Saint-Petersburg Electrotechnical University

LETI

Machine Learning on Big Data

Report 1

Auteurs:
Alexia GROSS

Enseignant:
Ivan IVANOVITCH

November 7, 2021

Contents

Chapter 1

Subject presentation

1.1 Analytic task for selected data

- General information of the selected data: the source provided, analytic tasks.
- The data analysis target
- Meta-information: format, number of attributes and vectors, attribute types, classes, etc.
- Data restrictions: missing values, anomalies, etc.
- Machine learning algorithm for solving the task
- Necessary data settings for the machine learning algorithm.
- Expected mining models built by algorithm.
- Methods and criteria for assessing for built mining model.
- Conclusions

Chapter 2

The analysis

2.1 General information

2.1.1 The source provided

The data can be found at this link (Kaggle): `https://www.kaggle.com/c/tmdb-box-office-prediction/data?select=train.csv`

The dataset chosen contains 21 attributes. The following picture shows an extract of 10 attributes.

	id	belongs_to_collection	budget	genres	homepage	imdb_id	original_language	original_title	overview	popularity
0	1	[{'id': 313576, 'name': 'Hot Tub Time Machine ...	14000000	[{'id': 35, 'name': 'Comedy'}]	NaN	tt2637294	en	Hot Tub Time Machine 2	When Lou, who has become the "father of the In...	6.5754
1	2	[{'id': 107674, 'name': 'The Princess Diaries ...	40000000	[{'id': 35, 'name': 'Comedy'}, {'id': 18, 'nam...	NaN	tt0368933	en	The Princess Diaries 2: Royal Engagement	Mia Thermopolis is now a college graduate and ...	8.2489
2	3	NaN	3300000	[{'id': 18, 'name': 'Drama'}]	http://sonyclassics.com/whiplash/	tt2582802	en	Whiplash	Under the direction of a ruthless instructor, ...	64.3000
3	4	NaN	1200000	[{'id': 53, 'name': 'Thriller'}, {'id': 18, 'n...	http://kahaanithefilm.com/	tt1821480	hi	Kahaani	Vidya Bagchi (Vidya Balan) arrives in Kolkata ...	3.1749
4	5	NaN	0	[{'id': 28, 'name': 'Action'}, {'id': 53, 'nam...	NaN	tt1380152	ko	마린보이	Marine Boy is the story of a former national s...	1.1481

5 rows × 23 columns

Figure 2.1: Extract of the dataset

2.1.2 The dataset's Goal

The analytic task is to predict the worldwide box office revenue on over 7,000 past films from The Movie Database.
Data points include cast, crew, plot keywords, budget, posters, release dates, languages, production companies, and countries.

2.2 The data analysis target

Goal

The data analysis target is to **determine a movie's popularity before its release**. We will also analyze the dependence and influence of each attribute in order to observe if some of them could be removed and which one has the most significant impact on the popularity.
I will perform the study with two different models in order to compare them.

I used Keras library from Tensorflow to perform the Machine Learning part.

2.3 Meta-information

2.3.1 Format

The data are stored in CSV files.
Two files are available :

- train_set.csv : to train our model
- test_set.csv : to test our model

2.3.2 Number of attributes, attribute types and vectors

The dataset contains 22 attributes (without the id). Movies are labeled with "id". Two facts can be noticed :

- The data are multimodal. The attributes are either "int64" so Integer, "float64" or "object".
- Some data contain a big amount of "NaN" values.

Thus, I will have to perform a pretreatment on the data in order to manage the multimodal aspect and filter the outliers.

