Lab 1 – 094295

Stas Silberberg	319206850
Eyal Bar Natan	207630658

https://github.com/Silber93/da_2_lab_hw_1.git

Exploratory Data Analysis

a. Which features are available in the dataset

1) *backdrop_path*: probably a sort of a path to a picture or a poster that is related to the movie.
2) belongs_to_collection:
3) budget: the budget of the movie.
4) genres: a dictionary contains all the genres that this movies is related to.
5) homepage:
6) id: unique id of the movie.
7) imdb_id: the id of the movie on IMDB.
8) original_language: the original language of the movie.
9) original_title: the title of the movie in its original language.
10) overview: a brief description of the movie.
11) popularity: the popularity score of the movie in IMDB.
12) poster_path: some sort of link to the poster that's related to the movie.
13) production_companies: a dictionary explaining the companies participated the making of the movie.
14) production_countries: a dictionary containing the countries in which the movie was produced.
15) release_date: the release date of the movie.
16) runtime: the length (in minutes) of the movie.
17) spoken_languages: a dictionary containing all spoken languages in the movie
18) status: whether the movie released or not.
19) tagline: a special phrase that
20) title: title of the movie in English.
21) video: format of the movie.
22) vote_average: average score on IMDB.
23) vote_count: number of score votes on IMDB.
24) Keywords: a dictionary of keyword relevant to the movie.
25) cast: a dictionary containing information about every actor participating in the movie.
26) crew: a dictionary containing information about every staff that is related to the movie.
***27) revenue*: the revenue of the movie.**

b. feature distribution

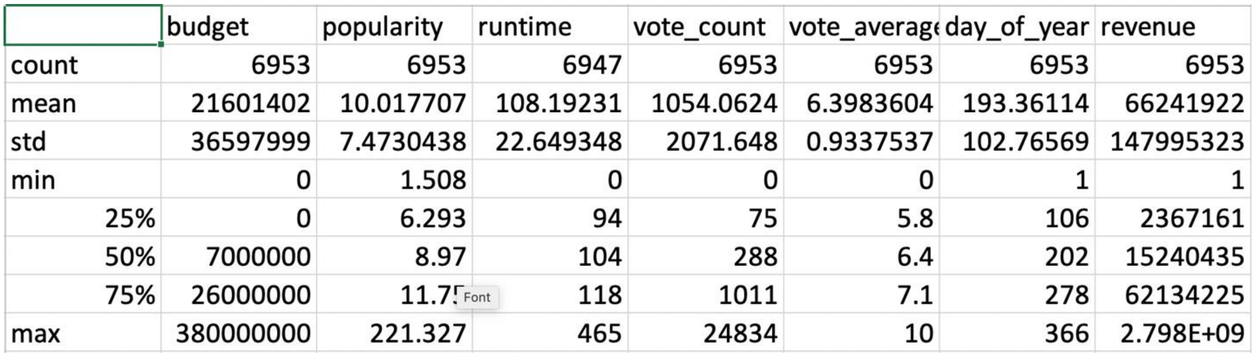
features with continuous distribution:

• Revenue	• Popularity
• Budget	• Vote_count
• Runtime	• Vote_average

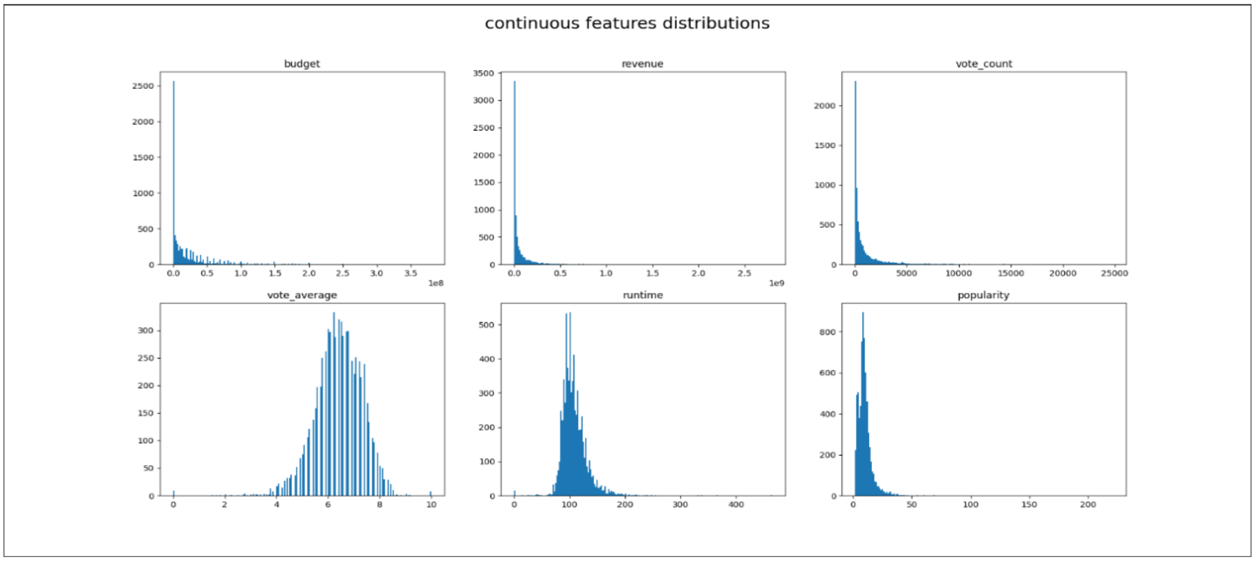
features with categorical distribution:

- Backdrop_path
- Belongs_to_collection
- Genres
- Homepage
- Id
- Imdb_id
- Original_language
- Original_title
- Overview
- Poster_path
- Production_companies
- Production_countries
- Spoken_language
- Status
- Tagline
- Title
- Video
- Keywords
- Cast
- Crew

Feature with time distribution: release date.



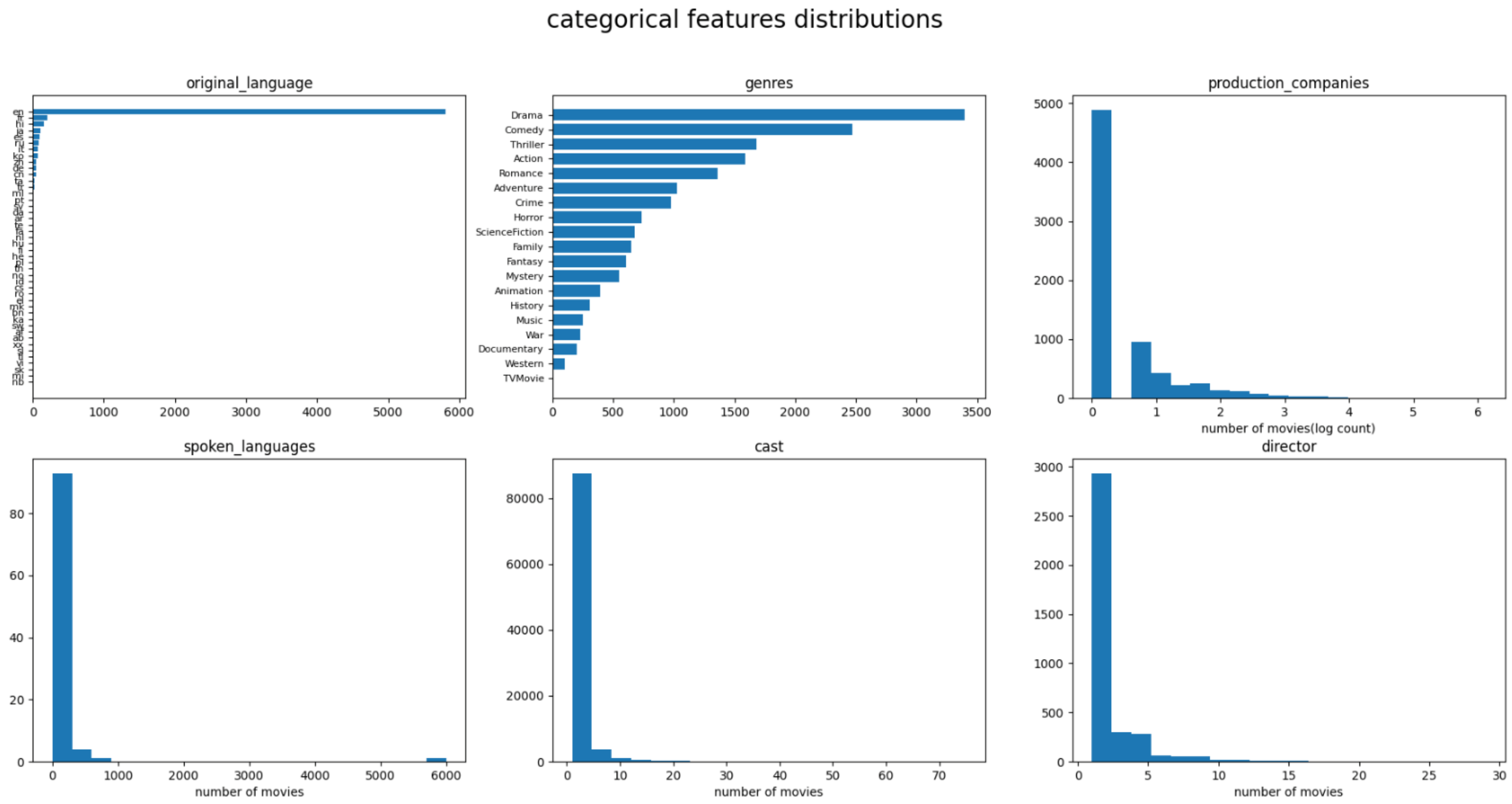
	budget	popularity	runtime	vote_count	vote_average	day_of_year	revenue
count	6953	6953	6947	6953	6953	6953	6953
mean	21601402	10.017707	108.19231	1054.0624	6.3983604	193.36114	66241922
std	36597999	7.4730438	22.649348	2071.648	0.9337537	102.76569	147995323
min	0	1.508	0	0	0	1	1
25%	0	6.293	94	75	5.8	106	2367161
50%	7000000	8.97	104	288	6.4	202	15240435
75%	26000000	11.7[illegible] Font	118	1011	7.1	278	62134225
max	380000000	221.327	465	24834	10	366	2.798E+09



