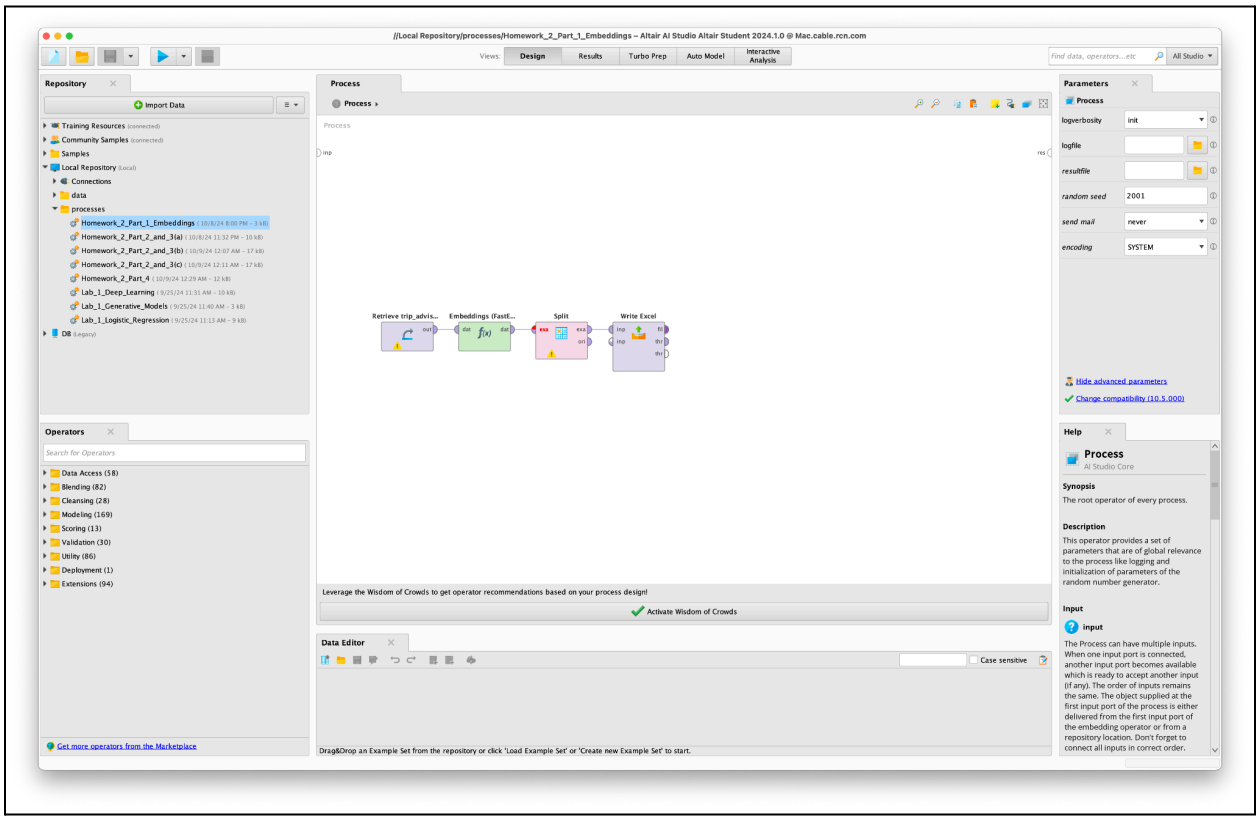
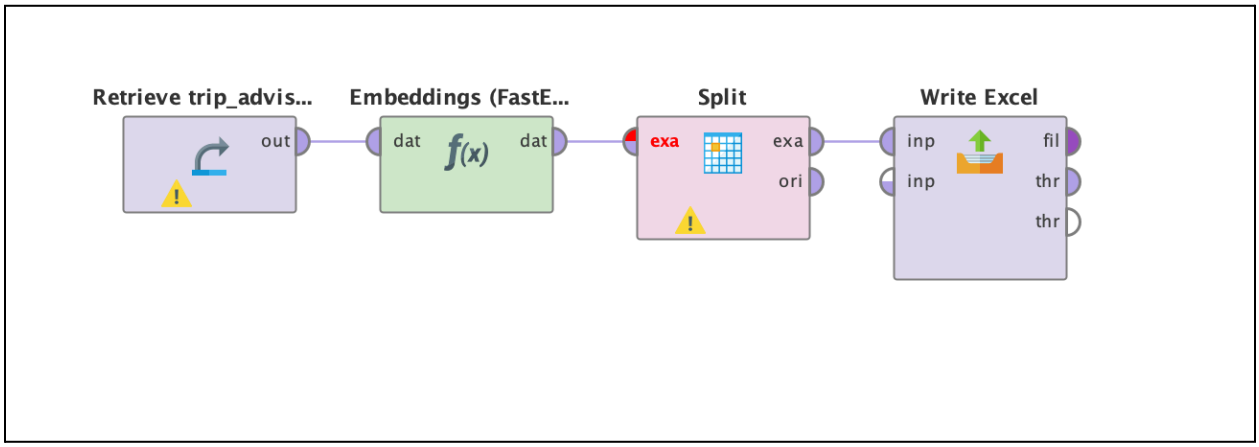
Assignment 2