Let us consider the attribute "Genre." The movie genre provides important information about a movie and thus, can impact the popularity. Our study includes 22 genres : Crime, Documentary, Music, etc. A movie could belong to several genres. For this reason, the genre variable is represented as a binary vector of length 22, which is initialized to zeros. For each genre a movie belongs to, the corresponding position in the vector was set to one.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3000 entries, 0 to 2999
Data columns (total 23 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   id                     3000 non-null   int64
 1   belongs_to_collection  604 non-null    object
 2   budget                 3000 non-null   int64
 3   genres                 2993 non-null   object
 4   homepage               946 non-null    object
 5   imdb_id                3000 non-null   object
 6   original_language      3000 non-null   object
 7   original_title         3000 non-null   object
 8   overview               2992 non-null   object
 9   popularity             3000 non-null   float64
 10  poster_path            2999 non-null   object
 11  production_companies   2844 non-null   object
 12  production_countries   2945 non-null   object
 13  release_date           3000 non-null   object
 14  runtime                2998 non-null   float64
 15  spoken_languages       2980 non-null   object
 16  status                 3000 non-null   object
 17  tagline                2403 non-null   object
 18  title                  3000 non-null   object
 19  Keywords               2724 non-null   object
 20  cast                   2987 non-null   object
 21  crew                   2984 non-null   object
 22  revenue                3000 non-null   int64
dtypes: float64(2), int64(3), object(18)
memory usage: 539.2+ KB
```

2.4 Data restrictions

2.4.1 Missing values

In the previous figure, we can observe that the dataset contains 3000 entries. For each attribute, we have the number of "Non-Null" values.
Thus, here is the number of missing values per attribute :

Attributes	Types	Number of missing values
belongs_to_collection	object	2 396
budget	int64	0
genres	object	7
homepage	object	2 054
imdb_id	object	0
original_language	object	0
original_title	object	0
overview	object	8
popularity	float64	0
poster_path	object	1
production_companies	object	156
production_countries	object	65
release_date	object	0
runtime	float64	2
spoken_languages	object	20
status	object	0
tagline	object	597
title	object	0
