DAVA Project 2 Report

Daniel Rozen, drozen3

Code used from Project 1:

It is OK to use code and data that you created in Part I, but if you do please mention it and include that code in this submission.

I used Project 1 code to convert Runtime to a numeric value (in minutes)

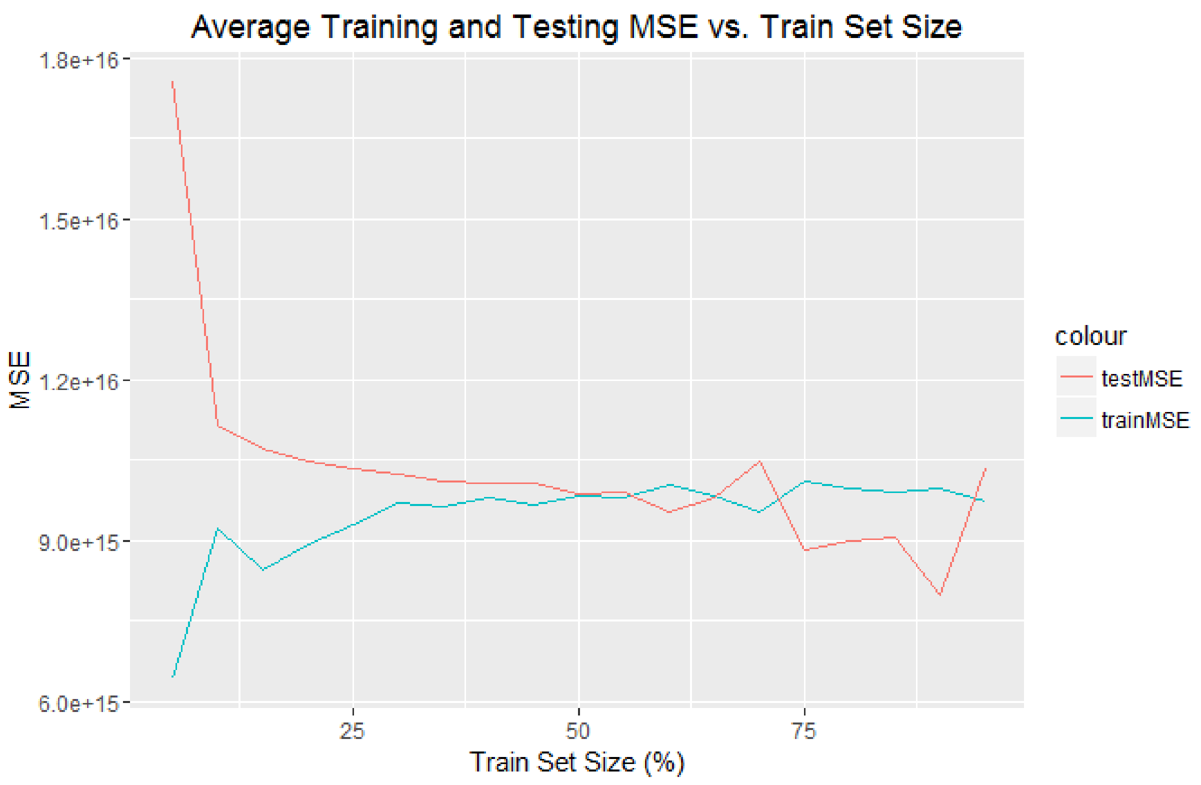
# From Project 1 Code: Convert Awards to a numeric column as a total # of awards and nominations

1. *Use linear regression to predict profit based on all available numeric variables. Graph the train and test MSE as a function of the train set size (averaged over 10 random data partitions as described above)?*

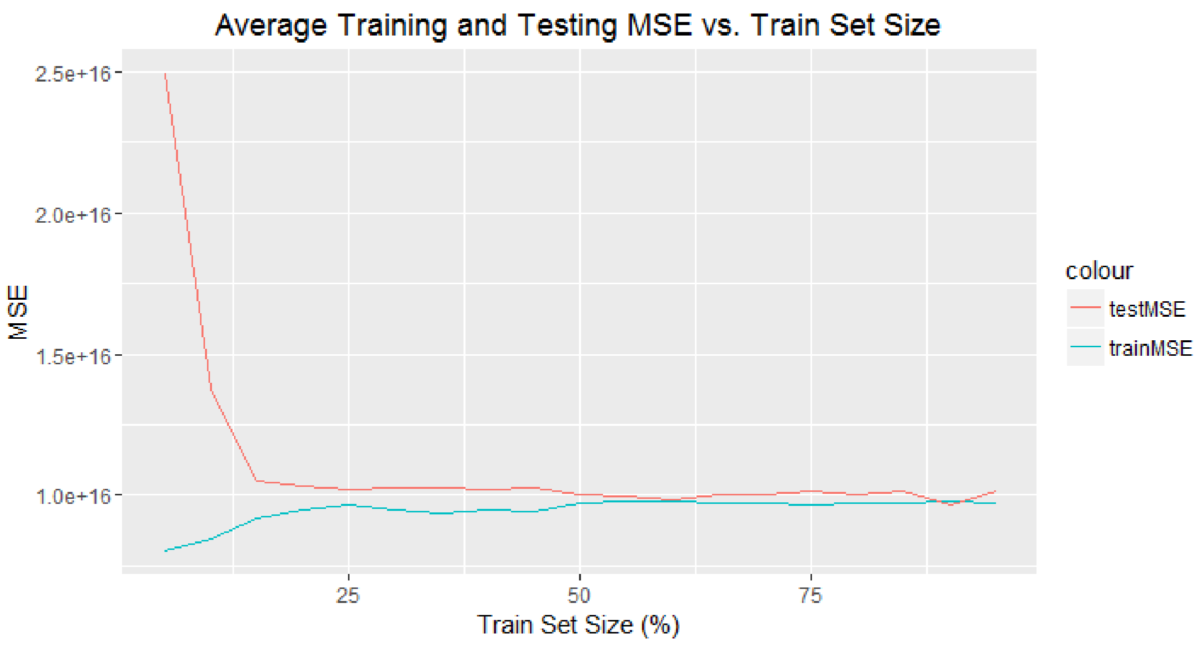
I kept in all the numeric variables that I felt made sense to use for linear regression.
Runtime + awardSum + Metascore + imdbRating + imdbVotes + tomatoMeter + tomatoRating + tomatoFresh + tomatoRotten + tomatoUserMeter + tomatoUserReviews + Budget

Since tomatoReviews = tomatoFresh + tomatoRotten, which is a derived column. I eliminated that column from the analysis.

Also since tomatoUserMeter and tomatoUserRating are highly correlated, I removed tomatoUserRating.



Averaging over 100 random data partitions we get the following smoother curve.



```
  trainSetSize     trainMSE      testMSE
1            5 8.039534e+15 2.496768e+16
```

