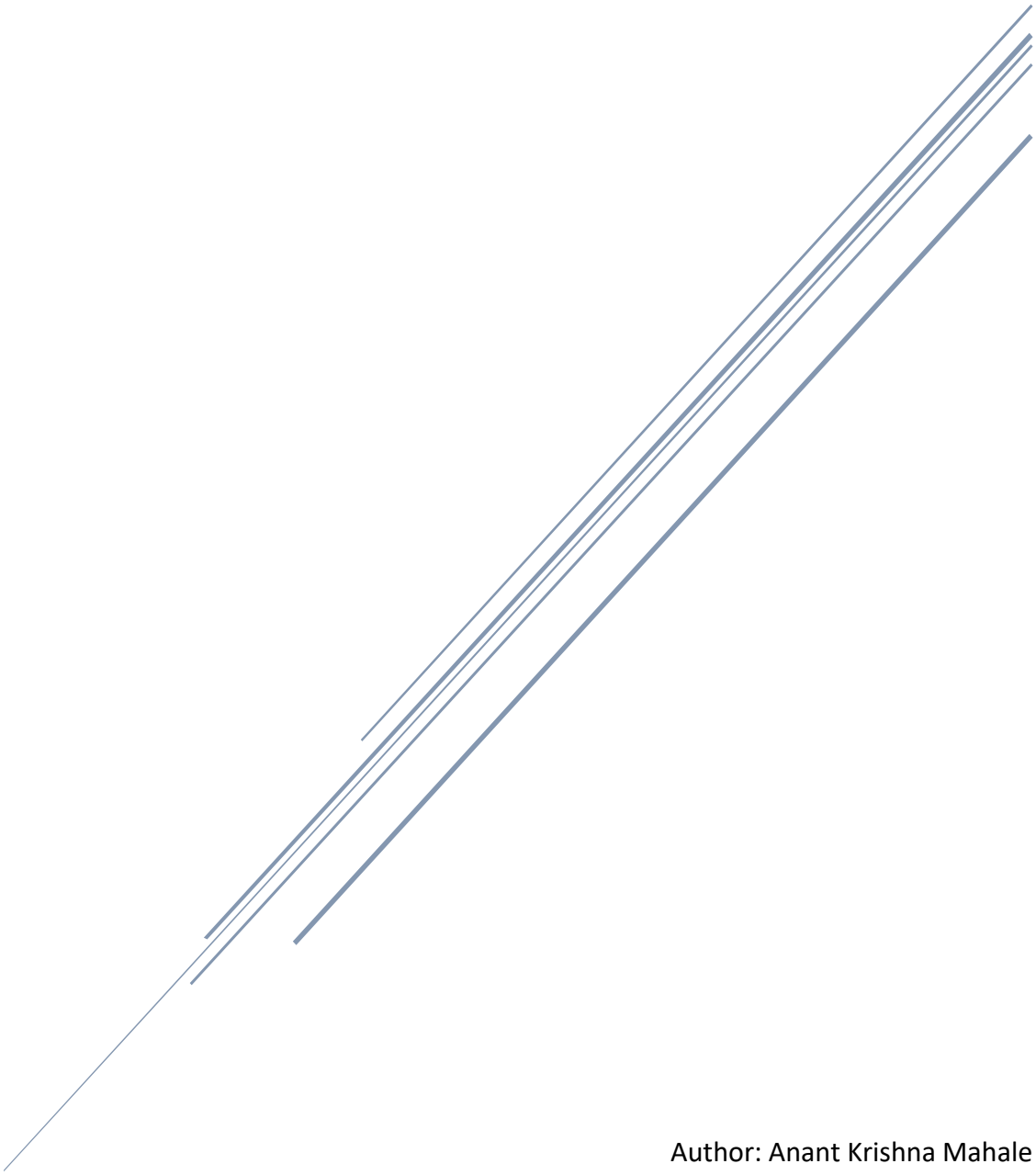
PREDICTING THE MOVIE REVENE AND RATING

Assignment 3 | COMP9321



Author: Anant Krishna Mahale
zID: z5277610

There are 2 parts in this document. First one talks about the Revenue prediction using Linear regression and seconds talks about the Movie rating Classification using Random Forest Classifier.

