IME672A - Data Mining & Knowledge Discovery

TMDB Box Office Revenue Prediction

Course Project - Group 21

Members: Archit Awasthi, Ashish P Murali, Ayush Gupta, Piyush, Saksham Gupta

Problem Statement

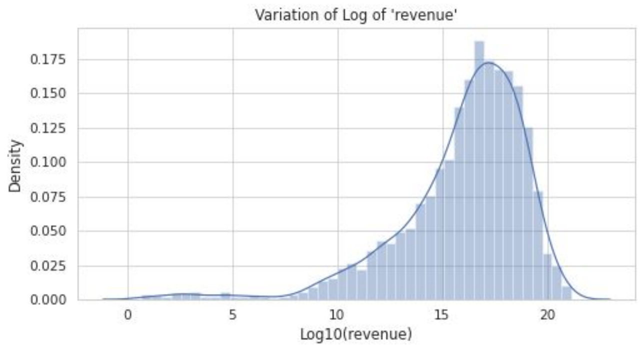
The goal is to predict the total revenue a movie will make at the box office worldwide. As the net income is a continuous variable, it is a Regression problem. The task is to develop the model using various Machine Learning techniques. From a business point of view, the film studios' main interests and related stakeholders predict the revenue a new movie can generate based on a few given input attributes before it is released.

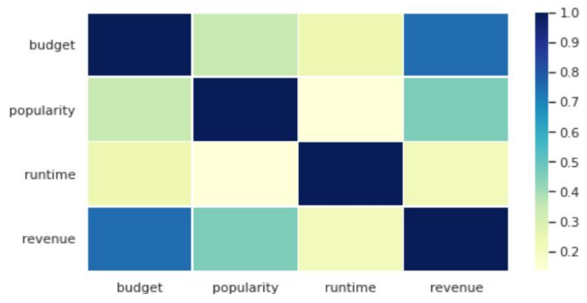
Data Analysis & Visualization

We use metadata on 7,398 movies with 23 features per film. The data comes from the Kaggle Competition available here. Apart from the initial data, we also combine data about some other movies available The complete list of features, along with a brief description, is given below:

Description of data columns

Data Description id	Integer unique id of each movie
belongs_to_collection	The TMDB Id, Name, Movie Poster and Backdrop URL of a movie in JSON format
budget	The budget of a movie in dollars. 0 values mean unknown
genres	All the Genres Name & TMDB Id in JSON Format
homepage	The official homepage URL of a movie
imdb_id	IMDB id of a movie (string)
original_language	Two-digit code of the original language in the movie
original title	The original title of the movie. It may differ from the title if not in English
overview	Brief description of the movie
popularity	The popularity of the movie in 'float'
poster_path	Poster path of a movie
production companies	Production company names and TMDB id in JSON format of a movie
production countries	Two-digit code and the full name of the country in JSON format
release_date	Release date of a movie in mm/dd/yy format
runtime	Total runtime of a movie in minutes (Integer)
spoken_languages	Two-digit code and the full name of the spoken language
status	Is the movie released or rumoured
tagline	The tagline of a movie
title	English title of a movie
Keywords	TMDB Id and name of all the keywords in JSON format
cast	Cast TMDB id, name, character name, gender (1 = Female, 2 = Male) in JSON format
crew	Name, TMDB id, profile path of various kind of crew members job like Director, etc
revenue	The total revenue earned by a movie in dollars