Keywords	object	276
cast	object	13
crew	object	16
revenue	int64	0

2.4.2 Anomaly

In order to count the outliers, I have drawn the boxplots of each attribute.

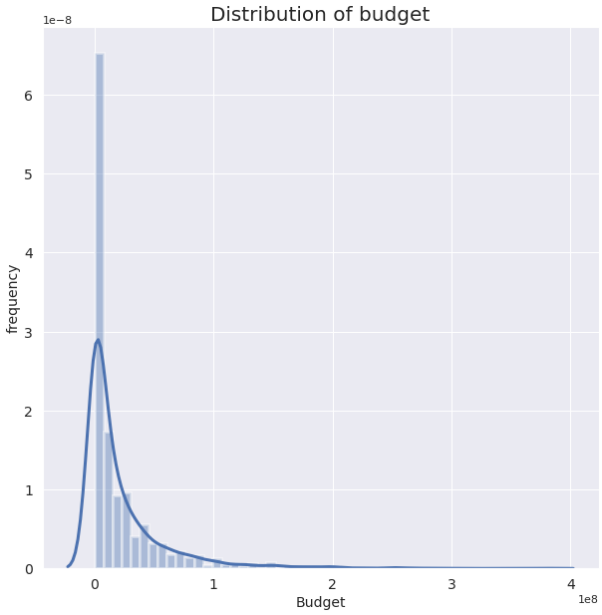

Figure 2.2: Budget Attribute - Histogram

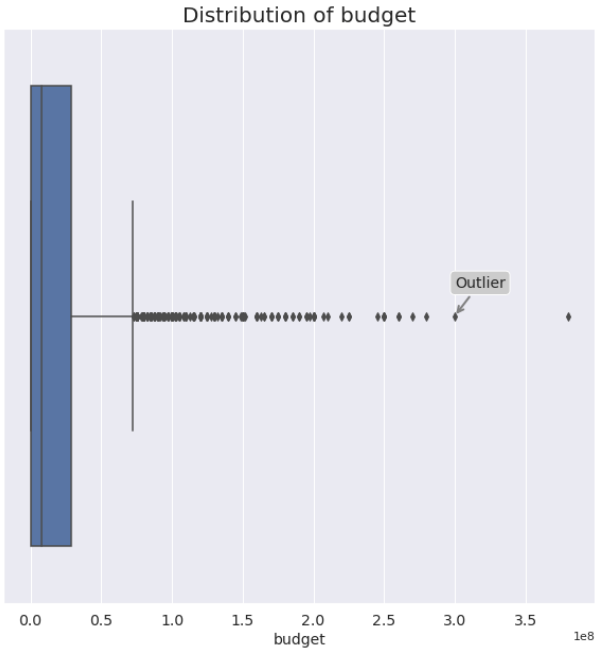

Figure 2.3: Budget Attribute - Box plot

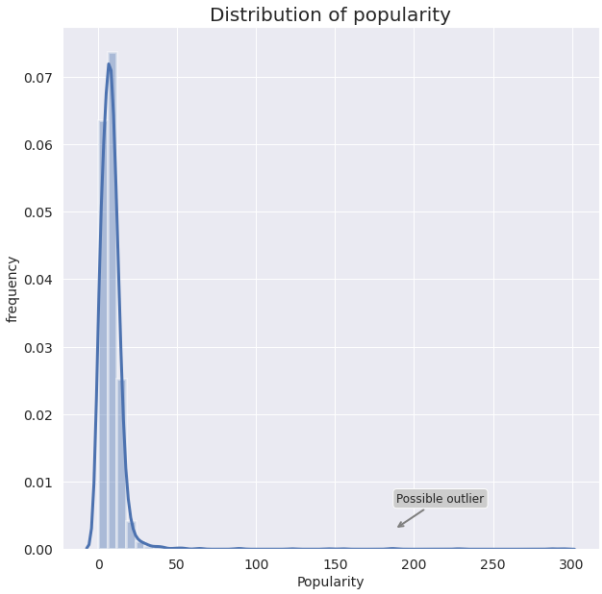

Figure 2.4: Popularity Attribute - Histogram

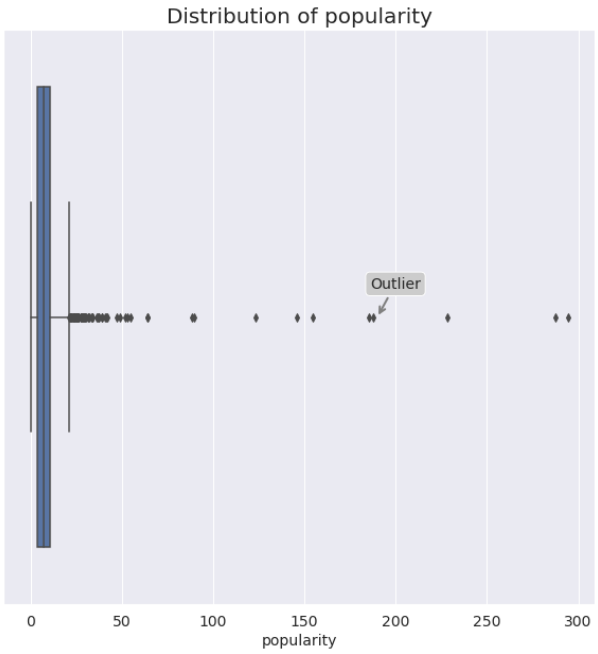

Figure 2.5: Popularity Attribute - Box plot

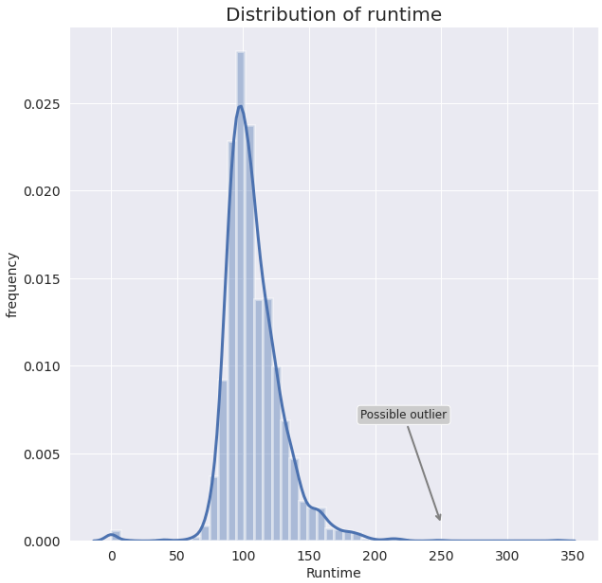

Figure 2.6: Runtime Attribute - Histogram

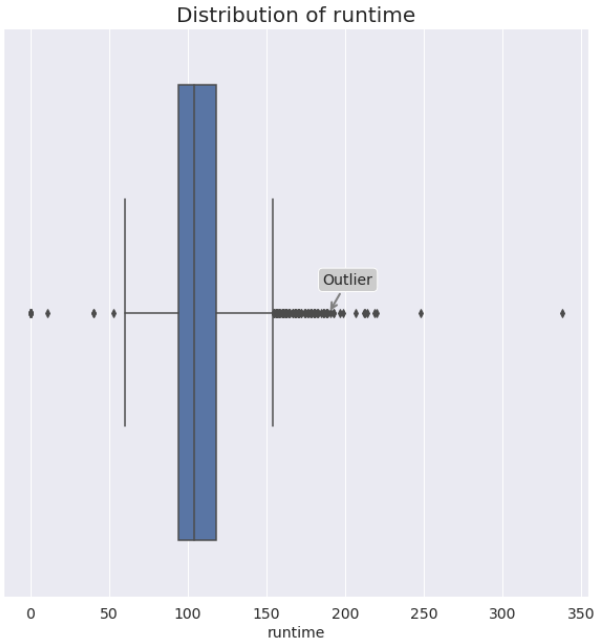

Figure 2.7: Runtime Attribute - Box plot

Figure 2.8: Revenue Attribute - Histogram

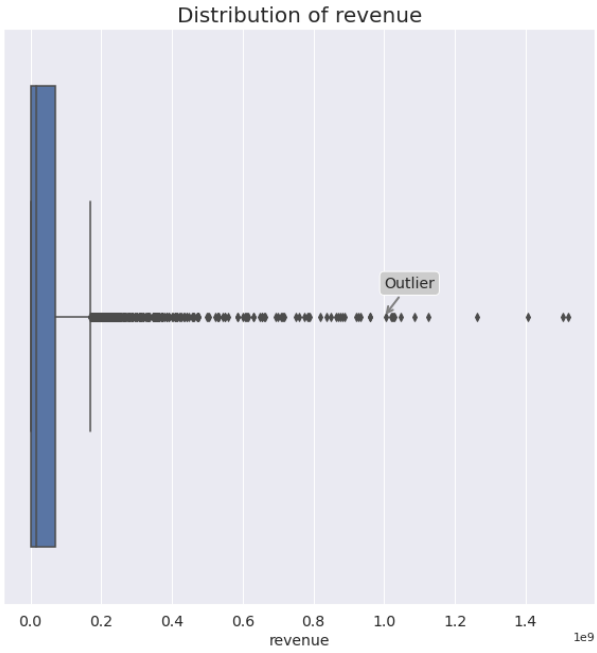

Figure 2.9: Revenue Attribute - Box plot

The number of outliers can't be counted by hand. So I implemented a function to calculate it. The results are shown in the following picture.
It seems to be a lot of outliers, but if we compare them to the number of values for these attributes initially, the number of these outliers is minimized :

- Budget : 249 outliers and 3000 values.
- Popularity : 70 outliers and 3000 values.

- Revenue : 315 outliers and 3000 values.
- Runtime : 126 outliers and 2998 values.

