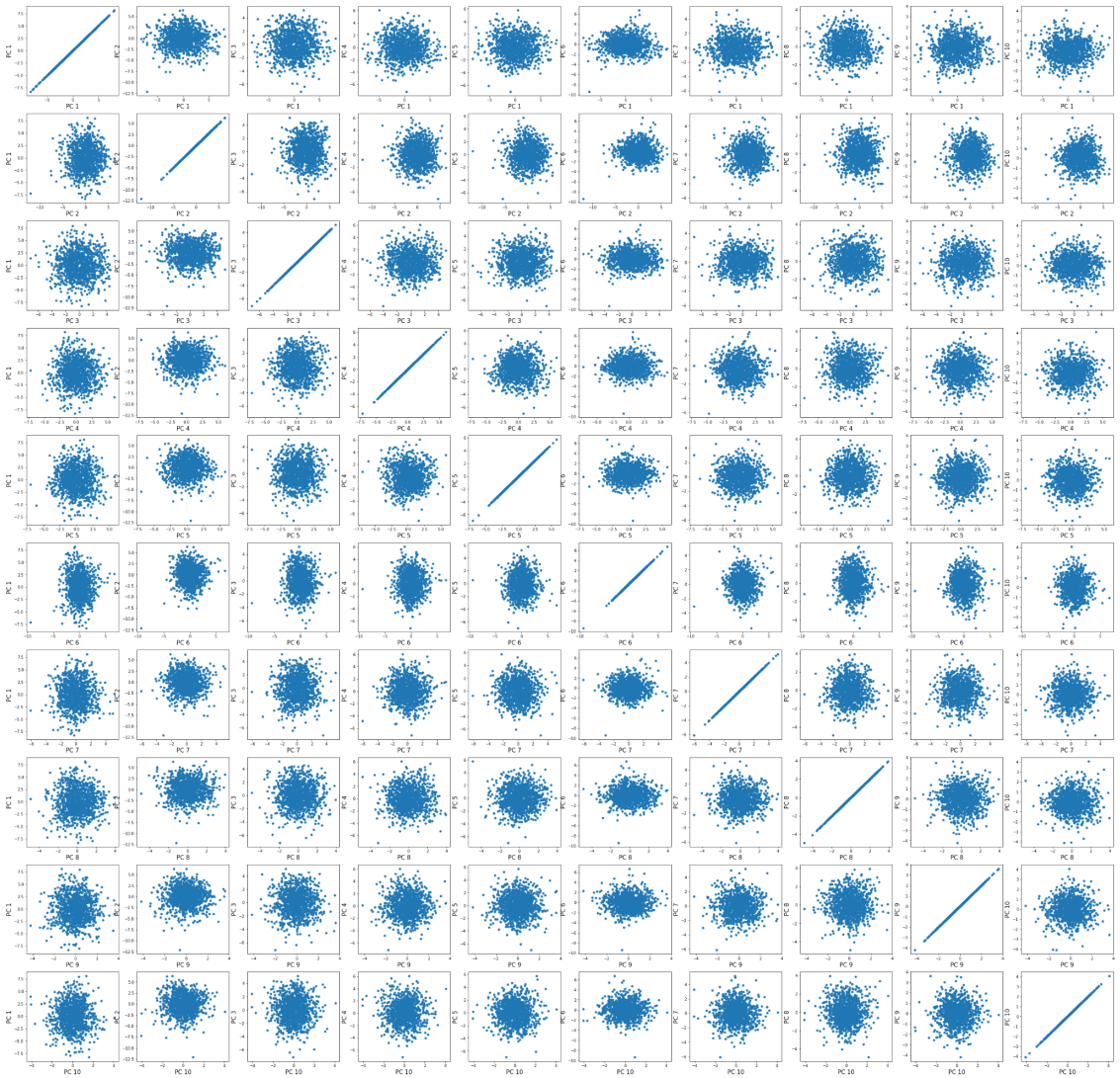


Figure 3.1: Silhouette scores for 2 to 10 clusters

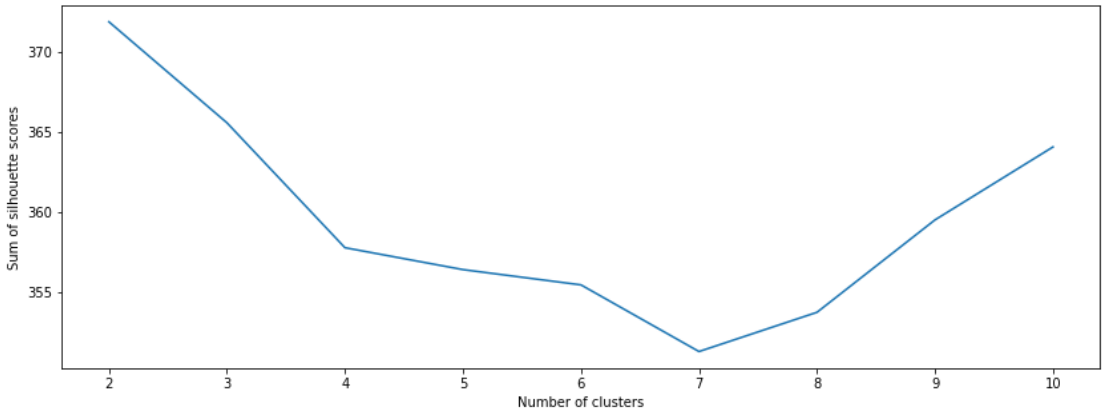