Aneesh Soni

Part 1

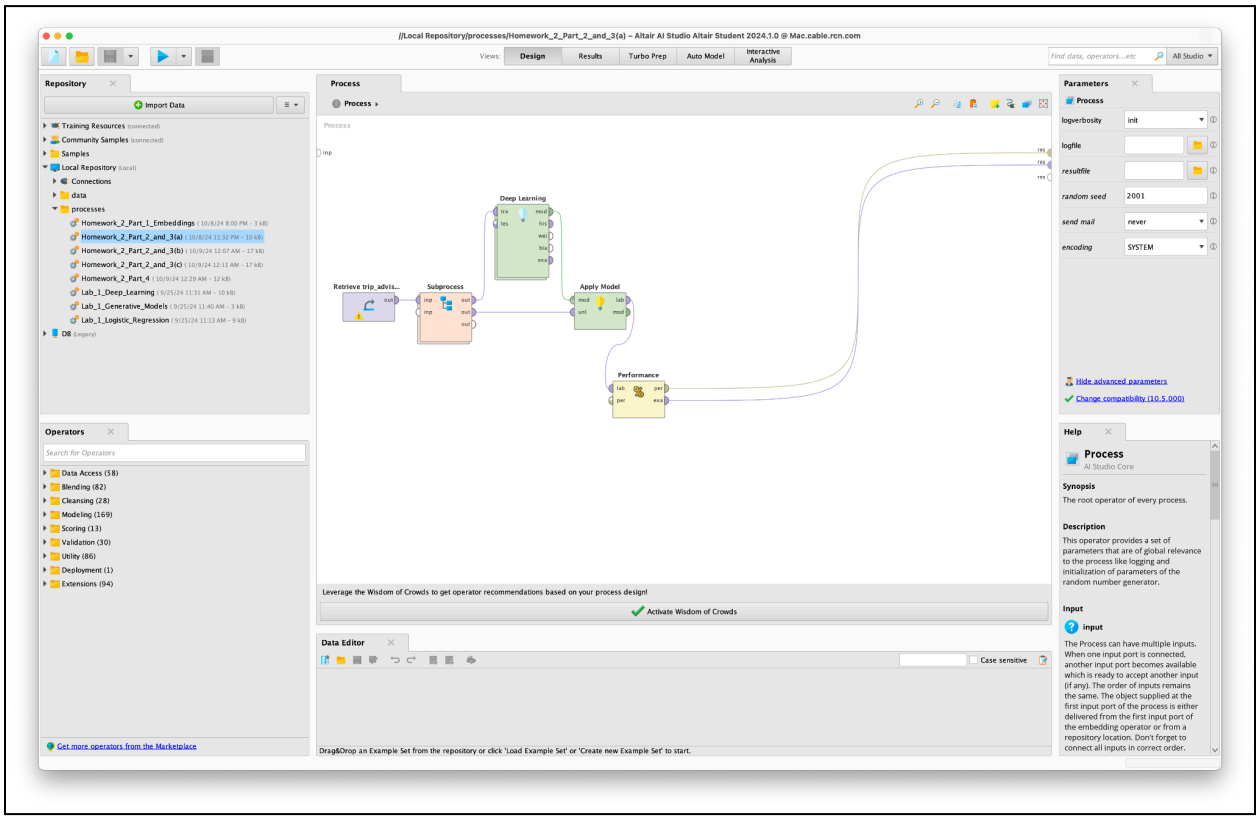
No questions asked but I've provided a screenshot of my model design and a zoomed in version of the modules