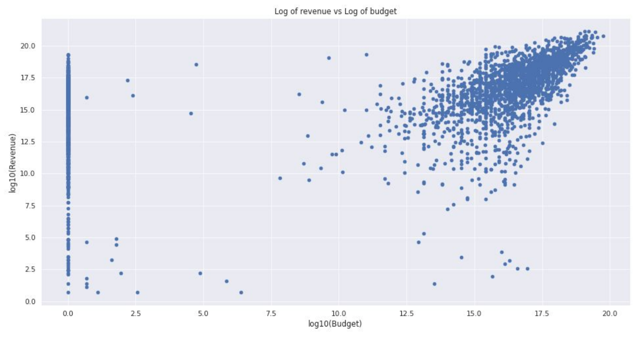


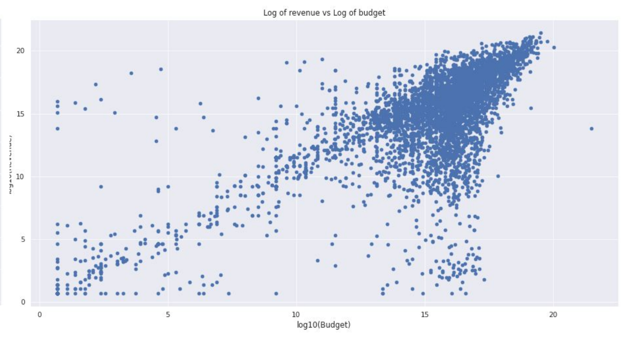


We use the log of revenue as it closely follows a normal distribution and will be easier to predict. All variables are categorical except four - budget, popularity and runtime; and the output revenue. As evident from the figure below, the budget is strongly correlated to its revenue (equal to 0.75).

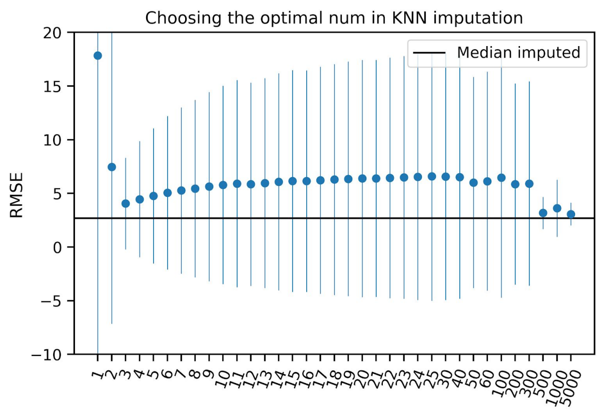
Pre-Processing the Budget Variable:

We further analyse the budget and fill in the missing values (set as zero in the original data). We fill the missing values using the K-nearest neighbours imputation method. The method finds the samples in the training set "closest" to missing values and averages these nearby points to impute the missing values. The data transformation is evident from the below figures (before and after interpolation):



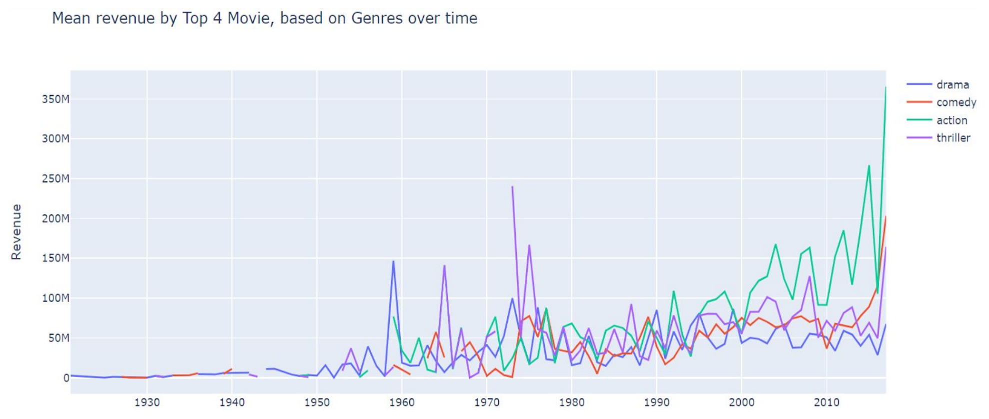


We experiment with different values of k trying to reduce the prediction error. We see that using the mean (k=5000) and median as the imputation value give lesser error than with smaller values of k. These errors are calculated using a simple linear regression model - and checked through a cross-validation scheme.



For the rest of the document, we use two versions of the dataset:

- Median imputed budget variable
- KNN imputed budget variable (k=3)



The genre and release year of the movie is a crucial feature. The variation is shown on the left. The shift at the top from 'comedy' to 'action' movies is evident from the figure.