Figure 3.2: Plot the data from columns 421-474 in the new coordinate system

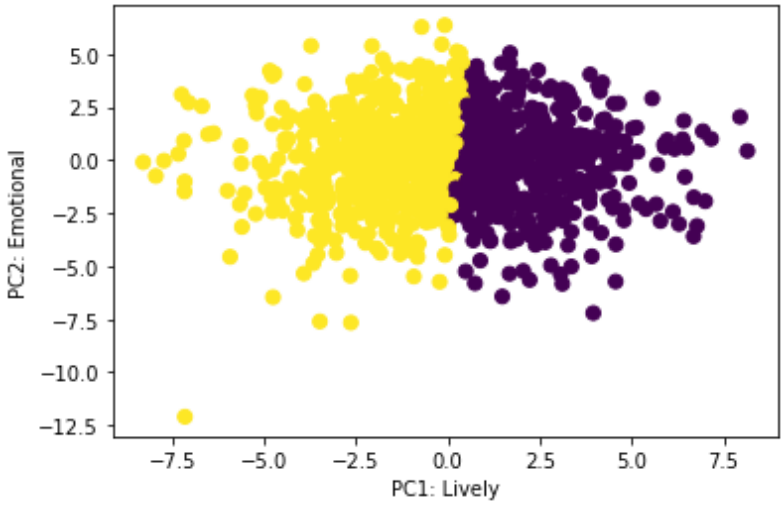
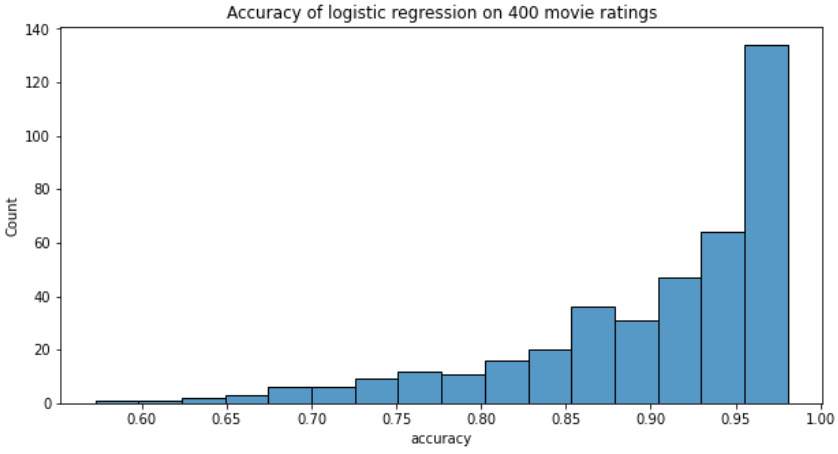