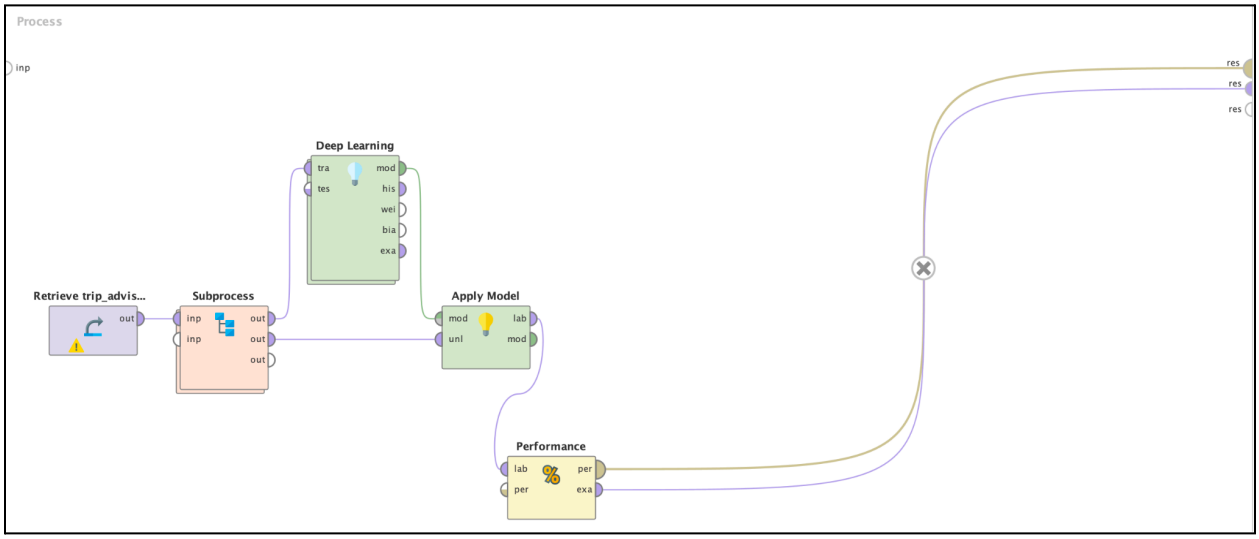




Part 2

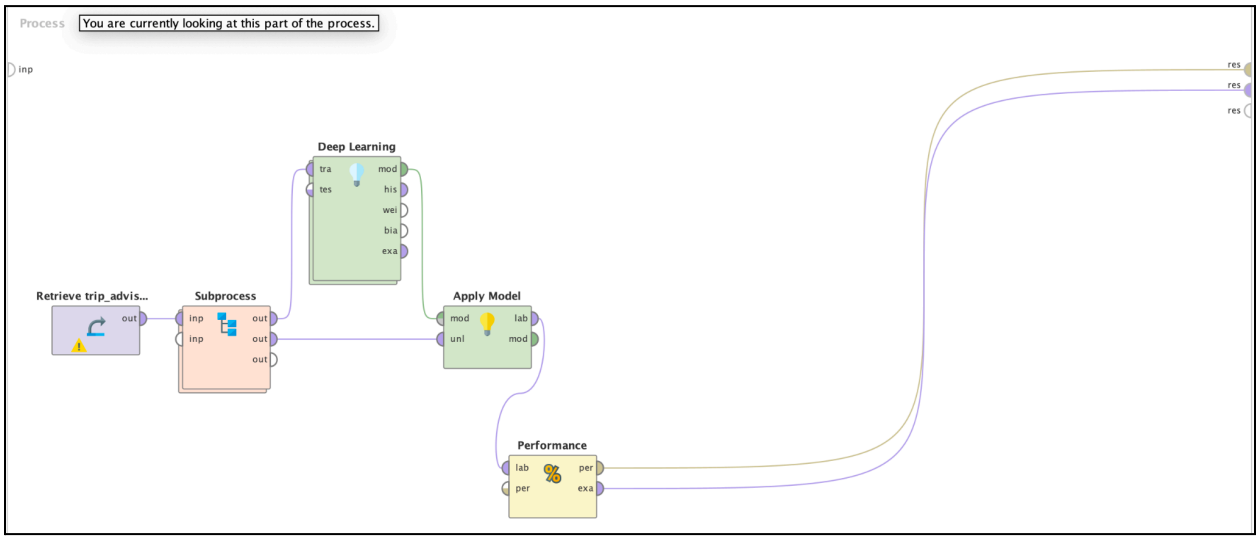
No questions asked but I've provided a screenshot of my model design and a zoomed in version of the modules