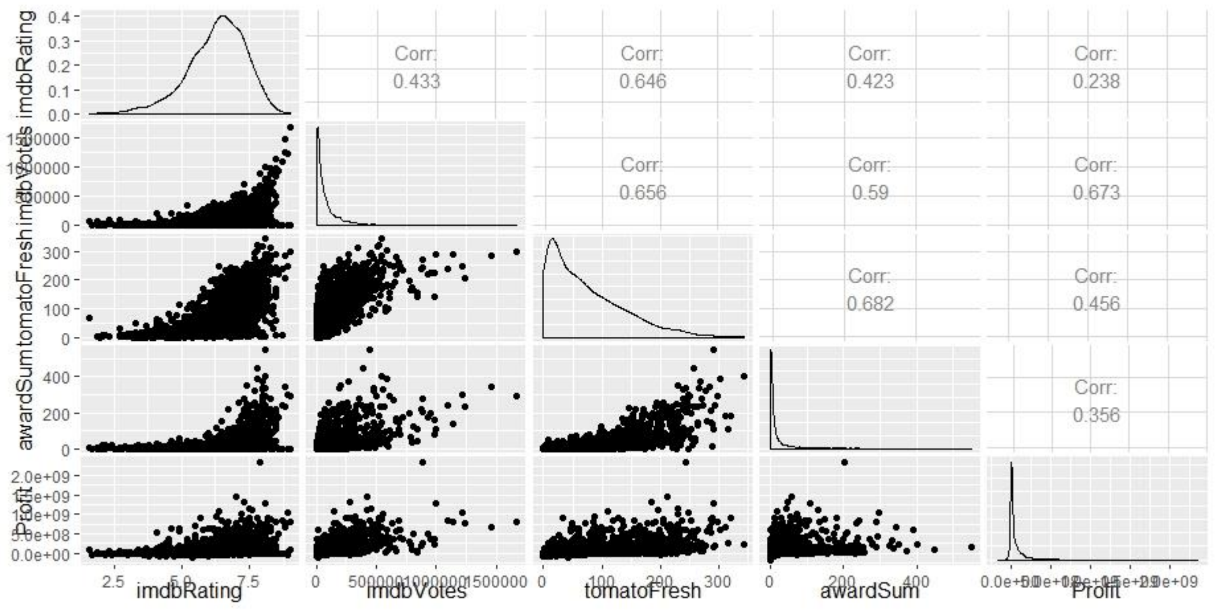
```
2          10 8.444581e+15 1.373990e+16
3          15 9.200654e+15 1.048341e+16
4          20 9.468301e+15 1.031235e+16
5          25 9.670488e+15 1.021803e+16
6          30 9.495577e+15 1.024507e+16
7          35 9.377200e+15 1.028868e+16
8          40 9.496291e+15 1.021311e+16
9          45 9.434765e+15 1.027005e+16
10          50 9.695225e+15 1.003948e+16
11          55 9.776149e+15 9.943176e+15
12          60 9.793006e+15 9.866561e+15
13          65 9.708156e+15 1.003894e+16
14          70 9.712323e+15 1.002749e+16
15          75 9.674353e+15 1.015216e+16
16          80 9.732602e+15 1.000886e+16
17          85 9.719682e+15 1.013549e+16
18          90 9.781492e+15 9.669042e+15
19          95 9.745892e+15 1.011614e+16
```

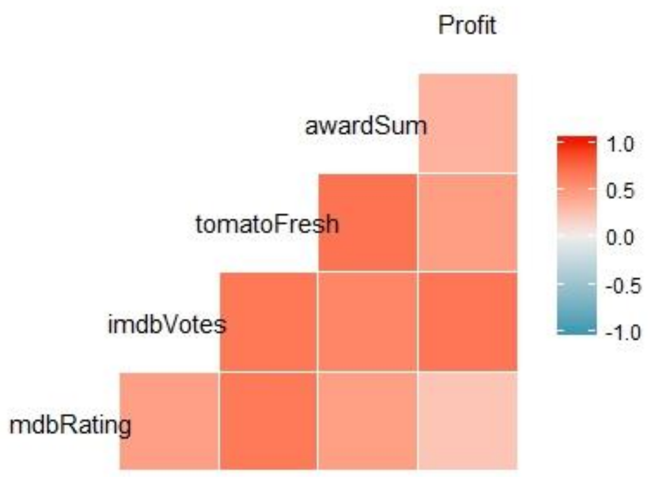
We see that testMSE converges to trainMSE at around 15% train set size. This seems to be sufficient to accurately predict the linear regression model. Also trainMSE increases slightly with increasing Train Set Size, possibly do to less overfitting.

2. *Try to improve the prediction quality in (1) as much as possible by adding feature transformations of the numeric variables. Explore both numeric transformations such as power transforms and non-numeric transformations of the numeric variables like binning (e.g., is_budget_greater_than_3M). Explain which transformations you used and why you chose them. Graph the train and test MSE as a function of the train set size (averaged over 10 random data partitions as described above)?*

Using code from project 1 we saw that month released tended to affect gross revenue, and perhaps profit.
Similarly we see that imbdVotes had a high strong positive correlations with Profit (.673) followed by tomatoFresh (.456), and awardSum (.356)

Therefore I attempted to include these in the model

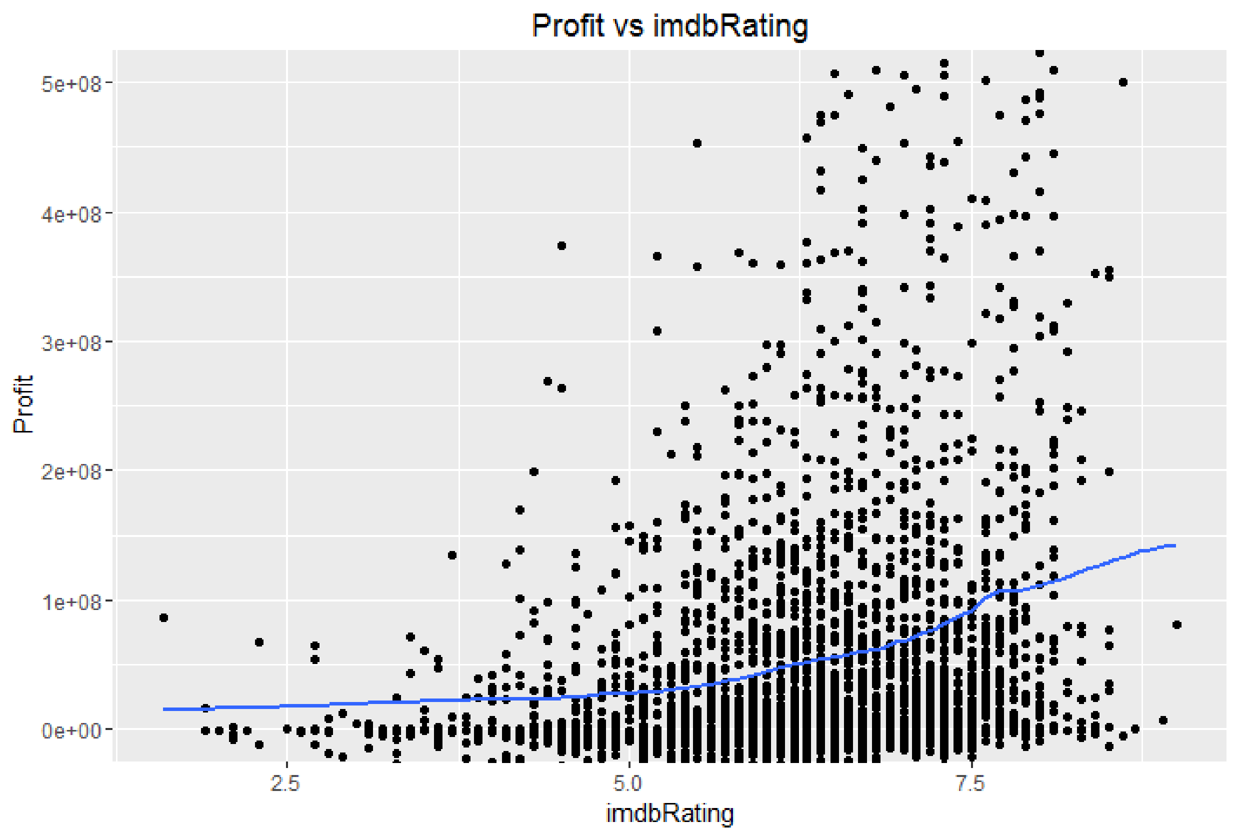




Results:

Using a linear model, and a squared and cubed model didn't yield better results.