Regression

Understanding the structure of the data:

The dataset has 19 columns with following datatype:

1	movie_id	int64
2	cast	object
3	crew	object
4	budget	int64
5	genres	object
6	homepage	object
7	keywords	object
8	original_language	object
9	original_title	object
10	overview	object
11	production_companies	object
12	production_countries	object
13	release_date	object
14	revenue	int64
15	runtime	float64
16	spoken_languages	object
17	status	object
18	tagline	object
19	rating	int64

Since I'm not supposed to use the rating for the regression, dropping the column, straightaway.

On checking further, the columns: cast, crew, genres, keywords, production_companies, production_countries, spoken_languages have data in JSON format.

Since Machine Learning models need data in the numerical format, the relevant data need t o be extracted and processed.

Data Pre-processing and Cleaning:
Before processing any data in JSON, I wanted to remove the rows with any missing numeric al data. I used string function isdigit ().

Cast is one of the columns which has JSON data. Taking close look at one of the cells, I found that ‘name:’ contains important data hence decided to remove other data.

Pasting one instance of one such cell.
"cast_id": 242, "character": "Jake Sully", "credit_id": "5602a8a7c3a3685532001c9a", "gender": 2, "id": 65731, "name": "**Sam Worthington**", "order": 0

Once I figured this out, I applied same technique to rest of the columns which had JSON data.

Looking at the status column data, every movie has status as released. Hence decided to drop this column.

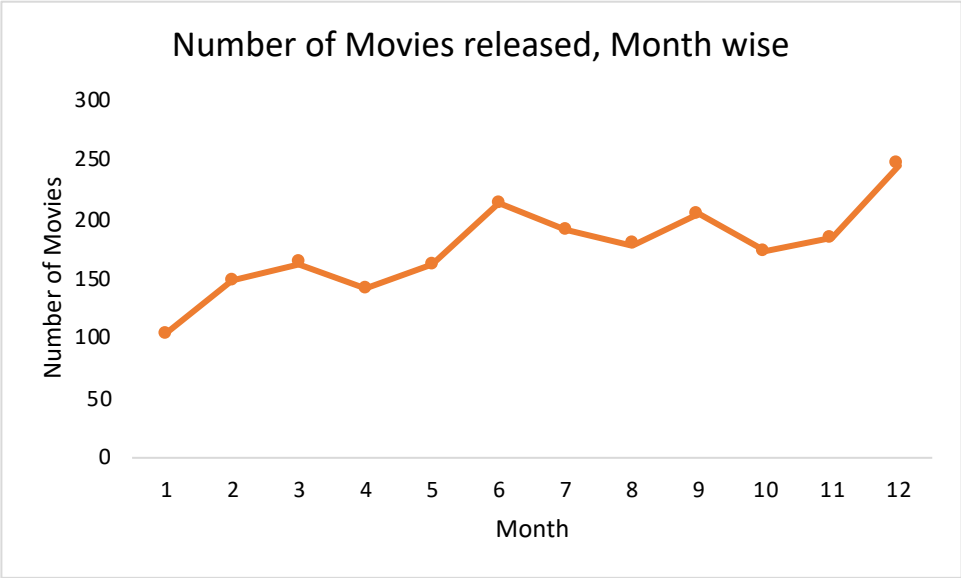
Also, I decided to drop Homepage, tagline and overview as they are subjective to individual movie.

Feature Selection:

Budget: Budget plays very important feature in the movie making as it decides various other factors of the movie.
To check that I checked the correlation between budget and revenue. I found that there is a 0.683 co-relation between movie and budget alone. Hence it plays very important role.

Release date: This is the very important feature. For example, if the movie’s target audience is children, then the movie should be realised in the holiday season. If it is released during exam time, children might not be able to watch the movie hence it might affect the revenue.