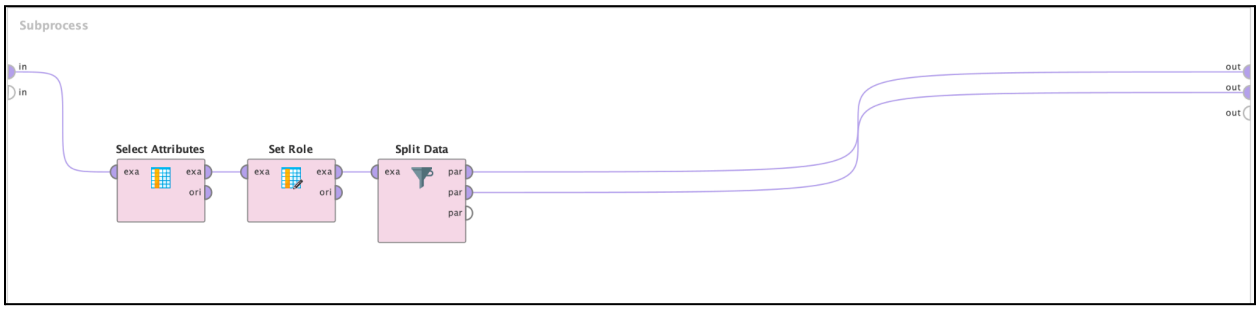


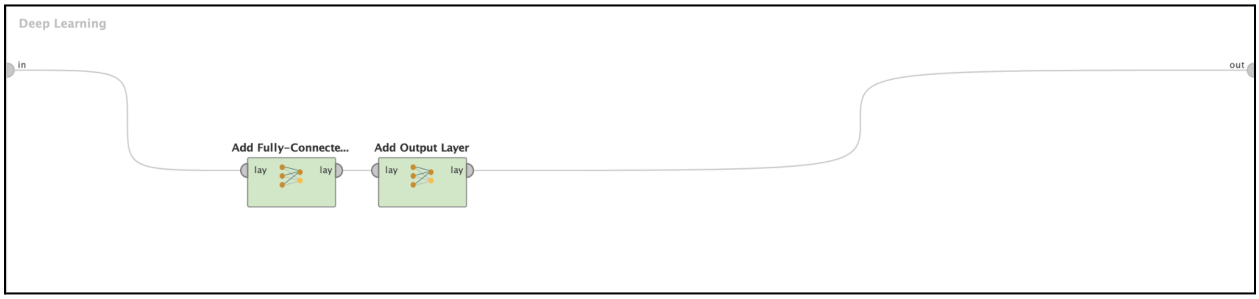


Part 3

Model Design Flowcharts: (Note that the flowcharts are the same for part a, b and c for question 3. As a result I've provided a flowchart of one of them below)