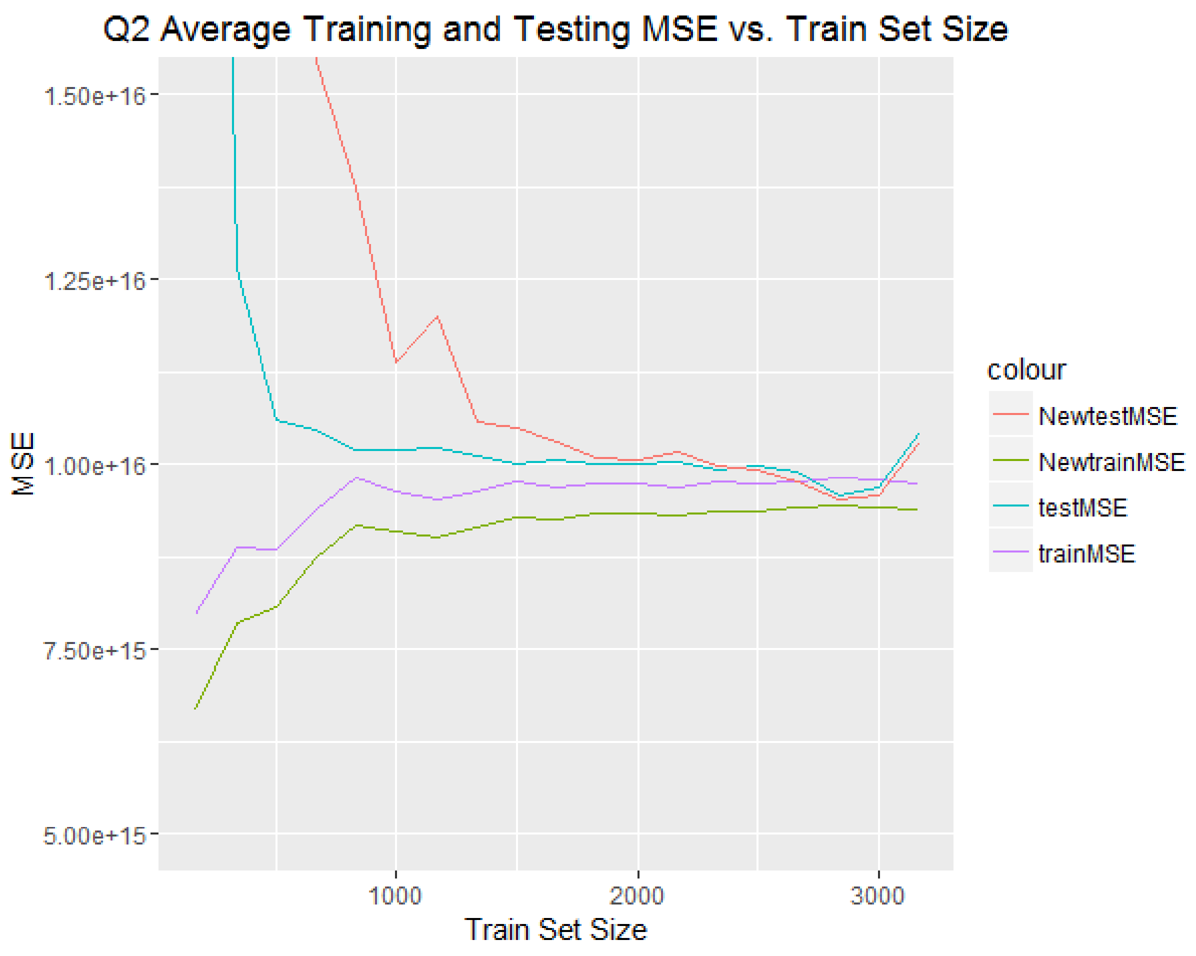
The following graph tends to suggest that imdbRating has a non-linear x^2 or x^3 relationship.



Therefore we accounted for this by adding the following power terms to the previous models in order to better account for a non-linear model.

M2 = lm(Profit~Runtime + awardSum + Metascore + imdbRating + imdbVotes + tomatoMeter + tomatoRating + tomatoFresh + tomatoRotten + tomatoUserMeter + Budget + I(awardSum^2) + I(awardSum^3) + I(imdbVotes^2) + I(imdbVotes^3) + I(tomatoFresh^2) + I(tomatoFresh^3) + I(imdbRating^2) + I(imdbRating^3), dfTrain1);

Averaging 100 times we get



```
> MSEAvDF
  trainSetSize     trainMSE      testMSE    trainMSE2     testMSE2
1          166 7.970591e+15 3.681479e+16 6.684223e+15 4.638629e+17
2          333 8.876111e+15 1.262917e+16 7.858402e+15 2.806443e+16
3          499 8.845720e+15 1.060512e+16 8.075358e+15 2.062962e+16
4          666 9.379174e+15 1.046160e+16 8.733877e+15 1.544702e+16
5          833 9.807979e+15 1.018626e+16 9.185361e+15 1.373207e+16
6          999 9.626584e+15 1.018223e+16 9.094659e+15 1.137104e+16
7         1166 9.508643e+15 1.023038e+16 9.021010e+15 1.199373e+16
8         1332 9.621643e+15 1.012454e+16 9.149570e+15 1.056793e+16
9         1499 9.760610e+15 1.001224e+16 9.284418e+15 1.047989e+16
10        1666 9.674450e+15 1.005391e+16 9.242252e+15 1.029372e+16
11        1832 9.723929e+15 9.998227e+15 9.320971e+15 1.009376e+16
12        1999 9.728445e+15 9.993953e+15 9.346922e+15 1.006865e+16
13        2165 9.694571e+15 1.003763e+16 9.297230e+15 1.016763e+16
14        2332 9.754661e+15 9.938164e+15 9.366973e+15 9.983085e+15
15        2499 9.732496e+15 9.971067e+15 9.350202e+15 9.936144e+15
16        2665 9.761670e+15 9.889224e+15 9.401970e+15 9.776355e+15
17        2832 9.810629e+15 9.582293e+15 9.441354e+15 9.529942e+15
```

```
18        2998 9.779440e+15 9.692982e+15 9.420247e+15 9.562918e+15
19        3165 9.733470e+15 1.040459e+16 9.381727e+15 1.028092e+16
```

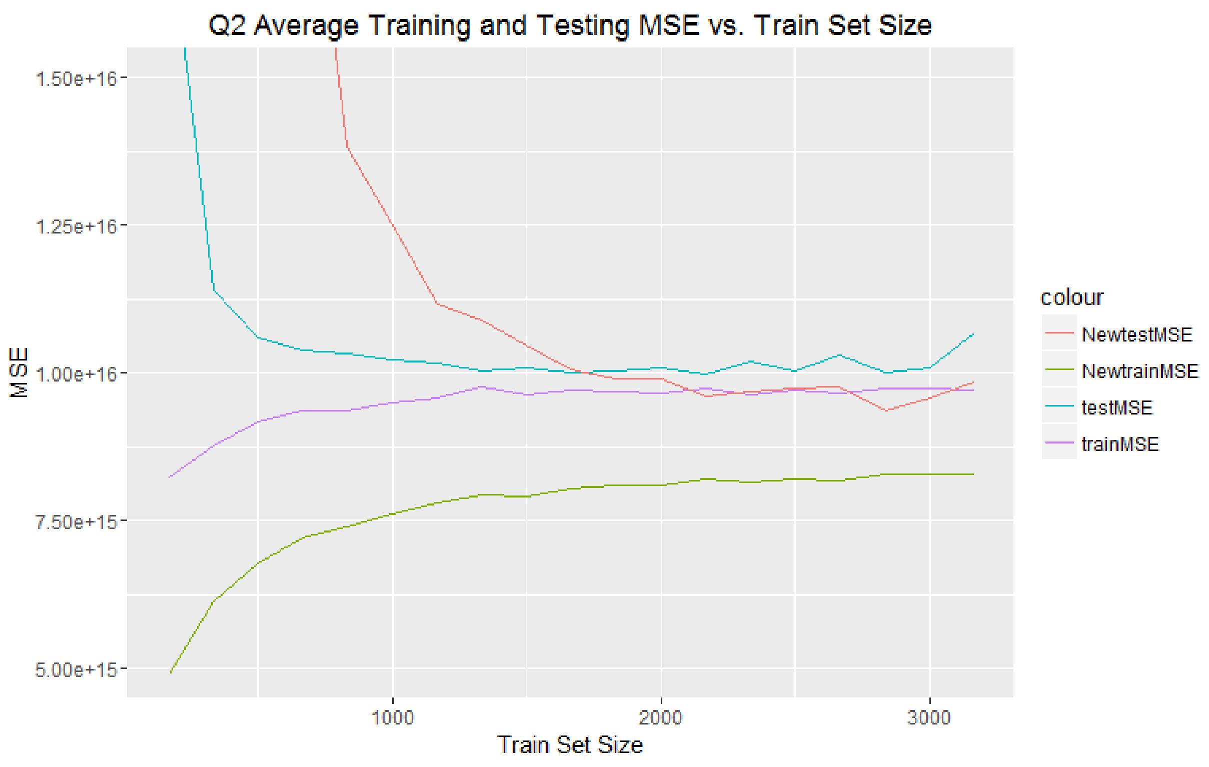
This achieves a small improvement. We see once we reach a trainSetSize of 2500 the new testMSE is lower than the old testMSE.

