Predictive Movie Analytics Using IMDB on Hollywood Movies

I. Introduction

"Cinema is the most beautiful fraud in the world" – Jean-Luc Godard. You spend money on cinema to relive a visual experience which has been crafted by a team in all likelihood assuming that you are not going to experience it in your own life. The catch with the fraud, is that people spend money on them. Our goal is to predict how much money people are going to spend on a movie given a few features of the movie and in turn make Hollywood community really rich.

II. Pipeline/Method

a.) Hardware

Given the hardware of more than 100 GB's of storage, 96 GB of RAM (Random Access Memory) coupled with 4 Dual cores Xeon processor running at 2.3 Ghz, translating to 8 virtual cores.

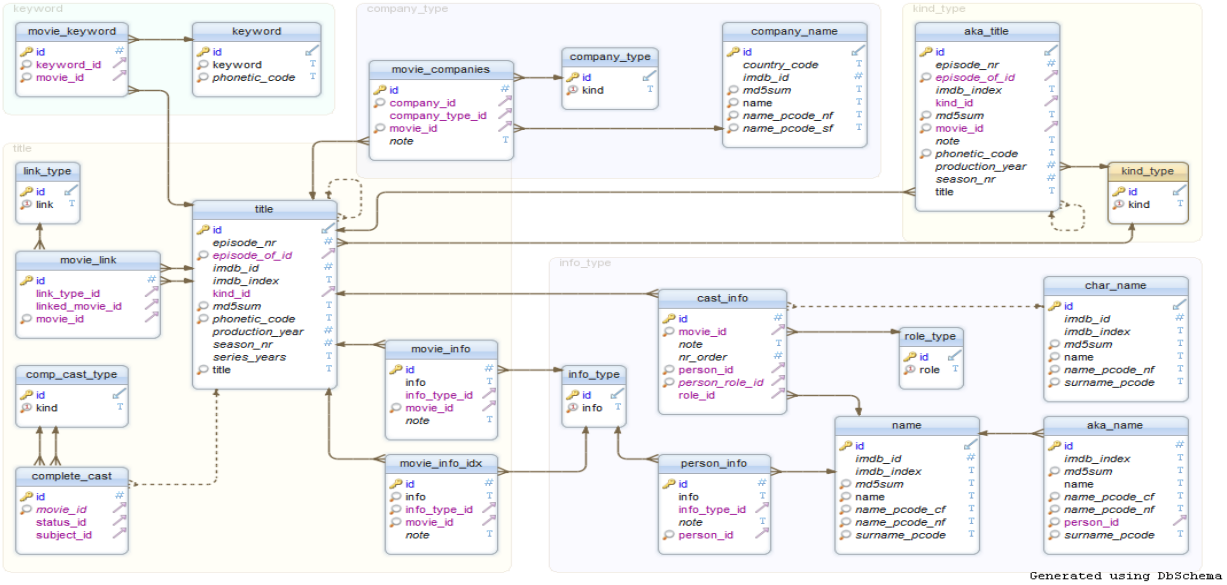
b.) Dataset

IMDB (International Movies Database) allows their data to be accessible to general public in form of plain text data files which are compressed [1]. They also offer Unix command line search programs, but going that way would have been unfruitful as it's not supported by IMDB anymore and our criteria was grabbing as much as data as possible [2].

The plain text files extracted from IMDB contained information along the lines of Actors, Actresses, Movie Budget, Gross Revenue of the movie, Genres, Countries, Languages, Title and Directors. For a full list of the files please refer [3]. This data set was approximately 650 MB (Megabytes) when compressed. On decompression it is equivalent to 2.4 GB (Gigabytes).

c.) Extraction API

Multiple open source libraries/APIs (Application Programmable Interface) exist for accessing IMDb online as well. Our selection was IMDbPY API which could retrieve data from both IMDb web server and a local copy of the whole database [4]. Data Base Schema [6]:



d.) Database

Setting up a local copy of the database was necessary as retrieving information per movie one by one using the web server would be cumbersome and require a lot of time as each retrieval would be a ping to the server. However IMDbPY offers a useful script imdbpy2sql.py [5] which can take an input directory containing the plain text files in compressed format and create a Database in different schemes for example sqlite, mysql, postgres, firebird and more. Our selection was postgresql.

The total Database size when generated is 8.98 GB.

e.) Methodology

postgresql queries can wrapped in python to execute them collectively one by one. Psycopg API allows multiple features of the psql to be accessed by python [7]. Multiple sub-queries were written to extract specific features per movie (See Feature Selection). The sub queries were all amalgamated into one single giant query to obtain a matrix.