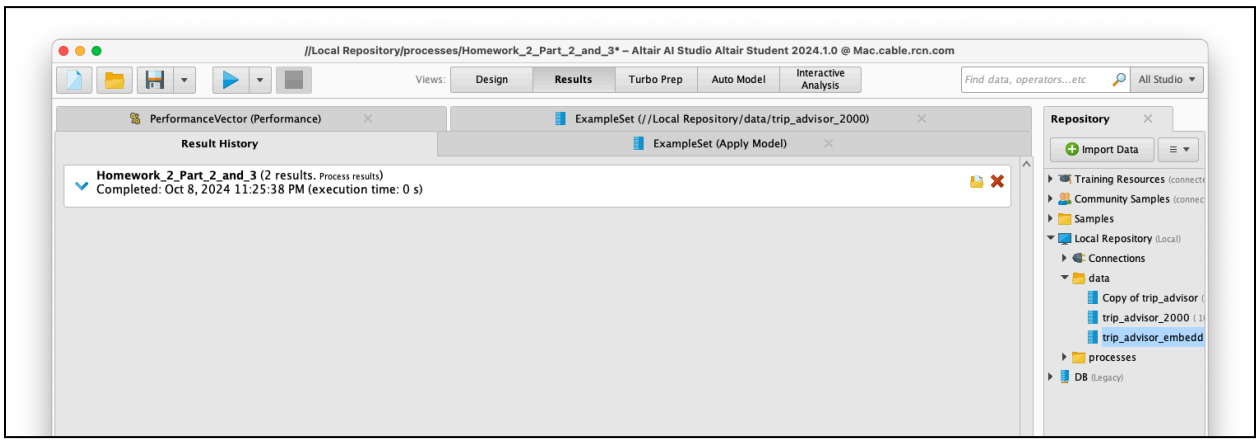






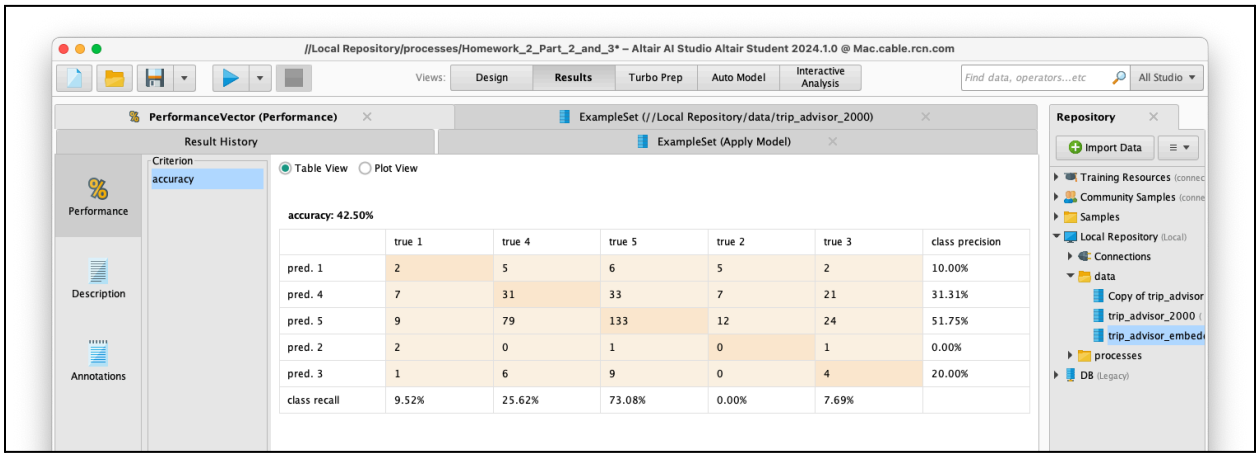
Part 3 (a)

Execution Time: 0 seconds (picture below)



Performance Matrix

Accuracy: 42.5%



	true 1	true 4	true 5	true 2	true 3	class precision
pred. 1	2	5	6	5	2	10.00%
pred. 4	7	31	33	7	21	31.31%
pred. 5	9	79	133	12	24	51.75%
pred. 2	2	0	1	0	1	0.00%
pred. 3	1	6	9	0	4	20.00%
class recall	9.52%	25.62%	73.08%	0.00%	7.69%	

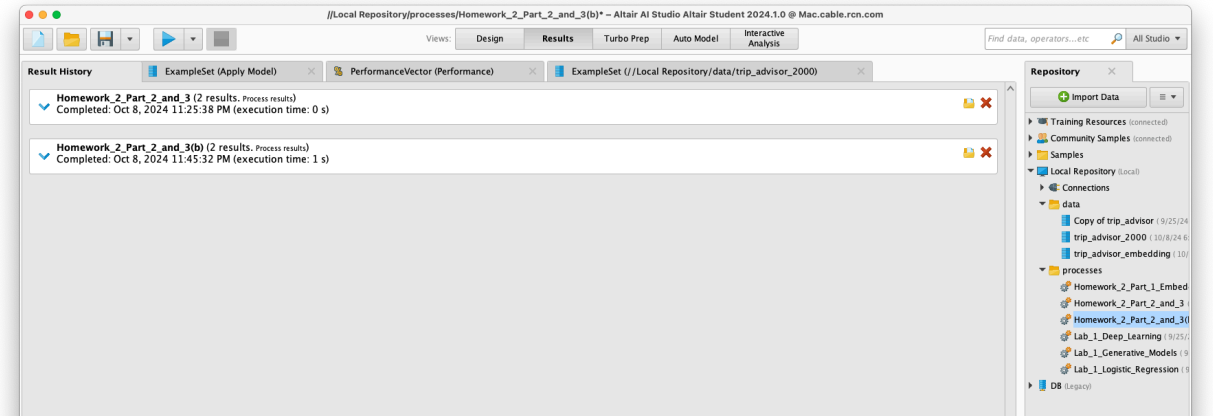
Explanation:

Comment briefly on the performance results

The Simple Feedforward Network Based on Ratings approach ultimately yielded a low accuracy of 42.5%. This is likely due to the approach used here and the limited selection of attributes we used for prediction. The model had the best class recall for **true 5** at 73.08% which indicates it is pretty good at indicating instances of this particular class, and especially far better than the other classes (i.e. class recall for **true 2** was 0%). Similarly, the class precision is also highest for **true 5** indicating that when the model predicted **5** it was right 51.75% of the time.