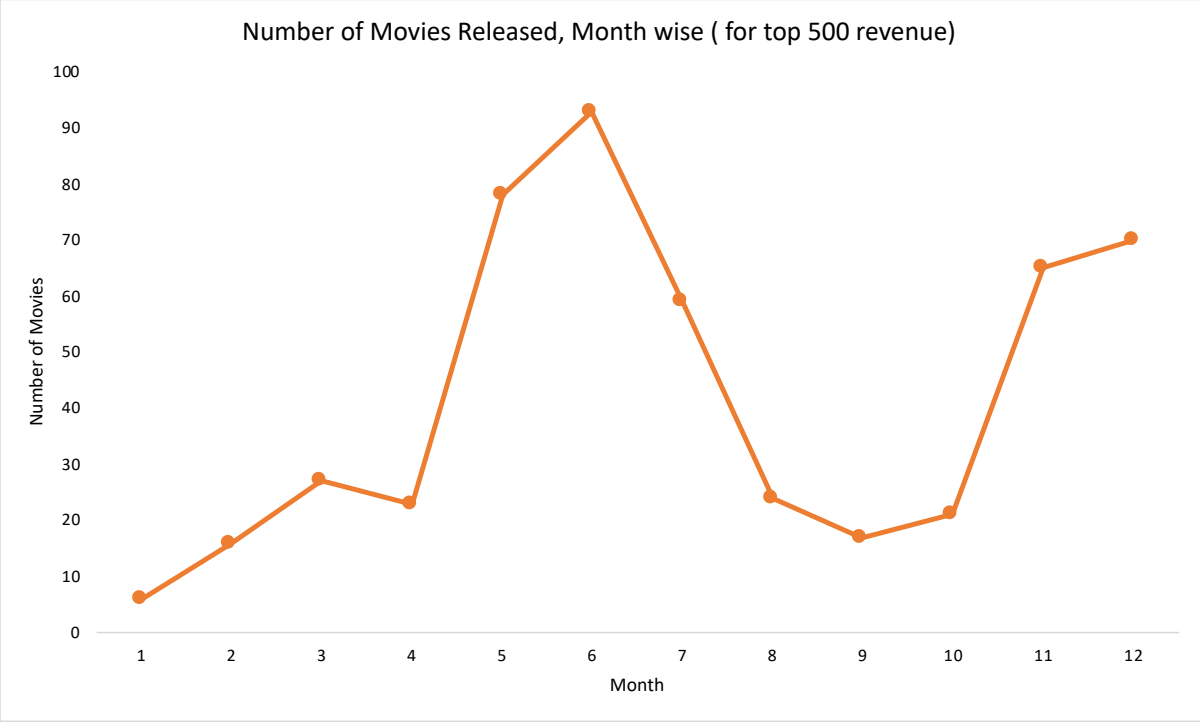
I wanted to check if there is a trend for particular month, hence plotted a graph month vs number of movies released.



I do see number increases in the month of May, June, November and December.

However, I even wanted to check the data for top 500 movies which had highest revenue.

To check that I made a graph out of top 500 revenue movies and found that movies most of them were released during April, May, November and December which is a holiday season.



Hence to make use of the release date, I decided to put 1 for above columns and 0(Zero) for rest of the columns.

Cast: The actors also play a very important role while determining the revenue of the movie. The biggest problem here was that, the actors count was in thousands.

Similarly, the problem was applicable to most of the JSON encoded columns.

Revenue & Budget:
The biggest problem that has to be addressed with respect to budget is inflation. Since were not allowed to use any external library, decided to go ahead with as it is values.

Also, I found that Revenue and Budget has values which are in 3 digits. This is certainly not true. Assumed it is a human error and decided to remove them.

Handling the JSON data:
Initially I converted all the JSON data into individual columns for example, all the actors will be converted to column and mark 1 if the actor is present in the movie or 0(zero) otherwise. The problem was doing this was taking a lot of time. Since the program had to execute within 3 mins, this process was not efficient.

From machine learning assignment, I learnt about the sklearn library's CountVectorizer.

I limited list obtained while parsing JSON data to 500 (due to computation time) and applied CountVectoizer on each column('cast','crew','genres','keywords','production_companies','production_countries','spoken_languages'). Upon doing this, I merged the data back to the original data-frame. Hence I could make use of most of the data.