Additionally, it made sense that a movie with high awardSum, Metascore, imdbRating, and imdbVotes would be a very good movie and therefore achieve a high profit. Therefore, I added interaction terms with awardSum * Metascore * imdbRating * imdbVotes

Using the following model:

```
   M2 = lm(Profit~Runtime + awardSum * Metascore * imdbRating * imdbVotes + tomatoMeter
+ tomatoRating + tomatoFresh + tomatoRotten + tomatoUserMeter + Budget + I(awardSum^2)
+ I(awardSum^3) + I(imdbVotes^2) + I(imdbVotes^3) +  I(tomatoFresh^2) + I(tomatoFresh^3) +
I(imdbRating^2) + I(imdbRating^3), dfTrain1);
```

We achieved the results:

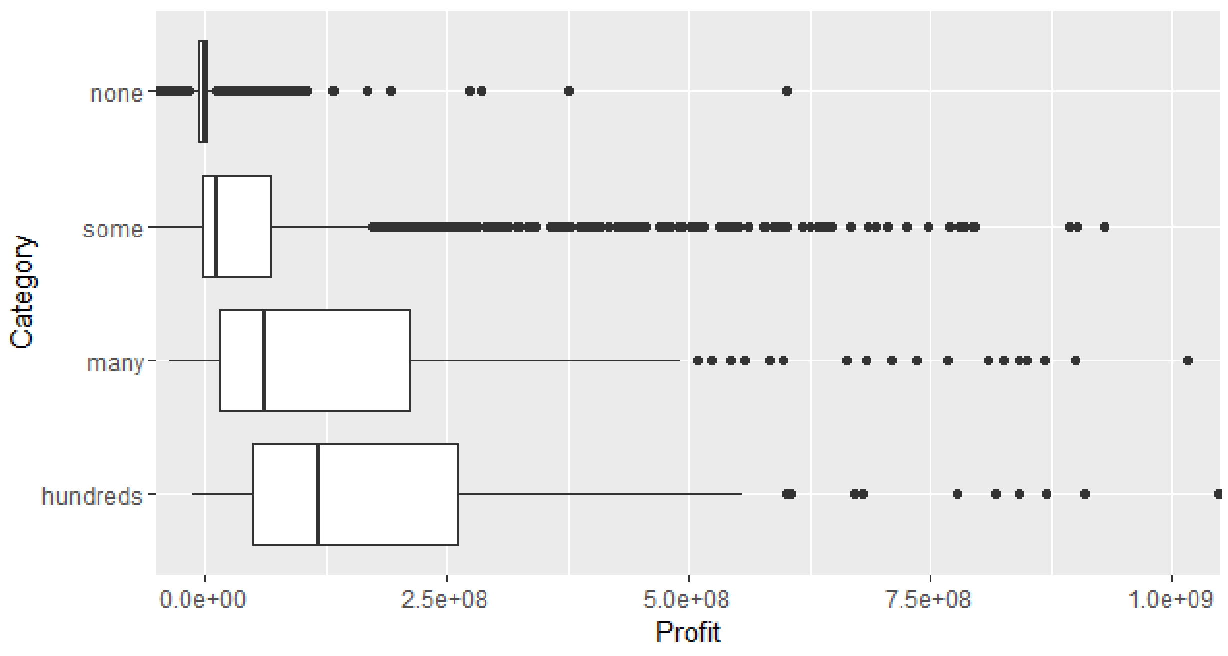


```
> MSEAvDF
  trainSetSize    trainMSE     testMSE   trainMSE2     testMSE2
```

```
1         166 8.225126e+15 1.778582e+16 4.902699e+15 3.246134e+17
2         333 8.777131e+15 1.141122e+16 6.137254e+15 7.908073e+16
3         499 9.161563e+15 1.058574e+16 6.773274e+15 3.015866e+16
4         666 9.371604e+15 1.036912e+16 7.201939e+15 1.988497e+16
5         833 9.373445e+15 1.032762e+16 7.388061e+15 1.383309e+16
6         999 9.495990e+15 1.022398e+16 7.604960e+15 1.253010e+16
7        1166 9.588803e+15 1.016710e+16 7.794577e+15 1.115059e+16
8        1332 9.759624e+15 1.003522e+16 7.944230e+15 1.090452e+16
9        1499 9.634142e+15 1.009504e+16 7.914301e+15 1.045984e+16
10       1666 9.705221e+15 1.001911e+16 8.046222e+15 1.006167e+16
11       1832 9.694999e+15 1.002391e+16 8.100783e+15 9.895332e+15
12       1999 9.655649e+15 1.008656e+16 8.090493e+15 9.887850e+15
13       2165 9.733730e+15 9.975882e+15 8.199203e+15 9.611055e+15
14       2332 9.638502e+15 1.020064e+16 8.140781e+15 9.686142e+15
15       2499 9.714232e+15 1.003726e+16 8.192280e+15 9.730487e+15
16       2665 9.655635e+15 1.030172e+16 8.186579e+15 9.765926e+15
17       2832 9.737810e+15 9.999057e+15 8.280160e+15 9.372787e+15
18       2998 9.735060e+15 1.009723e+16 8.278034e+15 9.588734e+15
19       3165 9.716870e+15 1.067407e+16 8.296845e+15 9.843038e+15
```