The matrix had int and string data types, which was all converted to int. For example Gross revenue extracted from the database had country and year listed in it. Data cleaning was performed using python and the final matrix was generated in the form of a list of lists. Where a single list per movie contains all the features.

This aggregate matrix was dumped using **pickle** python object serializable format implying that this dump could be read by another python program in the same format as it was dumped.

Aggregate matrix generated after querying and cleaning was broken up into input and output matrices where input is a 2D matrix of 'samples' by 'features' and output is 1D matrix which contains the dependent variable. The trained machine learning model has to predict the dependent variable.

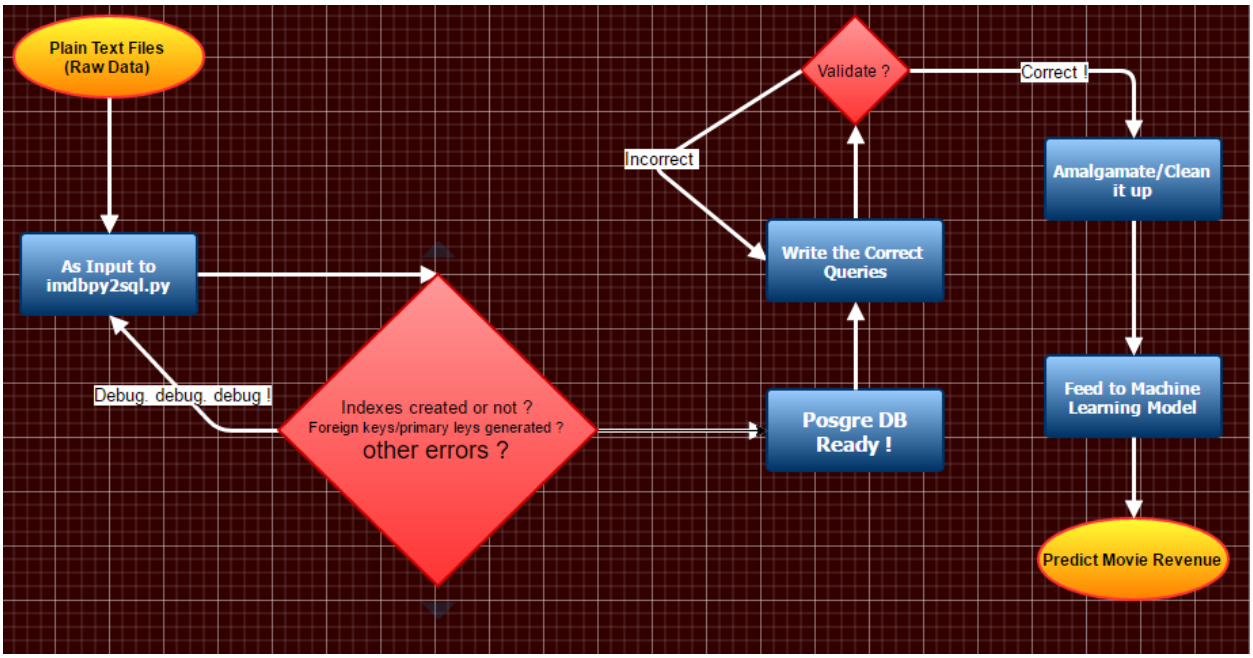


Fig 1.2 Pipeline (was created using [8])

III. Machine Learning

a.) Feature Selection

All features were extracted by movie per their 'id': Gross Revenue, Budget, Genres tagged to a movie, Languages in which the movie is/will be released, Number of countries where movie will be released.

b.) Data Points

Total distinct titles present in this dataset were 2,069,235. However they all could not be considered valid data points. A data point was only considered if all the features above existed and then stored as a list in a list. To prevent currency violation we only considered movies where the revenue and the budget were mentioned in dollars. Queries were written with regex matching to make the same happen. Following all the conditions the final resulting set was of 68,679 data points which consisted of all features row wise like a list.

c.) Machine Learning Models

As mentioned above in Pipeline the resulting matrix was broken into input 'X' and output matrix 'Y'. Machine Learning models implemented: **Linear Regression, Lasso Regression, Logistic Regression** and **Gaussian Naïve Bayes**. Scikit learn library was used to implement these models [9]. Linear and Lasso were used to predict as to how much gross revenue the movie will make. Their score is the coefficient of determination its value should lie between 0 and 1, if it is effectively learning something, it could be negative arbitrarily suggesting the model is not good at all for the scenario. Determination coefficient is plotted as function of different test, train sizes.

Logistic and GaussianNB were used to answer a question as to whether gross revenue will exceed (>) x* budget, where x is used as input to the function for accuracy of the models. x = {2,3,4,5,6,7}

IV. Results

In Fig 3.1 Determination Coefficient varies between 0.41 and 0.33 signifying that the Model does learn fair amount. However it goes on to show that increasing the number of data points will probably increases its effectiveness.