For curiosity I used **RFE** in sklearn to rank the features. Found below features ranked in increasing order.

```
'budget', 'runtime', 'Michael Papajohn', 'Tom Felton', 'Ian McKellen', 'Imelda Staunton', 'Bob Bergen', 'Terrence oward', 'Sacha Baron Cohen', 'Steve Carell', 'Tom Cruise', 'Mona Marshall', 'Jane Lynch', 'Ray Stevenson', 'Matt Walsh', 'George Lopez', 'Rowan Atkinson', 'Andrew Menzies', 'Mauro Fiore', 'Ken Fischer', 'Ian Fleming', 'Andrew Stanton', 'Trudy Ramirez', 'Lee Orloff', 'James Vanderbilt', 'Phil Tippett', 'Dana Goldberg', 'Barrie M Osborne', 'Julie D Antoni', 'Don Burgess', 'Chris Sanders', 'Simon Otto', 'Shane Prigmore', 'Joy Zapata', 'Tom Reta', 'Pete Cavaciuti', 'Michael Andrews', 'Greg Powell', 'Avy Kaufman', 'Jon Title', 'William R DeanRichard Francis BruceJames Ashwill', 'Ruth Lambert', 'Dean Cundey'.
```

Training and Testing Data:

Initially I used only numerical values to train the model. I got r2 score around 0.4. With above processed data, we can see that the performance has increased.

To train the model, I split the dataset into 70 and 30 ratio. Here is the performance in different regressors:

Algorithm	MSE	r^2 Score	Pearson Co-efficient	p-value
Linear Regression	128756125.332	0.5618	0.7691	4.125589808144647e-136
RandomForestRegressor	128465703.077	0.5638	0.7530	2.276000076637452e-127
KNeighborsRegressor	153295562.117	0.3789	0.6690	7.30686268162071e-91

Though Linear regression and Random Forest Regressor are performing in same lines, I wanted to make use of Random-Forest regressor as the dataset has large number of columns. Due to time limitations, I chose Linear Regression.

Validating the model with Validation.csv

When I used validation.csv to validate the model, I got following results.

Algorithm	MSE	r^2 Score	Pearson Co-efficient	p-value
Linear Regression	99697267.0755	-0.2284	0.3189	6.654003365131483e-11
RandomForestRegressor	78336214.0262	0.2415	0.5392	1.4696521448207196e-31
KNeighborsRegressor	98350831.4383	-0.1954	-0.0661	0.18689077945239124

I was shocked to see the results, took some to realise the cause. Going back to basics, the budget had major influence on revenue in the training data. Hence, I decided to check the co-efficient between budget and revenue in the validation dataset and found below results:

	Pearson Co-efficient	p-value
Train Data	0.6820570173582146	6.640893726560272e-287
Test Data	-0.0094740785779431	0.8501768148740113

Since, there is huge difference between Pearson co-efficient of budget and revenue in train and validation set, the Pearson co-efficient of y_validate and y_predicted is very small.

Future Scope:
Given no time restriction on running program and computation power, we can run feature reduction techniques and have only those columns which significantly influence the model.

----------**********----------

Classification

This part of the document focuses on predicting Movies Rating using Classification Techniques.

Data Pre-processing and Cleaning:
There is restriction on Revenue for classification model. Hence dropping the column straight away.

Before processing any data in JSON, I wanted to remove the rows with any missing numeric al data. I used string function isdigit ().

I checked the number of classes in the training dataset. I found there are 3 different classes [1, 2, 3].

Upon checking the distribution, I found below results.

```
print('Original dataset shape %s' % Counter(df_raw_train['rating']))
```

```
Original dataset shape Counter({3: 1361, 2: 735, 1: 1})
```

There is only one entry with **rating 1**, it won't be sufficient to train the model hence decided to delete it.

The class 3 has ~50% more data than rating 2. There are 2 choices, either to under sample ra ting 3 or oversample the rows with rating 2. Since sklearn do not have inbuilt function to ov ersample like imbalance-learn library, I decided to under-sample the rating 3 class.

Cast is one of the columns which has JSON data. Taking close look at one of the cells, I found that 'name:' contains important data hence decided to remove other data.