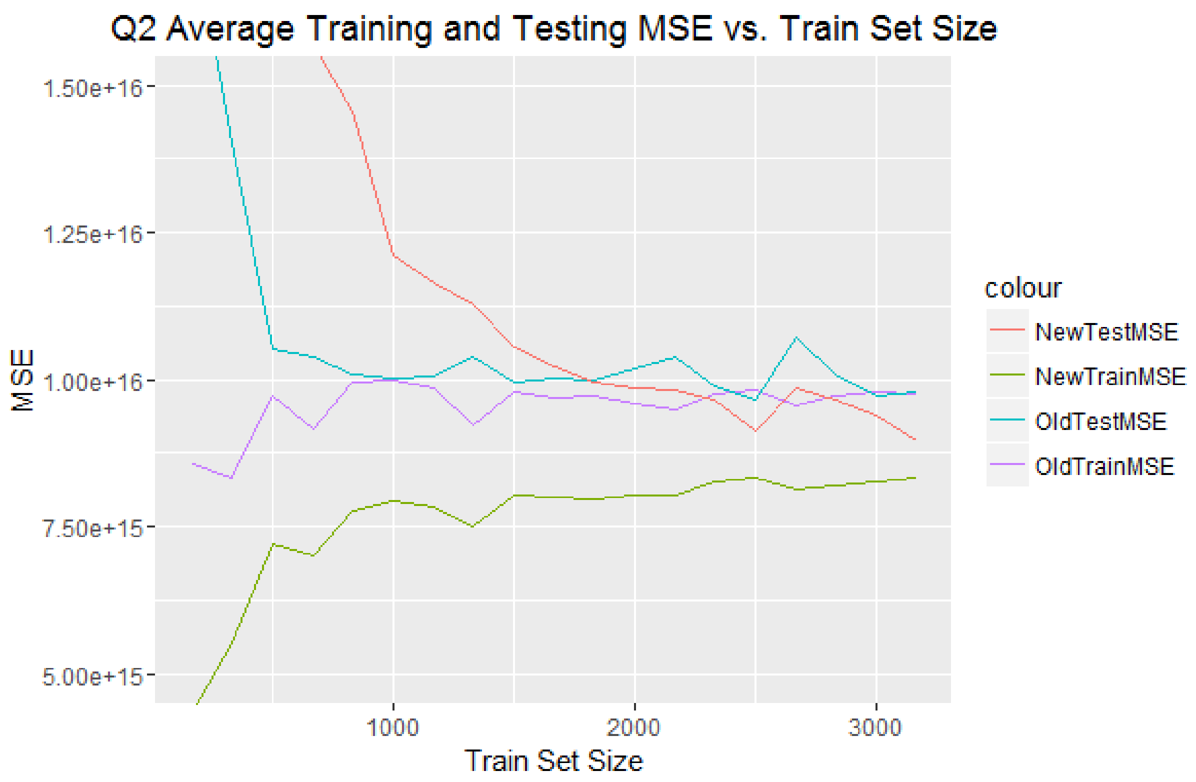
This improved the results somewhat significantly. Now newtestMSE surpasses old test MSE at a lower train set size = 1666

Also, we saw from Project 1 that binning awardsSums into 3 categories yielded significant results:
Using Binning of awardsSums into 4 categories similar to we see significant results with respect to profit. Therefore we decided to add this as a non-numeric transformation by binning in order to better fit the model to a non-linear model.



Adding awardCategory to the model we get

```
   M2 = lm(Profit~Runtime + awardSum * Metascore * imdbRating * imdbVotes + tomatoMeter
+ tomatoRating + tomatoFresh + tomatoRotten + tomatoUserMeter + Budget + I(awardSum^2)
+ I(awardSum^3) + I(imdbVotes^2) + I(imdbVotes^3) +  I(tomatoFresh^2) + I(tomatoFresh^3) +
I(imdbRating^2) + I(imdbRating^3) + awardCategory, dfTrain1);
```



Here again we see NewTestMSE surpasses the oldTestMSE at around trainSetSize of 1800.

The general trend of more complicated models starting off at a higher MSE and then lowering at greater Train Set Sizes makes sense since a more complicated model takes more data to generalize it.

3. *Write code that featurizes genre (can use code from Part-I), actors, directors, and other categorical variables. Explain how you encoded the variables into features.*

We encoded the categorical variables into features by first separating the commas into separate strings. We then used unnest() to make each separate feature entry a separate row with the correct corresponding Profit. We then used dcast to convert the features into binary vectors using one-hot encoding vector with 1s representing the presence of a feature and 0s the absence.

In order to handle the encoding properly, we separated by commas, replace spaces and dashes with _, replaced foreign characters "áéó", with "aeo", and also removed quotations marks and brackets.

From project 1 we saw that genre does influence gross, which may influence profit.

Therefore, we chose to encode Genre, Actors, Directors, Writer, Production, Rating, Language, and Country, as these variables made sense to me to that they could have an influence on profit.

Since there were too many binary vectors, we chose to only select a top proportion of each category k+1, using collapseCat() adapted from Damien Beneveste's function. This function would collapse all insignificant binary vectors whose proportion was less than the assigned threshold, and collapsed them all to one single collapsed_category binary vector. This was in order to avoid overfitting.

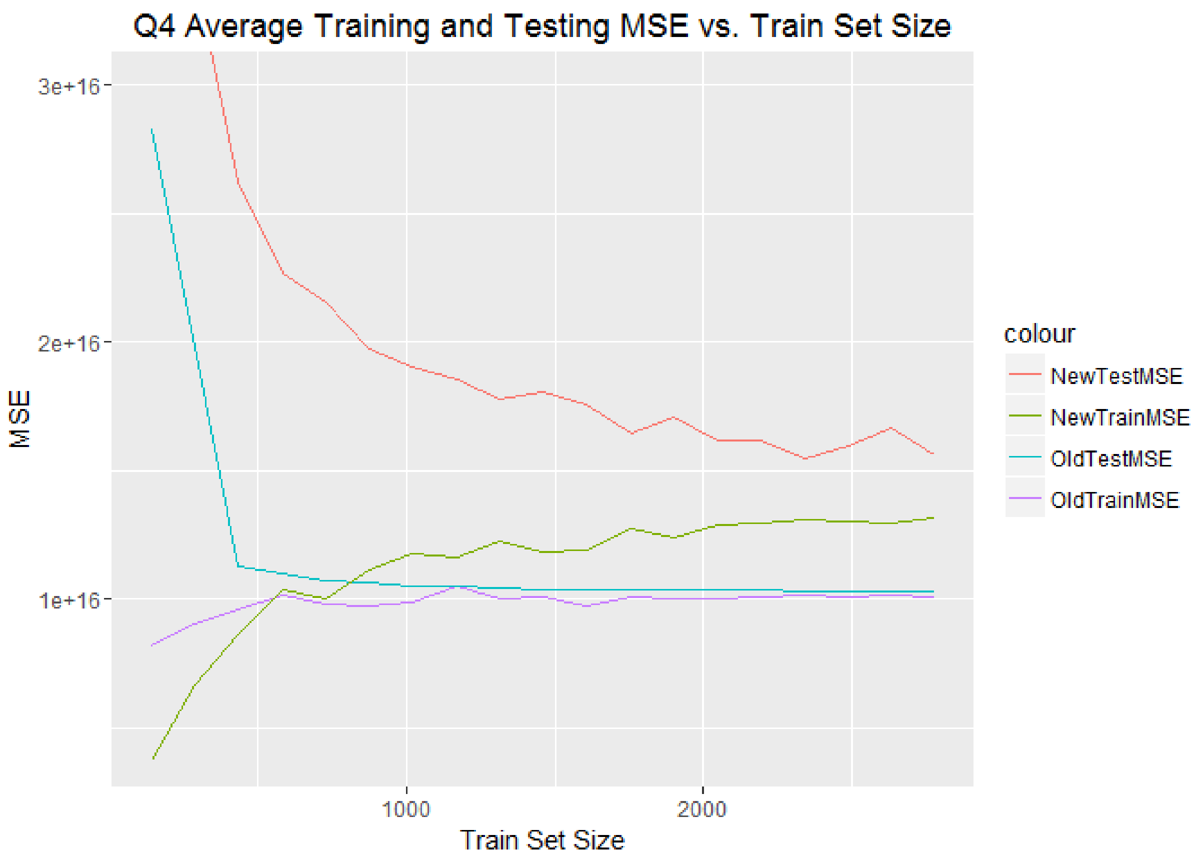
This greatly reduce the amount of binary vectors from 13976 to 138!

4. *Use linear regression to predict profit based on all available non-numeric variables (using the transformations in (3). Graph the train and test MSE as a function of the train set size (averaged over 10 random data partitions as described above)?*

Using the encoding from Q3 as binary vectors we get the following results Averaging over 20 times:

We chose these 8 because we felt they had the best predictor ability on Profit.