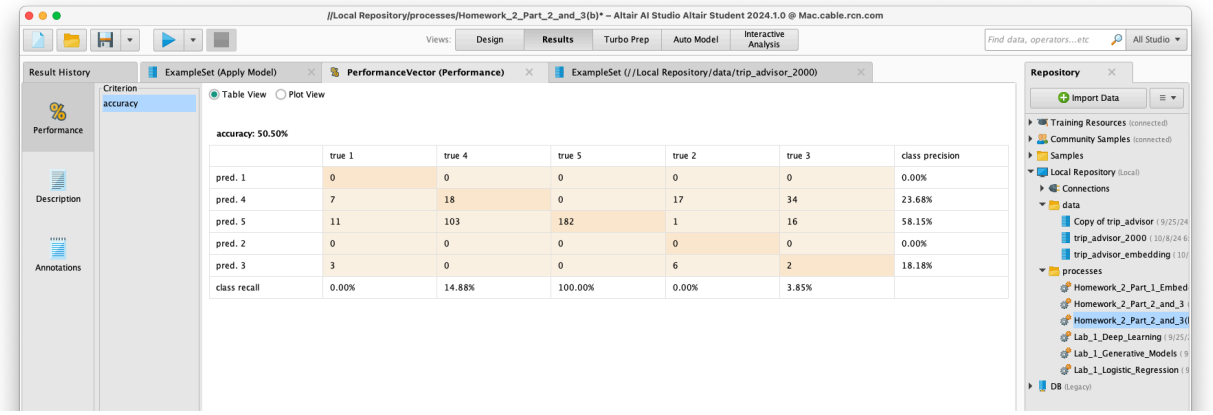
Part 3 (b)

Execution time: 1 second (picture below)



Performance Matrix

Accuracy: 50.50%



Explanation

How would these performance results compare with those achieved by the Ratings-based model from Part 3(a)?

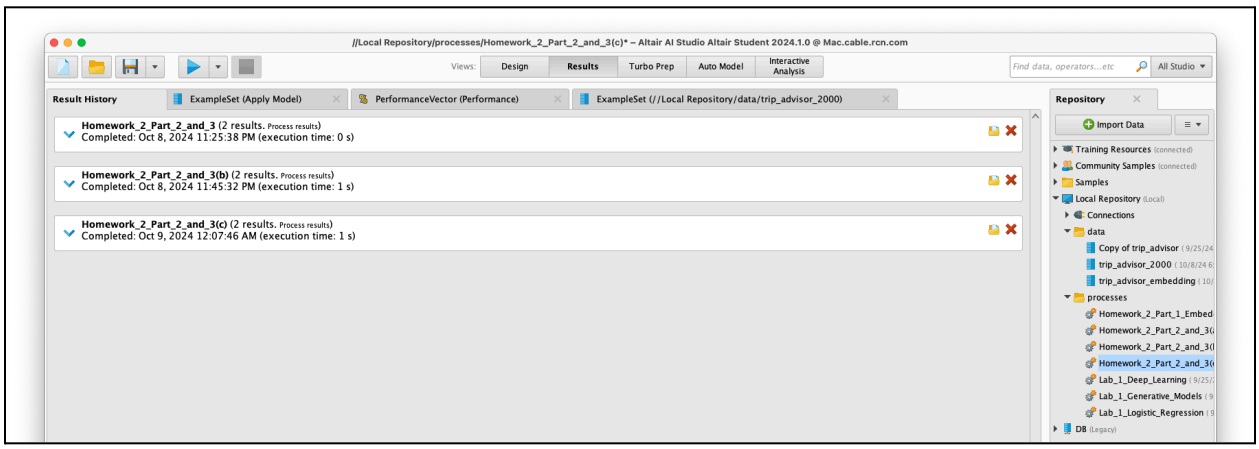
The performance results, as measured by accuracy, for the model in part 3(b), accuracy of 50.50%, is better than that of the Ratings-based model from Part 3(a), accuracy of 42.5%.

How would you explain the performance differences?

This indicates that we can derive more meaning and a better ability to predict when we model using the embeddings of written reviews versus simply numerical/categorical review ratings. By leveraging vector embeddings we can achieve deeper sentiment analysis of written text than purely numerical data. Once again, this model does well with predicting **class 5**. However, this isn't a substantial enough difference overall.

