My first step in working on this task was to evaluate the state of the data and see if I needed to make any changes. There weren't any fields with NA values, however, there were some rows in the training dataset whose format was different to the others, i.e., a particular film would have '12' for its revenue, when on Wikipedia its revenue was 12 million. This was a unit of measure issue, however, I decided it would be safer to remove the records which fit into this category in either the revenue or budget fields, as a) while it seemed like a unit of measure issue in the training dataset, I couldn't be sure that the final test dataset wouldn't have more complex issues, and b) some of the values were incorrect according to Wikipedia. Overall it just seemed safer to remove them as they were <=1% of the total dataset for both the training and validation sets. I set the limit for removal at 1,000 (i.e., any film with revenue or budget less than 1000 would be removed), as it seemed very unlikely that any movie could be made for less than 1000 dollars, however, there were cases of low-budget films having high revenue and/or ratings ('sleeper hits'), and I didn't want to remove these or artificially inflate their budget for fear of skewing the model.

The next phase was to expand all the JSON fields and use as much data as possible to train the model so it would make more accurate predictions when evaluating the test dataset. So I tried expanding all the JSON fields in the dataset. Unfortunately, not only was this not a good way to train the model (as I learned when testing the model later on), but it actually wasn't possible to expand all fields as the size of the dataset quickly expanded to around 100 million records and consumed 15 GB of RAM. After realising I wouldn't be able to use all the fields, I chose only those which I thought would be relevant to a film's rating and revenue: the cast (since actors would certainly be a draw for both categories) and the crew (since certain cast members can draw additional revenue as well as increase a film's rating). I also decided that while there was ample data within the JSON fields in the dataset, it would be easiest to whittle it down to just the id for each category – a human being, after all, would understand that a big name would increase a film's revenue, but all the model really needs is the unique ID. This was also because I encountered an issue with cast and/or crew members having the same name, which threw an error for the model as the indices were not unique. After this I encoded each ID as its own column with a '1' value if the ID was present in the film, and a 0 value if the ID was not. I then joined these columns to the original dataframe iteratively, dropped any columns I hadn't used, and began working on the models.

I tested a variety of models for both classification and regression, but found that the inconsistencies between the training and validation sets caused issues: namely, that the training set had more columns because it had more cast/crew. I amended this by normalising the number of IDs that could come from each of the JSON fields, and to ensure there was no overlap I appended a suffix to each ID so the columns would be unique. I was quite disappointed with the models' performance as I'd expected better, but the classifier was around 0.5 accuracy and the regression model was around 0.1 Pearson correlation coefficient (PCC). I decided to adjust some features and narrow down the field of possible features for the models to choose from, as I thought it likely that they were essentially losing the valuable information in the forest of largely irrelevant information. I narrowed down the cast field significantly, and rather than using the most popular actors (by a simple groupby and count) I decided to try using those actors whose gross revenue was highest (by grouping by the actors and the summing the revenue field). This led to a substantial improvement in the regression models (around 0.3-0.4 PCC), however, the classifier actually performed worse after this change so I reverted to the simple count for it.

I decided to whittle down the crew field to only those I thought would influence the movie's rating/revenue, i.e., the directors, writers, and producers. This also led to an improvement in both models, so I continued experimenting by adding new fields (such as genres/keywords) and mixing and matching models, fields, and top features to select. In the end I used Gradient Boosting Classifier for my classification and achieved an accuracy of 0.75, and a Gradient Boosting Regressor for my regression model and achieved a PCC of 0.67. Unfortunately after this I realised that it was incorrect

to use revenue in predicting which values to take, so I explored the dataset again and replaced the sum of total gross earnings with an average of the budget of the film for each ID, which yielded a PCC of 0.5. Finally, as per a comment from one of the tutors, I realised that it was incorrect to take the top X cast/crew/genres etc from the test dataset as this counted as a form of analysis, so I ensured the two datasets were consistent for the model by using exactly the same columns (i.e., if an ID existed in test but not training I'd remove it, if it existed in training but not test I'd add it to test with all 0s as values, and if it existed in both I'd keep it). Using this method my PCC hovered around 0.39-0.42, and my classifier hit an accuracy of around 0.72.