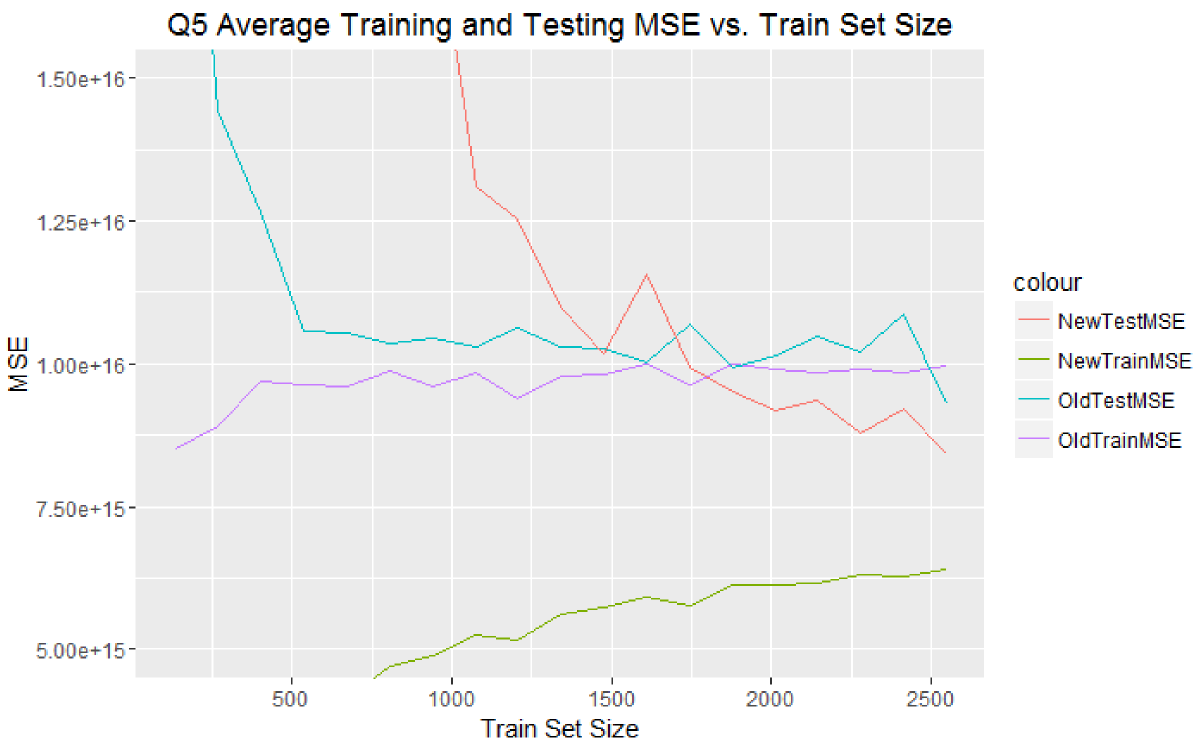
Here are the results:



We see the results are significantly worse than the original. Perhaps because of overfitting and too small train set sizes. Also perhaps due to the fact that these non-numeric variables are insufficient to predict on their own without the numeric variables and transformations from Q2.

5. *Try to improve the prediction quality in (1) as much as possible by using both numeric and nonnumeric variables as well as creating additional transformed features including interaction features (for example is_genre_comedy x is_budget_greater_than_3M). Explain which transformations you used and why you chose them. Graph the train and test MSE as a function of the train set size (averaged over 10 random data partitions as described above)?*

Using Q2's formula + Q4's formula averaged 50 times we obtain the following results:

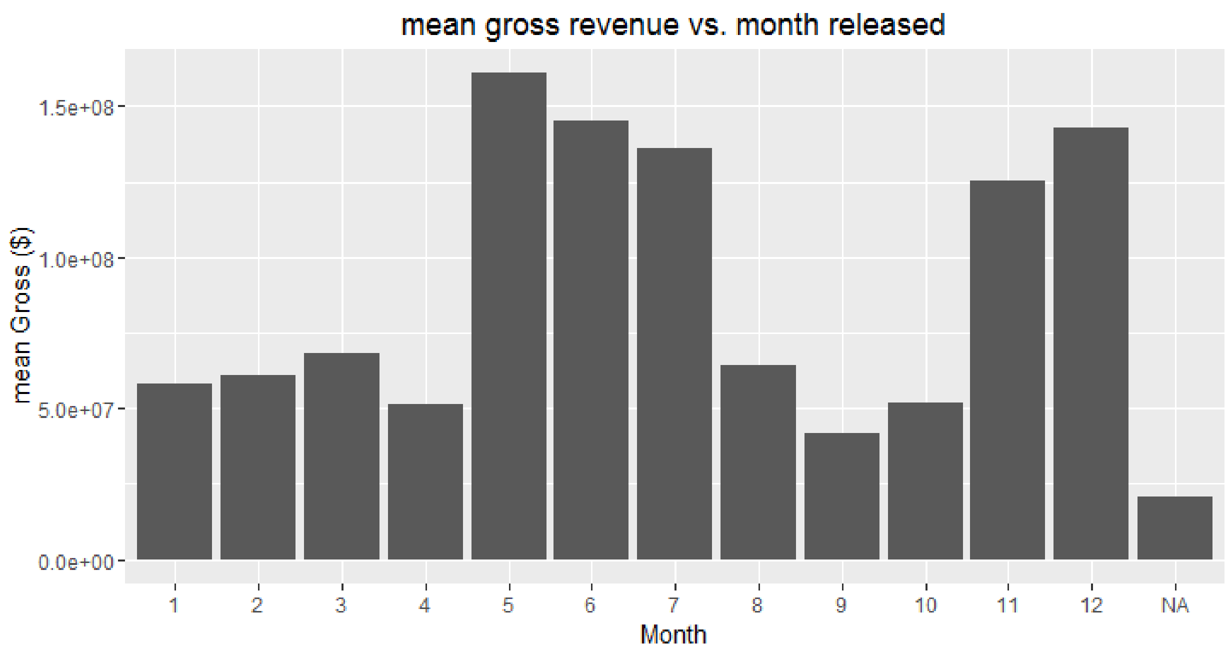


```
> MSEAvDF
   trainSetSize     trainMSE      testMSE    trainMSE2     testMSE2
1           134 8.514141e+15 2.393816e+16 5.410724e+09 8.340620e+22
2           268 8.896586e+15 1.443711e+16 1.530023e+15 3.448118e+17
3           402 9.701861e+15 1.264183e+16 2.878897e+15 1.212700e+17
4           536 9.618149e+15 1.055440e+16 3.491865e+15 4.140144e+16
5           670 9.602623e+15 1.053380e+16 4.119338e+15 2.770121e+16
6           804 9.864536e+15 1.034303e+16 4.713162e+15 2.917988e+16
7           939 9.607557e+15 1.043884e+16 4.884367e+15 1.834966e+16
8          1073 9.831825e+15 1.029878e+16 5.261157e+15 1.310547e+16
9          1207 9.400165e+15 1.064011e+16 5.167117e+15 1.253062e+16
10         1341 9.778419e+15 1.028692e+16 5.608406e+15 1.099550e+16
11         1475 9.810765e+15 1.026987e+16 5.730768e+15 1.018048e+16
12         1609 1.000052e+16 1.003374e+16 5.903800e+15 1.154652e+16
13         1743 9.626135e+15 1.069729e+16 5.777799e+15 9.928686e+15
14         1878 1.000565e+16 9.934009e+15 6.114536e+15 9.513676e+15
15         2012 9.913812e+15 1.014086e+16 6.125722e+15 9.179483e+15
16         2146 9.842407e+15 1.046153e+16 6.150633e+15 9.349920e+15
17         2280 9.910591e+15 1.021216e+16 6.293367e+15 8.794647e+15
18         2414 9.846921e+15 1.086678e+16 6.290579e+15 9.203654e+15
19         2548 9.974263e+15 9.283532e+15 6.397478e+15 8.414803e+15
```