Pasting one instance of one such cell.
"cast_id": 242, "character": "Jake Sully", "credit_id": "5602a8a7c3a3685532001c9a", "gende r": 2, "id": 65731, "name": "**Sam Worthington**", "order": 0

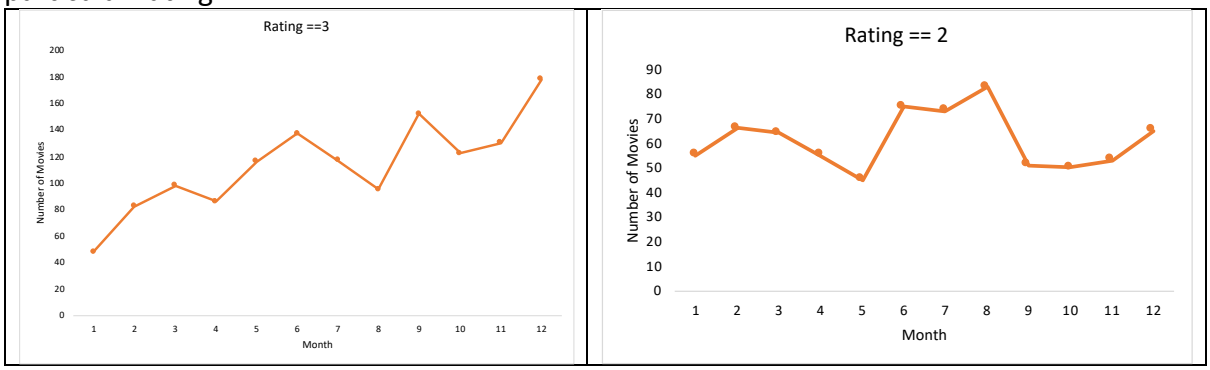
Once I figured this out, I applied same technique to rest of the columns which had JSON dat a.

Also, I decided to drop Homepage, tagline and overview as they are subjective to individual movie.

Feature Selection:

Budget: Initially thought that budget has impact on revenue but not really the effect on the rating. To verify that, I checked the coefficient between rating and budget and found it very low. Hence decided to drop it. Once, other features were decided and model was trained, I decided to include, budget to see if it improves the performance. Surprisingly, it improved around by 5% hence decided to retain it.

Release Date: I checked the month vs rating trend; I did not see a particular month for particular rating.



Dropped this column initially, but again this feature was influencing the performance of the model. It boosted the precision by ~5%

JSON Columns: I converted all the values (names) in JSON columns to individual rows and used to 1 or 0 to mark their presence in particular movie. However, the problem here is that the list is very big [in thousands]. Using all the columns would consume more time. Also running column reduction algorithm would take time.

From machine learning assignment, I learnt about the sklearn library's CountVectorizer.
I limited list obtained while parsing JSON data to 500 (due to computation time) and applied CountVectoizer on each
column('cast','crew','genres','keywords','production_companies','production_countries','spoken_languages'). Upon doing this, I merged the data back to the original data-frame. Hence, I could make use of most of the data.

Training and Testing Data:
With above methods and processed data, I decided to train the model and check the performance with same dataset by splitting the dataset in 70:30 ratio.

Here is table describing the performance of the model.

Classifier	Accuracy Score	Avg. Precision	Avg. Recall	f1-score
KNeighbors	0.68	0.68	0.68	0.68
SVC	0.65	0.65	0.65	0.65
DecisionTree	0.69	0.69	0.69	0.69
RandomForest	0.69	0.70	0.68	0.68

We can see that Random Forest and Decision Tree scores are better compared to KNN and SVC Classifiers. Since the dimension of the data is huge, trees utilize subset of data to make the decisions.
Specifically, random forest utilizes bagging and boosting techniques to makes decisions. This decreases the variance without increasing the bias. Hence, I chose Random Forest Classifier over the others.

Validating the dataset with Validation.csv
Upon validating the model with Validation Dataset, I got following results:

Classifier	Accuracy Score	Avg. Precision	Avg. Recall	f1-score
KNeighbors	0.59	0.51	0.51	0.51
SVC	0.47	0.48	0.47	0.45
DecisionTree	0.68	0.63	0.64	0.64
RandomForest	0.72	0.68	0.69	0.68

We can see that performance metric obtained while predicting the values in validation set are in line with Trained Model performance.

Future Scope:
Given no time restriction on running program and computation power, we can run feature reduction techniques and have only those columns which significantly influence the model.