Other features such as rating, popularity votes are imputed using the median. Ratings for the movies are clipped between 1 and 10.

Feature Engineering

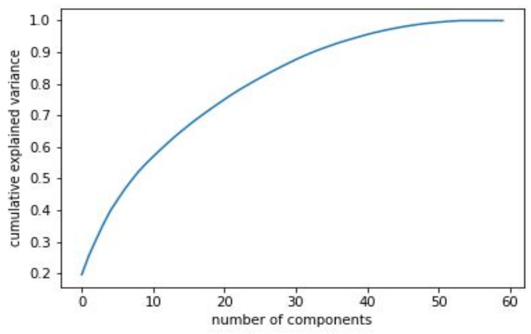
From the Movie Database's official website, we added two new features - rating and the total number of votes. We also added several other features as follows:

Type	Name	Details of the Feature
Binary Features	isTaglineNA	Equal to '1' if the tagline is present, else '0'
	isTitleDifferent	Equal to '1' if the original title is same as the title, else '0'
	isMovieReleased	Equal to '1' if movie already released, else '0'
	BelongsToCollection	Equal to '1' if movie belongs to a collection, else '0'
	isOriginalLanguage	Equal to '1' if the original language of the movie is English
	HasHompage	Equal to '1' if there is a movie webpage, else '0'
Numerical	WeightedRating	Equal to the log of the (total number of votes) *(rating)
	MeanPopularitybyYear	Average popularity of movies released in a given year
	MeanRuntimebyYear	Average runtime of movies released in a given year
	MeanRatingsbyYear	Average rating of movies released in a given year
	MeanTotalVotingsbyRating	Average number of votes to a movie for a given rating
	MeanTotalVotesbyYear	Average number of votes to a movie in a given year
	BudgetRuntimeRatio	Ratio of budget and runtime of a movie

We also replaced cast, crew and production countries with their representative count in for each movie and also included the count of each gender in each movie (0 is unspecified, 1 is female, and 2 is male)

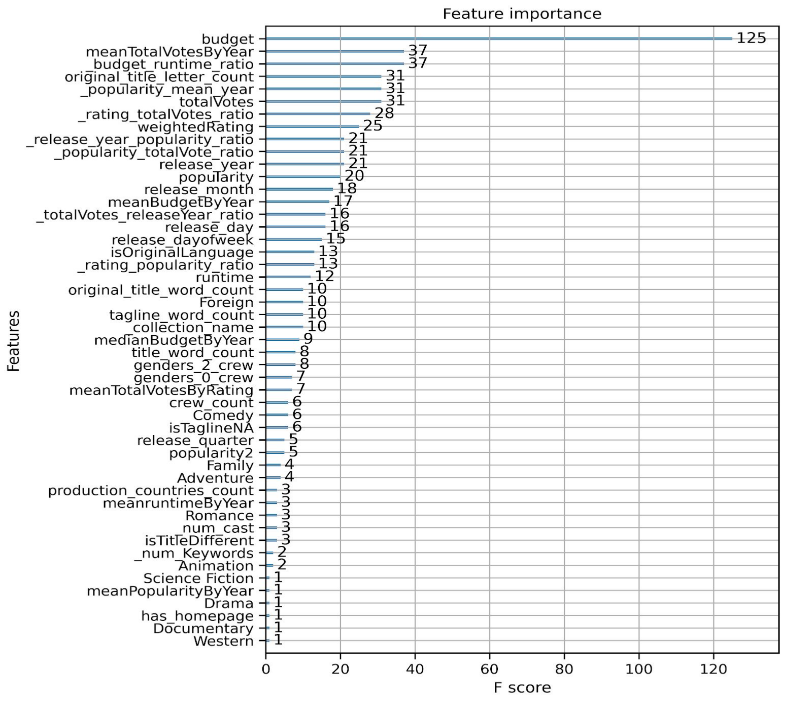
We extracted release year, month and quarter from the release date. Lastly, we used One-Hot encoding for the genres (there are 19 different genres).

Principal Component Analysis



We perform principal component analysis using sklearn. We achieve 95% variance from around 55 features. Therefore we conclude that PCA will not reduce the data size significantly as all features contribute to revenue prediction. We will not use features generated using PCA further in our Machine Learning Model selection.