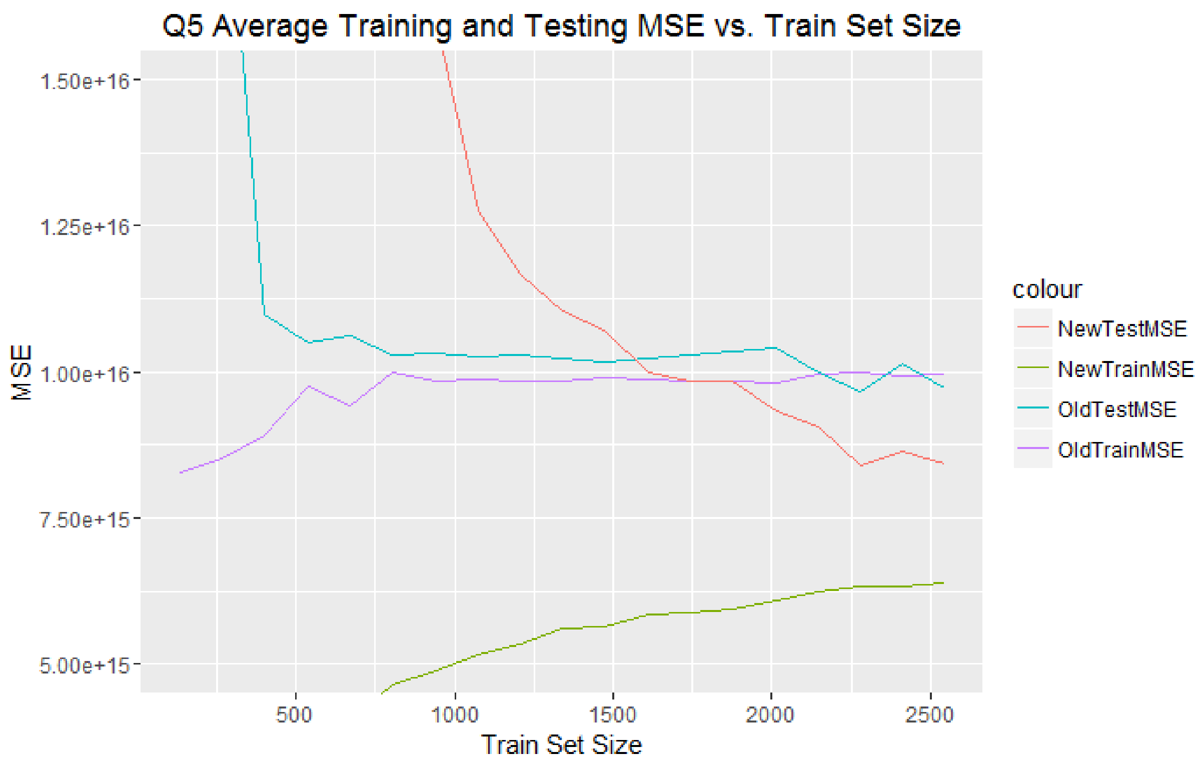
We see NewTestMSE surpasses OldTestMSE at around 1500 train set size.

This is a significant improvement to the results from Q4 and also an improvement from Q2.

We saw from Project 1, that released month appeared to have a strong influence on gross:



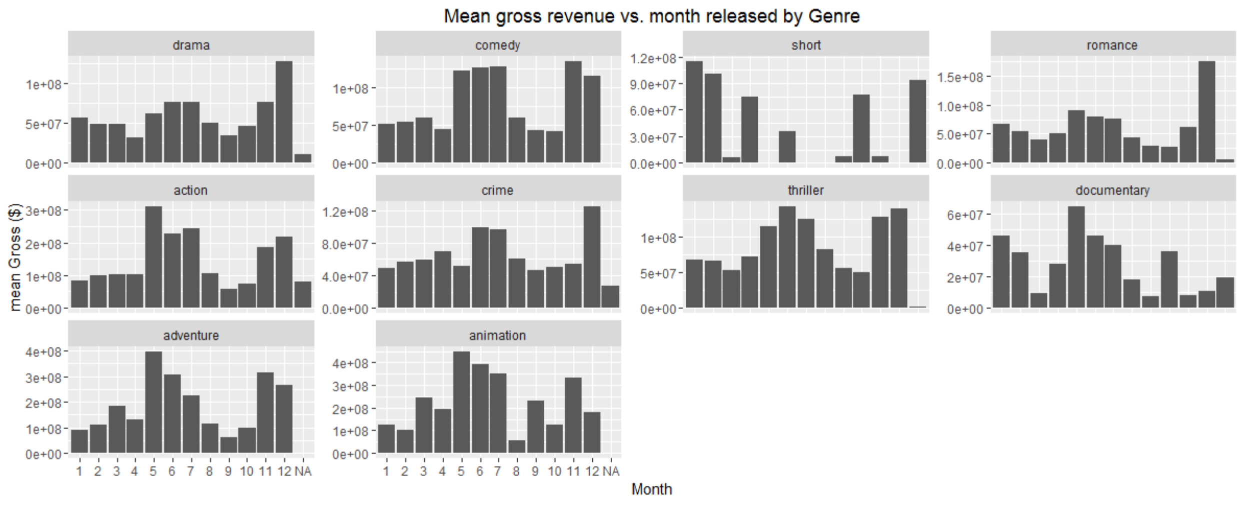
Therefore we converted them into binary categorical variables to use in our model with the following results (averaging 50 times):



```
  trainSetSize     trainMSE      testMSE    trainMSE2     testMSE2
1          134 8.274851e+15 1.703220e+16 0.000000e+00 1.473808e+23
2          268 8.525927e+15 1.930967e+16 1.295663e+15 2.173979e+17
3          402 8.918368e+15 1.097348e+16 2.456163e+15 1.356510e+17
4          536 9.748425e+15 1.050517e+16 3.671830e+15 4.840614e+16
5          670 9.414367e+15 1.062392e+16 4.027899e+15 2.182621e+16
6          804 9.997897e+15 1.029131e+16 4.645702e+15 1.920054e+16
7          939 9.851084e+15 1.031446e+16 4.900826e+15 1.601598e+16
8         1073 9.861889e+15 1.025269e+16 5.171126e+15 1.274143e+16
9         1207 9.850662e+15 1.028347e+16 5.336941e+15 1.165912e+16
10        1341 9.842891e+15 1.024007e+16 5.603173e+15 1.105209e+16
11        1475 9.905687e+15 1.016673e+16 5.643170e+15 1.070925e+16
12        1609 9.857850e+15 1.023697e+16 5.839583e+15 9.990383e+15
13        1743 9.847040e+15 1.028647e+16 5.872934e+15 9.851137e+15
14        1878 9.828335e+15 1.036094e+16 5.950133e+15 9.849034e+15
15        2012 9.825110e+15 1.042241e+16 6.081218e+15 9.325893e+15
16        2146 9.950305e+15 1.000243e+16 6.225397e+15 9.059822e+15
17        2280 1.000449e+16 9.658281e+15 6.319335e+15 8.394544e+15
18        2414 9.923750e+15 1.014922e+16 6.320094e+15 8.650860e+15
19        2548 9.952534e+15 9.723655e+15 6.373193e+15 8.434518e+15
```

This appeared to slightly improve the results, and gave more reliable results.

We also saw from Project 1, that the influence of ReleaseMonth on Gross was dependent on Genre



Therefore, we decided to include some interaction features off the relationships which were most apparent.

Eg. Comedy is highest in months 3,4,5, 11,12 = I(Comedy * (3,4,5, 11,12))

Drama is highest in month 11. Action, Adventure, and Documentary, is highest in month 5.