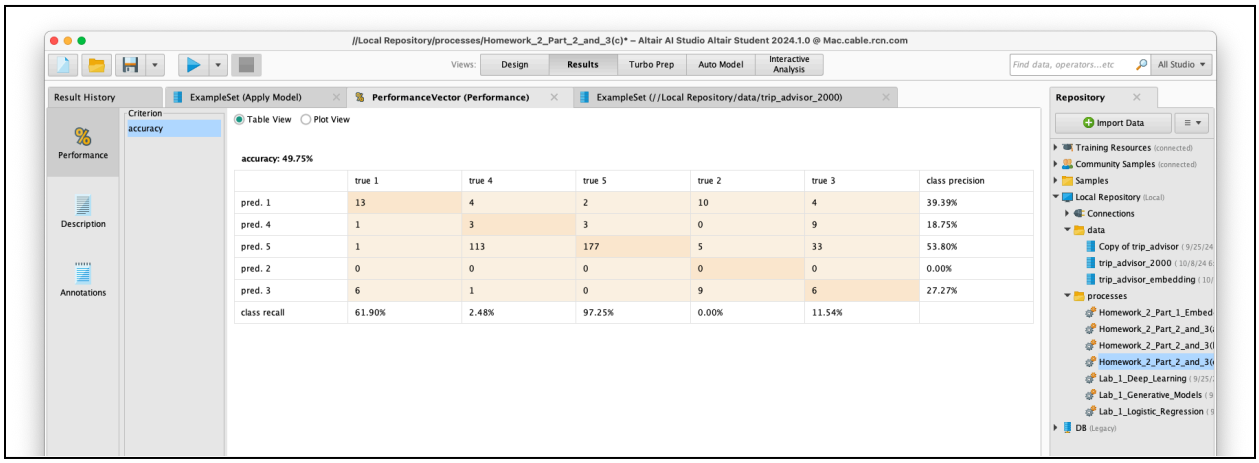
Part 3 (c)

Execution Time: 1 second (picture below)



Performance Matrix

Accuracy: 49.75%



Explanation

How would these performance results compare with those achieved by the models from Parts 3(a) and (b)?

Surprisingly, this is lower than just the embedding model from 3(b) but better than just the numeric ratings from 3(a). This has an accuracy of 49.75%.

How would you explain the performance differences?

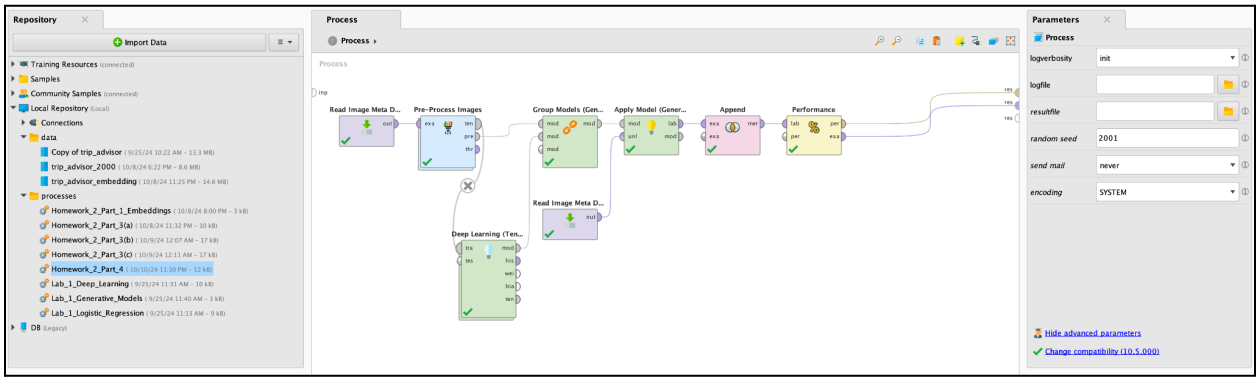
This indicates to me that the numeric ratings actually “confuse” the model potentially due to their lack of specificity versus written ratings that we converted into embeddings. We perhaps have redundant or noisy data that could cause the model to either overfit or incorrectly assess the usefulness of a particular feature.