Feature Importance



Using gradient boosted trees' ability to find the feature importance; we plot the relative values of their significance on the right. We see that the budget value is much higher than other features—our engineered features such as mean total votes & budget runtime ratio in the top 3.

Modelling

After processing the dataset to get rid of inconsistent values, we create predictive regression models for our problem. The target variable is the log of the revenue, and we consistently check the performance of our models against this variable.

K-fold Cross-validation

We use k-fold cross-validation with k = 10, for comparing different models and their results. This ensures that we do not make an overly optimistic or pessimistic estimate of our test error, which can happen while working with random splits.

K = 10 is chosen due to the standard convention, and since our dataset size is small, we could efficiently perform this computation in short times.

Normalisation

Although we don't have much numeric data, we still perform normalisation on the data while feeding it to models sensitive to scale such as Support Vector Regressions, and

Neural Networks. We achieve this by using sklearn's StandardScaler function, which performs **z-score normalisation**. We ensure that we don't leak information about the data by first splitting it and then executing it.

Metrics:

We mainly have different variants of the squared error between the predicted values and the ground truth for regression problems. Here, we compare the models based on two other metrics:

1. **Root Mean Squared Error**
   (tries to minimise the deviation between ground truth and predicted)
2. **R-Squared Error**
   (a measure of how close the data is to the predicted regression line)

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2}$$

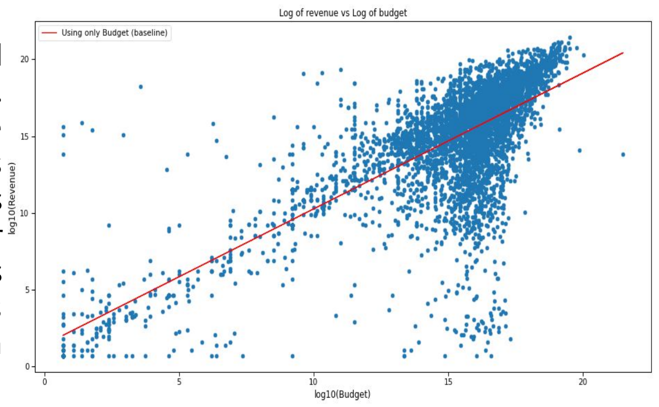
$$R^2 = 1 - \frac{\sum(y_i - \hat{y})^2}{\sum(y_i - \bar{y})^2}$$

The final Kaggle competition uses the Root-Mean-Squared-Logarithmic-Error (RMSLE) as the evaluation metric, equivalent to our RMSE because we have consistently worked with the log of the revenue.

Results of Models and Inferences:

1. **Baseline model**

   It is a simple linear regression model between only the 'budget' feature and the 'revenue' generated by the movie. Among all the features, 'budget' is the most significant one in terms of its impact on the revenue collection. It is taken as a benchmark model in our analysis and all the other regressors for which we use all the features generated are modelled to perform better than this model.



   RMSE = 2.688 ± 0.123, *R2 Score* = 0.543 ± 0.037

2. **Linear Regression**
   A linear regression model fits a linear model with coefficients $w = (w1,\ w2, ...wp)$ to minimise the residual sum of squares between the observed targets in the dataset, and the targets predicted by the linear approximation. Mathematically it solves a problem of the form:

$$\min_{w} ||Xw - y||_2^2$$

   As stated in the table below, this performs poorly than the baseline model itself.

3. **Decision Tree Regressor**
   Decision tree regression observes features of an object and trains a model in a tree's structure to predict data in the future to produce meaningful continuous output. Continuous output means that the output/result is not discrete, i.e., it is not represented just by a discrete, known set of numbers or values.
   It performed poorly than a simple linear regression, possibly due to the fact that it does not have enough complexity to understand the dataset.

4. **Random Forest Regressor**
   Next, we implemented a random forest regressor because it generally outperforms decision trees in terms of accuracy. This is an ensemble learning method for regression and other tasks that operate by constructing a multitude of decision trees at training time and outputting the average prediction (regression) of the individual trees.