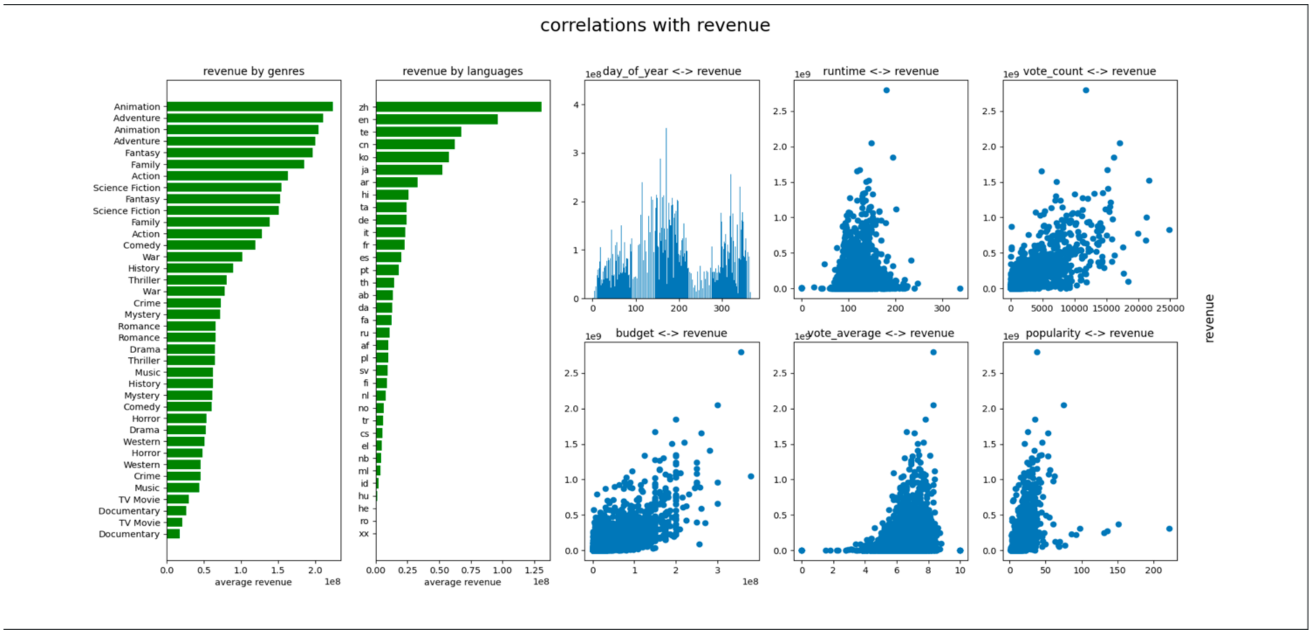
To stay relevant to the task, we chose to analyze the categorical features that might have the greatest effect on the movie's revenue in our opinion. We chose the next features:

- ○ Original_language
- ○ Spoken_languages
- ○ Genres
- ○ Cast
- ○ Production companies
- ○ Director (extracted from ‘crew’ feature)

Feature	Unique values
Original language	44
Spoken_language	99
Genres	19
Cast	93751
Production companies	7186
Director	3733



Next, we want to analyze the correlation of some features with the movie’s revenue in order to determine whether a feature is valuable and relevant to our task.



c. Missing data

1) the features ‘budget’ and ‘revenue’: we can observe that out of 6953 total movies, there are 1981 movies with zero budget, which doesn’t make sense since nothing in this world is free, especially movies which might demand millions of dollars for production. We decided to define every budget and revenue values that is under 1000 as a missing value, which results 2029 missing values for ‘budget’ and 126 missing values for ‘revenue’

2)

feature	Missing values
backprob_path	747
poster_path	217
belongs_to_collections	5534
homepage	4580
Imdb_id	16
Overview	7
Tagline	1371
genres	19
Production companies	222
Spoken languages	13
Cast	17
Crew	10
Budget (under 1000)	2029
Revenue (under 1000)	126
runtime	6

All other features have no missing values.

