This report will show the problems faced during the process of creating regression and classification models for predicting revenue and rating of films.

Understanding the data and pre-processing

It is essential to understand the given data. Therefore, it was checked the type of data in the columns, missing values, percentiles, unique values (to know the number of categories for rating for example) and it was made scatter plots to visualise outliers in an easier way. After those techniques, it was decided to discard rows with revenues greater than 2B dollars and smaller than 10,000 dollars (it was found that there were small revenues which were considered in another magnitude unit. In addition, just Avatar has a revenue above 2B dollars and there are just 5 films in the history which have surpassed the 2B dollars revenues[1]). There were no missing values in the given data, nevertheless it was created a function to deal with them in case they appeared in the future. Moreover, if a row has more than 1 missing value in some columns, which are considered important[2] for the model, this row is dropped. It was noticed that the model (in the regression part) is misleading when two or more than those columns is missed for a row. Also, it was observed that there was just one sample classified as rating 1 in the training set, so it is an imbalanced data problem.

Another problem of the data was the json format of some columns. To overcome that problem there were designed different functions to extract specific values of the jsons and an own one-hot encoding function was created, because most of the used variables (actor, producers, directors) were categorical variables with more than two categories, so label encoding was not enough.

Selecting features were another big challenge. Variables as 'homepage' or 'title' could be discarded easily, but the others are not the same case. 'Original language' was discarded because most of the films in the data set were English, so it wouldn't be a good predictor. In addition, it was used correlation to know that 'budget' is well correlated with 'revenue' (0.68), but 'runtime' and 'revenue' are just 0.22 (so it was discarded to predict 'revenue'). Nevertheless, 'runtime' and 'rating' has a 0.31 of correlation, so it was keep it (at the end it gave better results using that column than not using it). Finally, the selected original columns were: 'cast', 'crew', 'production_companies', 'budget', 'genres', 'release_date' and 'runtime' (the latter was used just for the classification model).

So, for the columns 'cast', 'crew', 'production_companies' and 'genres' it was used one-hot encoding (it is important to consider that to ranked actors, directors and producer companies it was used frequency criteria), which created so many features. For example, if it is considered the 500 most popular actors, then 500 features are created (one column for each actor). Therefore, in order to reduce the quantity of features it was used Principal Components Analysis (PCA) to know how many actors, producer companies and directors explained most of the variance. Thus, actors were isolated and using PCA it was possible to noticed that the around 600 actors explain the variance of the all 30,000 actors. Same method was used for directors and producer companies.

About the regression model

At the beginning this part was tried to be answered using a liner regression model. As it was mentioned, using PCA it could be noticed that not all actors, directors and producers are needed, nevertheless the numbers are still huge. Then, through trial and error it was found that using the top 25 actors, 120 directors and 120 producer companies in conjunction with 'release_date', 'budget' and 'genres' gave a performance around 0.35, the problem was the overfitting (the performance was almost 0.8 in the training data). One additional problem of the poor performance was that the model was predicting some negative values for revenues. Therefore, it was decided to change the model and it was used Gradient

[1] https://en.wikipedia.org/wiki/List_of_highest-grossing_films

[2] ['cast','crew','budget','genres','production_companies','release_date']

Boosting Regressor. This is an ensemble model and it uses gradient descent. The performance improved immediately, now the correlation in the validation data was around 0.45, but the performance in the training data was still around 0.8 (overfitting). Then, it was tried to do some groups, so instead of have one columns for every top 25 actor, it was better to have one column which count the number of total actors in the top25 for each movie. Same thing was tried in producers and directors (doing groups of 10, 25 and 50, but there were not good results: trying to decrease the features associated with directors and producer companies does not work to improve performance and decrease overfitting). Finally, it was decided to use the top25 actors in one group (as just one feature) but the 60 producer companies and 60 directors count as 120 features, in addition it was used the other mentioned features. The final performance was around 0.48 of correlation in the validation data and around 0.8 in the training data, so there is still a lot of opportunities to improve the overfitting and performance.

About the classification model

It was made the assumption that features which help to predict revenue are similar to the features which contribute to classify according to rating. Therefore, it was used the same features than in the regression model plus 'runtime'.

At the beginning it was used cross validation to check which classification model could be better. It was compared: K-nearest neighbours, decision tree, logistic regression, linear discriminant analysis and random forest. As it could be expected k-nn was the worst (given the quantity of features) meanwhile random forest gave the better performance. All the models were compared using their default parameters.

After deciding the model, it was observed that there was a considerable overfitting: the performance in the training data was 1 for each of the three metrics (average precision, average recall and accuracy). Therefore, considering the overfitting and the imbalanced data problem (it was noticed in the first section and this is the reason why accuracy metric is not enough by itself to evaluate the performance of the model), it was decided to use class_weight='balanced_subsample' to compensate the imbalanced data and set a min_sample_leaf and max_features (features used in every split) to reduce the overfitting. Then, through a trial and error process to tune the parameters it was possible to improve the performance and decrease the overfitting.

	Final model (in validation data)	Final model (in training data)	Baseline (majority class)
Macro-avg-precision	0.71	0.50	0.34
Macro-avg-recall	0.74	0.82	0.5
Accuracy	0.74	0.74	0.69

How to continue improving?

To improve the two exposed model, it could be useful to make some work with the columns 'overview' and 'keywords'. Maybe make a first model (sentiment analysis) to classify those columns and then use that result as input in these models. Another method could be get the keywords most related with revenue and rating and count how many of those appear in the columns.