Fig 3.2 shows increasing accuracy with the question being answered as to whether the movie will make 'n' times of the budget or not. However in all likelihood it is misleading as such a high accuracy is generally not expected. An accountable reason could be that there are very few data points that satisfy this criteria and hence the model learns little and in the test set there might not be any data points at all with the same thus resulting in faulty accuracy. However using this to answer where n = 3 or 4 seems to be a suitable match.

Fig 3.1 (Obtained from linear regression and lasso regression)

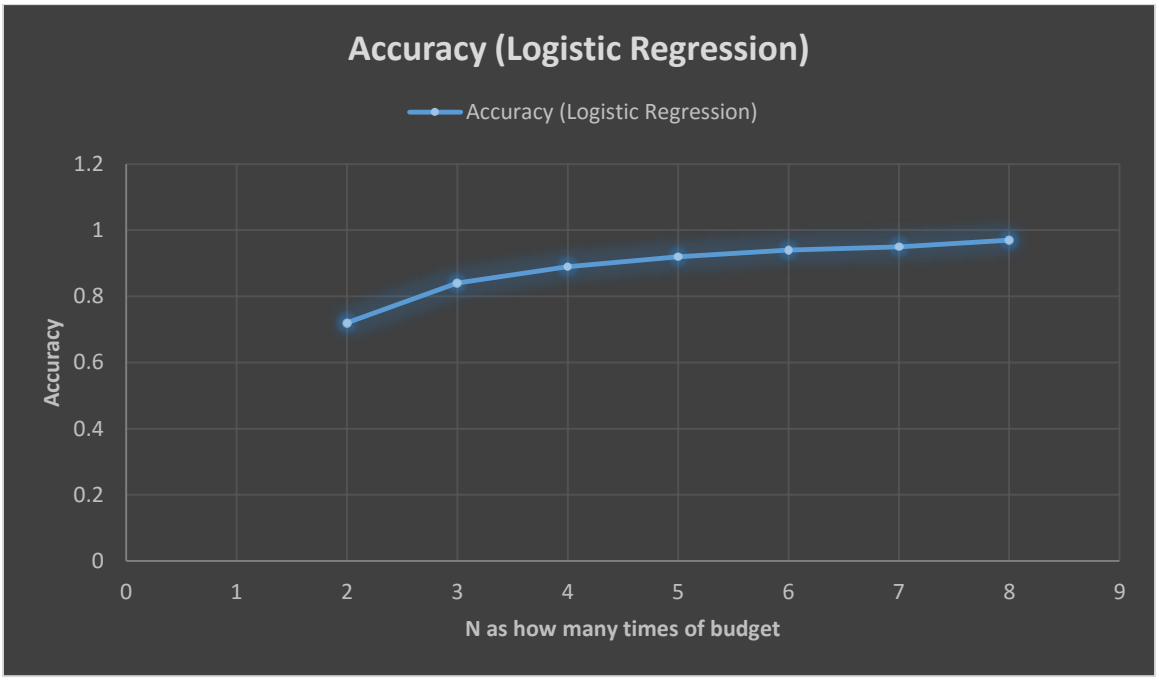


Fig 3.2 (Obtained from Logistic Regression and Gaussian Naïve Bayes)

V. Future Work

To improve upon this work I would like to incorporate more useful features. For example a success fraction of the actors, actresses and directors involved in the movie. This fraction could be defined by the number of total successful movies (if revenue crosses a certain threshold in relation to the budget) divided by their total number of movies. Also implement this on different machine learning models with more dependent variables rather than just one.

VI. Big Data Keywords that I can tag here: Postgresql, Scikit learn, pickle, Big data set, IMDbPY, Large plain text files, Cluster, Larger than traditional memory.

References:

1.) ftp://ftp.funet.fi/pub/mirrors/ftp.imdb.com/pub/
2.) http://www.imdb.com/interfaces
3.) actors.list.gz, directors.list.gz, movies.list.gz, actresses.list.gz, distributors.list.gz, ratings.list.gz, business.list.gz, genres.list.gz, release-dates.list.gz, countries.list.gz, language.list.gz, taglines.list.gz
4.) http://imdbpy.sourceforge.net/
5.) http://imdbpy.sourceforge.net/docs/README.sqldb.txt
6.) http://stackoverflow.com/questions/25926003/description-of-the-sql-movie-databse-made-by-imdbpy
7.) http://initd.org/psycopg/
8.) https://www.draw.io/
9.) http://scikit-learn.org/stable/

Submitted by:

Vishwesh Nath

Email: vishwesh.nath@vanderbilt.edu