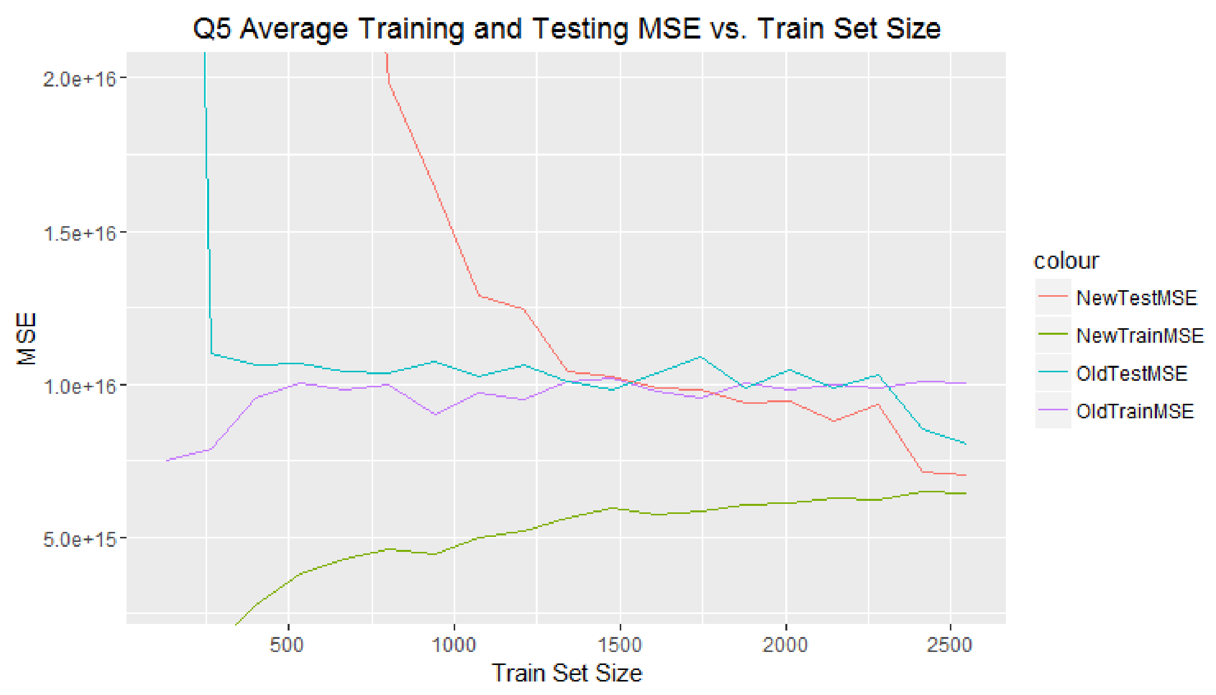
Crime is highest in month 11. Thriller is highest in months 6,10,11

Documentary is highest in 5

Romance is highest at 11

Animation is highest at 5 and 6.

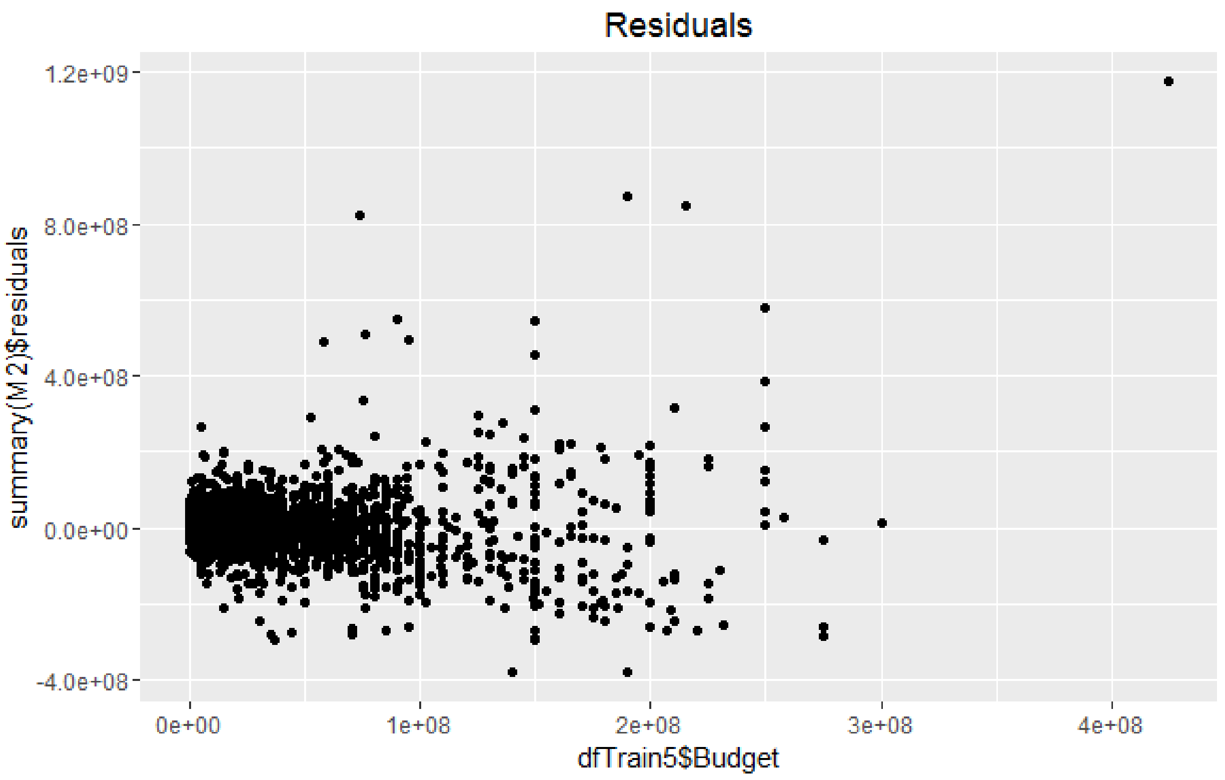
We captured all of the above in interaction features and achieved the following results:



```
> MSEAvDF
   trainSetSize     trainMSE      testMSE    trainMSE2      testMSE2
1           134 7.488099e+15 6.652830e+16 0.000000e+00 6.993110e+20
2           268 7.871605e+15 1.098508e+16 1.406285e+15 9.610378e+17
3           402 9.550010e+15 1.061882e+16 2.791047e+15 5.714788e+16
4           536 1.004477e+16 1.068288e+16 3.795942e+15 3.136941e+16
5           670 9.832101e+15 1.038495e+16 4.310154e+15 3.474755e+16
6           804 1.000312e+16 1.035350e+16 4.590125e+15 1.984136e+16
7           939 9.029915e+15 1.072496e+16 4.442826e+15 1.643971e+16
8          1073 9.691617e+15 1.025533e+16 4.993663e+15 1.289055e+16
9          1207 9.492780e+15 1.062828e+16 5.219745e+15 1.247239e+16
10         1341 1.006513e+16 1.010725e+16 5.631746e+15 1.040925e+16
11         1475 1.019042e+16 9.814758e+15 5.954319e+15 1.022731e+16
12         1609 9.746549e+15 1.036932e+16 5.759875e+15 9.886056e+15
13         1743 9.525036e+15 1.088962e+16 5.858559e+15 9.840149e+15
14         1878 1.000812e+16 9.883190e+15 6.065841e+15 9.413148e+15
15         2012 9.796415e+15 1.048550e+16 6.104194e+15 9.429567e+15
16         2146 9.982430e+15 9.847423e+15 6.277889e+15 8.805561e+15
17         2280 9.896540e+15 1.029124e+16 6.225519e+15 9.346543e+15
18         2414 1.010525e+16 8.511698e+15 6.480899e+15 7.153922e+15
19         2548 1.003854e+16 8.053591e+15 6.446766e+15 7.026580e+15
```

We see that these interaction terms greatly improved the results of the fit. As we say the lowest testMSE's so far of around 7e+15. This makes sense based on the above observations of high gross for the correct genre in the correct month.

Plotting the residuals vs. budget showed that the model is mainly unbiased and homoscedastic, which suggests that the model did a pretty good job. Of course improvements can be made, as the right side of the graph looks slightly more heteroscedastic, but unfortunately I ran out of time.



In summary, we can conclude from the above that the combination of numerical variables, feature transformations and featurizing non-numeric variables achieved the best results.

Also, the more advanced the model, the more overfitting occurs at lower training data sizes, but the higher the accuracy will be as we move towards higher training data set sizes.