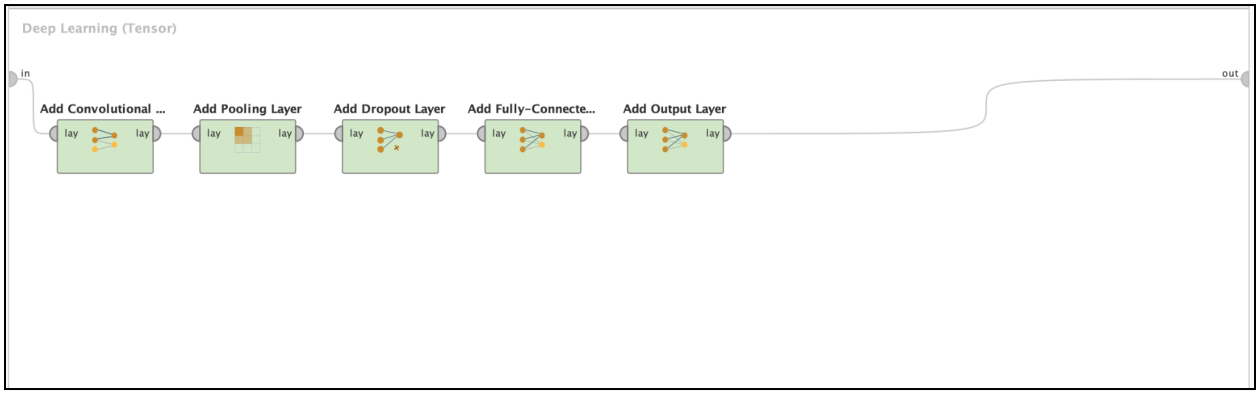
Which of these three models would you pick for the Trip Advisor if being asked by them as a consultant and why?

Given the relatively low accuracy of all 3 models (none performing much better than 50%) I would not recommend any of these 3 models. They don't provide the accuracy you need in order to make meaningful business decisions off of, however, **if I had to choose** I would opt for the **embedding model** given generation of vector embeddings is not super computationally heavy and yields the best overall results in this instance. Nonetheless, I would recommend to the Trip Advisor team that we work on this problem a bit more to explore additional ways/data sources for us to leverage and yield better results over time.

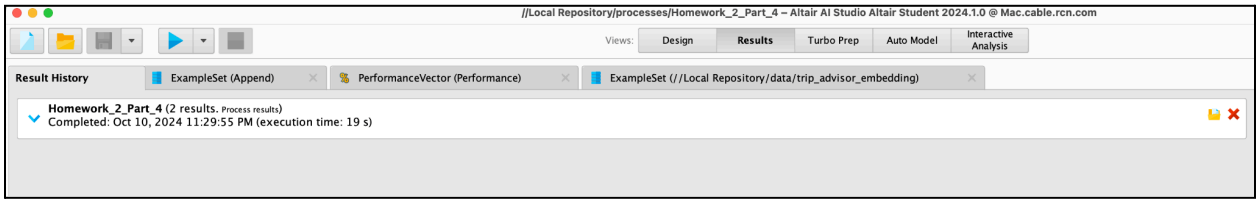
Part 4

Model Design Flowcharts





Execution Time: 19 seconds (picture below)



Performance Matrix (picture on follow page)

Accuracy: 86.00%

accuracy: 86.00%

	true 0	true 1	true 2	true 3	true 4	true 5	true 6	true 7	true 8	true 9	class precision
pred. 0	20	0	0	0	0	1	0	0	0	0	95.24%
pred. 1	0	19	0	0	0	0	0	0	1	0	95.00%
pred. 2	0	0	18	0	0	0	1	1	0	0	90.00%
pred. 3	0	0	0	18	0	1	0	0	2	0	85.71%
pred. 4	0	0	0	0	17	0	0	0	0	2	89.47%
pred. 5	0	0	0	0	0	17	1	0	2	0	85.00%
pred. 6	0	0	0	0	0	0	18	0	0	0	100.00%
pred. 7	0	0	2	0	0	0	0	13	0	0	86.67%
pred. 8	0	1	0	0	0	0	0	0	14	0	93.33%
pred. 9	0	0	0	2	3	1	0	6	1	18	58.06%
class recall	100.00%	95.00%	90.00%	90.00%	85.00%	85.00%	90.00%	65.00%	70.00%	90.00%	

Explanation

Comment on the performance results, i.e., how good they are and what they mean in the context of this prediction problem.

The model has an 86% accuracy overall. In specific, it seems to do quite well at recognizing 0, 1, 2, 3 and 6 as these have good recall and precision. Predicting **9** has low precision indicating that the model isn't good at positive predictions for this number (i.e. when it predicts something is a 9 it's wrong quite a bit). But there is high recall for 9 meaning that we do capture most of the true 9's in the dataset. Surprisingly, 1's and 7's are handled quite well by the model which I found interesting considering these can look quite similar many times depending on how each number is written. Overall, this model does a good job at handling the data!