   It did perform better than decision trees, due to lower bias - and even better than linear regression, capturing the deviations well.

5. **Support Vector Regressor**
   We have used a regression version of the SVM model using the same principles to minimise the error and individualise the hyperplane, maximising the margin.

   SVMs are notoriously hard to train on datasets of a couple of thousand data points, and therefore we do not try a lot of experiments on them. They performed slightly worse than the gradient boosted trees.

6. **Neural Networks**
   We implemented three different Neural Networks with Sigmoid, ReLu and Tanh activation functions. The model architecture was the same for all three. Neural networks excel at fitting the training data with complex non-linear models. Our dataset size is small for performing deep learning. Hence we do not experiment much.
   Model Specification:- Our model has one hidden layer with 50 hidden units, and it was trained for 400 epochs on the training data with *'adam'* optimiser with a batch size of 512. We use the concept of early stopping to prevent overfitting on datasets.

   Early Stopping: All neural network runs are monitored based on the loss of the validation split (10% of the training data) after every epoch. When the validation loss does not improve for more than 20 epochs, then the training is stopped. This helps us to automate the stopping process and avoid overfitting.

7. **XGBoost (Gradient Boosted Trees)**
   Gradient boosting is an approach where new models are created that predict the residuals or errors of prior models and then added together to make the final prediction.
   We implemented XGBoost because it is really fast compared to other gradient boosting implementations, and it dominates structured datasets on classification and regression predictive modelling problems. It was the best model in terms of the RMSE score.

Summary of Performance of models

The results of the evaluation of our models on different metrics are tabulated below:

REGRESSION MODEL	Median Imputed Data		KNN Imputed Data (k=3)	
	RMSE SCORE	R2 SCORE	RMSE SCORE	R2 SCORE
Baseline model	2.603 +/- 0.122	0.571 +/- 0.036	2.688 +/- 0.123	0.543 +/- 0.037
Linear Regression	2.698 +/- 0.663	0.518 +/- 0.251	4.053 +/- 4.270	-1.223 +/- 5.311
XGBoost (Gradient Boosted)	2.224 +/- 0.106	0.687 +/- 0.032	2.265 +/- 0.124	0.674 +/- 0.038
Random Forest	2.260 +/- 0.110	0.676 +/- 0.033	2.319 +/- 0.108	0.659 +/- 0.032
Support Vector Regression	2.404 +/- 0.129	0.634 +/- 0.032	2.426 +/- 0.138	0.628 +/- 0.034
Neural Networks (relu)	2.893 +/- 1.594	0.303 +/- 1.034	3.594 +/- 3.528	-0.630 +/- 3.785
Neural Networks (tanh)	2.360 +/- 0.107	0.647 +/- 0.031	2.396 +/- 0.109	0.637 +/- 0.029
Neural Networks (sigmoid)	2.354 +/- 0.105	0.649 +/- 0.031	2.380 +/- 0.113	0.641 +/- 0.031
Decision Tree	3.162 +/- 0.220	0.364 +/- 0.097	3.319 +/- 0.141	0.302 +/- 0.069

Hyperparameter Tuning performed on XGBoost:

For several hyperparameters in the gradient boosted trees algorithm, we perform a hyperparameter tuning experiment. We check 1728 different models, with 5-fold validation fits, and we find the best model. This model performs at an RMSE of 1.922 (when trained on the entire training dataset) on the Kaggle Leaderboard giving a rank of 390th amongst 1500+ participants.

Conclusion

Budget is the most important feature to predict revenue. A logical explanation for the same could be that the quality of crew, cast and other resources depends heavily on the initial investment. All these factors increase the correlation between revenue and budget.

We find that gradient boosted trees performed the best on this dataset. It explains approximately 70% variance in the movie's revenue at the box office worldwide.

Acknowledgements:

*We wish to express our sincere gratitude to our instructor **Dr. Faiz Hamid** for his valuable advice and guidance in completing this project. The way he presented each and every topic in the class made the topics very interesting and understandable, which was of essential essence in making our project possible.*