Figure 4: Histogram of the average accuracy and AUC-ROC score for all 400 movies



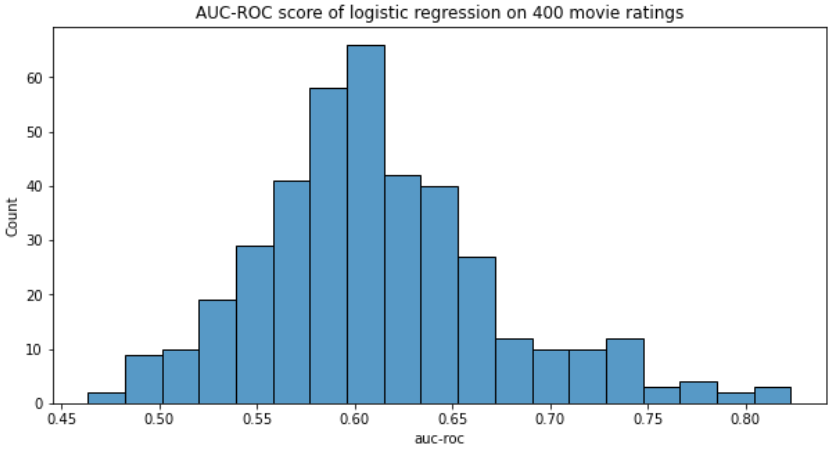


Figure 5: Accuracy for learning rates (1e-2, 1e-3, 1e-4) and occurences of each outcomes in actual label vs predicted by the model for three different outcome classes

```
Predicting 10 outcomes
Model 1
[0.5 0.5 0.5]
actual labels: {0.0: 1, 0.5: 2, 1.0: 2, 1.5: 2, 2.0: 1, 2.5: 207, 3.0: 3, 3.5: 1, 4.0: 1}
model output 0.01 : {2: 220}
model output 0.001 : {2: 220}
model output 0.0001 : {2: 220}
None

Model 2
[0.5 0.5 0.5]
actual labels: {0.0: 1, 0.5: 2, 1.0: 2, 1.5: 2, 2.0: 1, 2.5: 207, 3.0: 3, 3.5: 1, 4.0: 1}
model output 0.01 : {2: 220}
model output 0.001 : {2: 220}
model output 0.0001 : {2: 220}
None

Model 3
[10.4 10.4  2.1]
actual labels: {0.0: 1, 0.5: 2, 1.0: 2, 1.5: 2, 2.0: 1, 2.5: 2, 3.0: 3, 3.5: 2, 4.0: 1}
model output 0.01 : {0: 1, 1: 10, 3: 4, 4: 1}
model output 0.001 : {2: 2, 3: 14}
model output 0.0001 : {2: 16}
None

Model 4
[10.4 16.7  8.3]
actual labels: {0.0: 1, 0.5: 2, 1.0: 2, 1.5: 2, 2.0: 1, 2.5: 2, 3.0: 3, 3.5: 2, 4.0: 1}
model output 0.01 : {1: 9, 2: 1, 3: 6}
model output 0.001 : {0: 1, 1: 8, 2: 3, 3: 4}
model output 0.0001 : {0: 7, 1: 4, 2: 4, 3: 1}
None
```

```
Predicting 2 outcomes
Model 1
[96.8 96.8 96.8]
actual labels: {0: 7, 1: 213}
model output 0.01 : {1: 220}
model output 0.001 : {1: 220}
model output 0.0001 : {1: 220}
None

Model 2
[96.8 96.8 96.8]
actual labels: {0: 7, 1: 213}
model output 0.01 : {1: 220}
model output 0.001 : {1: 220}
model output 0.0001 : {1: 220}
None

Model 3
[50.  50.  58.3]
actual labels: {0: 8, 1: 8}
model output 0.01 : {1: 16}
model output 0.001 : {1: 16}
model output 0.0001 : {1: 16}
None

Model 4
[45.8 41.7 47.9]
actual labels: {0: 8, 1: 8}
model output 0.01 : {0: 4, 1: 12}
model output 0.001 : {0: 10, 1: 6}
model output 0.0001 : {0: 11, 1: 5}
None
```

```
Predicting 3 outcomes
Model 1
[94.1 94.1 94.1]
actual labels: {0: 7, 1: 207, 2: 6}
model output 0.01 : {1: 220}
model output 0.001 : {1: 220}
model output 0.0001 : {1: 220}
None

Model 2
[94.1 94.1 94.1]
actual labels: {0: 7, 1: 207, 2: 6}
model output 0.01 : {1: 220}
model output 0.001 : {1: 220}
model output 0.0001 : {1: 220}
None

Model 3
[54.2 50.  50. ]
actual labels: {0: 8, 1: 2, 2: 6}
model output 0.01 : {0: 12, 2: 4}
model output 0.001 : {0: 16}
model output 0.0001 : {0: 16}
None

Model 4
[45.8 45.9 50. ]
actual labels: {0: 8, 1: 2, 2: 6}
model output 0.01 : {0: 13, 2: 3}
model output 0.001 : {0: 15, 2: 1}
model output 0.0001 : {0: 16}
None
```