Feature Engineering

a. Features to use

The features that we will use are:

- Budget – the budget of a movie is directly connected to its revenue, due to financial reasons: every movie aims towards a revenue that is higher than the budget, so higher budget will usually lead to higher revenue.
- Original language – the original language of a movie has a great effect on the number of people who can watch it. Movies in Chinese or in English for example, would have a potential to easily reach billions of people, and thus will have greater revenue.
- Popularity – the popularity of a movie represents the number of people who watched the movie
- Runtime – movies that are too short or too long may affect the will of people to watch it. Besides, we know that successful movies usually take 90 to 140 minutes.
- Spoken languages – gives us more information about the languages that were used in the movie, and therefore may tell us about the potential audience, similar to 'original language'.
- Vote count – a large voting rate means that the movie is more popular, and therefore the importance of this feature is similar to 'popularity'
- Vote average – the average score of vote represents how much people enjoyed the movie, and higher voting average will probably mean that the movie is more enjoyable, and more people will go and watch it.
- Release date – the time of release will usually have a crucial effect on the success of the movie. For example, we know that many companies wait until holiday season to release a movie, because people will have greater availability and tend to do more recreation activities, such as going to the movies.
- Cast – the actors may sometimes be the whole reason to watch the movie. Moreover, top class actors will usually accept to participate in top class movies that aim to high revenues.
- crew – the crew have the greatest impact on how the movie will eventually look like.
- Genre – from the exploratory section we see that there is a correlation between the genre of a movie and its revenue.

b. Feature transformation

1) 'Release date' to 'day of the year' + 'year': the 'release date' feature contain the date of the release (year-month-day). We extracted the day of the year and saved it instead and saved the year itself as another feature.
2) Categorical feature to vector:
   As said before, we chose to use a few categorical features in our task. The task is a regression problem, and to use the categorical features, we had to replace them

with numeric representation of themselves. Therefore, we created a dictionary for each feature with all its possible values (taken from both train and test files) and for every value in every feature we created an index vector. We used the following features:

   a. Genres
   b. Original language
   c. Spoken languages
   d. Production companies
   e. Cast – since there are 93751 different actors, the creation of 6953 vectors of length 93751 and using them in the model will be very expensive. Therefore, we created a vector from the leading 2 actors in each movie, resulting a vector of length 11496.
   f. Director – extracted from the 'crew' feature.

c. Handling missing data
   1) Budget and revenue – we will use mean imputation to complete the missing data.
   2) Director – movies that don't have a director in the 'crew' feature will be transformed to an index vector of zeros.
   3) Cast – same as director.
   4) Runtime – drop all 6 rows.

Prediction

We examined 3 options for applying a regression model that would predict the value of the movie's revenue. We assumed that there might be a linear relationship between the covariates we analyzed above and the revenue feature. Therefore, we chose the following methods for making such prediction:

1. Linear Regression
2. Ridge Regression
3. Custom Regression – iteratively minimizes the RMSLE of the prediction.

We applied a cross-validation on the ridge regression, where we could iterate over the 'alpha' parameter which determines the power of the regression's regularization.

In addition to the classic linear models of *sklearn* (the linear an ridge), we wanted to examine a regression that will make its loss optimization on the RMSLE function, same as the evaluation score function we use in this exercise. Thus, we built a custom model that applies a minimization that uses a custom loss function that we can manually define (different from the default options offered by sklearn library). We used the iterative minimization function of *scipy.optimize* library, with an initial regression vector of ones. We applied a cross-validation on this regression, which clearly had better results than the other two regressions.
We iterated over the **'method' parameter** which determines the kind of minimization which is applied in this regression. We received the following results:

	RMSLE
‘Powell’	8.19
‘CG’	11.024
‘BFGS’	12.29
‘Nelder-Mead’	15.63

We used the method ‘Powell’, which brought us the best result among these five methods. We came up with the following results while training these models on the **training data**:

	RMSLE
Linear Regression	6.993
Ridge Regression	6.993
Custom Regression	11.48

While performing these models on the **test data**, we got:

	RMSLE
Linear Regression	7.058
Ridge Regression	7.03
Custom Regression	11.31

We found out that the performance of our custom regression was relatively poor, so we tried to apply it with the second-best performance, the ‘BFGS’. We received a test error of 5.20.