```
((df < (Q1 - 1.5 * IQR)) | (df > (Q3 + 1.5 * IQR))).sum()

Keywords                 0
belongs_to_collection    0
budget                 249
cast                     0
crew                     0
genres                   0
homepage                 0
id                       0
imdb_id                  0
original_language        0
original_title           0
overview                 0
popularity              70
poster_path              0
production_companies     0
production_countries     0
release_date             0
revenue                315
runtime                126
spoken_languages         0
status                   0
tagline                  0
title                    0
dtype: int64
```

Figure 2.10: Outliers for each attributes

2.5 Necessary data settings for the machine learning algorithm

First, we will remove the attribute "Revenue." The goal is to predict a movie's popularity before its release; thus, it will be absurd to have the revenue already.

Secondly, we will remove all the attributes which contains less than 4000 non-null data : "belongs_to_collection", "homepage" and "tagline".

We take a look at the "popularity" attribute. We can observe that the mean is 8.46 with 75% of the results below 10.85. If we compare this result with the twenty-first film, we can notice outliers. To avoid an impact on our prediction, we fix the maximum to 40 (a figure which corresponds to the 20th film the most popular).

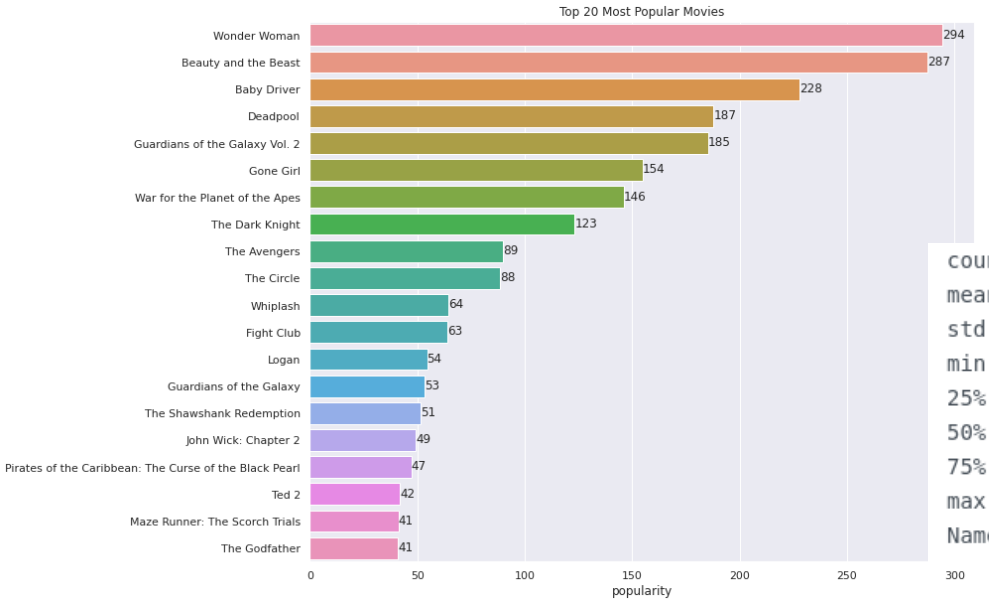

Figure 2.11: 20 first films

```
count   3,000.0000
mean        8.4633
std        12.1040
min         0.0000
25%         4.0181
50%         7.3749
75%        10.8910
max       294.3370
Name: popularity, dtype: float64
```

Figure 2.12: Popularity attribute

Additionally, the multimodal variables need to be taken into account. After some researches, Hoang Dang presents a pre-treatment system that corresponds to our type of data.
We first need to adapt our variables to match the ones used by Hoang Dang. The film release date is "2015-02-20". We need to separate it in three variables : $y = 2015$, $m = 02$, $j = 20$.
Regarding the other "text" variables, we will use a predefined class "InfoExtractor" to separate and extract the data.

2.5.1 Step 1

	id	belongs_to_collection	budget	genres	homepage	imdb_id	original_language	original_title	overview	popularity
0	1	[{'id': 313576, 'name': 'Hot Tub Time Machine ...	14000000	[{'id': 35, 'name': 'Comedy'}]	NaN	tt2637294	en	Hot Tub Time Machine 2	When Lou, who has become the "father of the In...	6.5754
1	2	[{'id': 107674, 'name': 'The Princess Diaries ...	40000000	[{'id': 35, 'name': 'Comedy'}, {'id': 18, 'nam...	NaN	tt0368933	en	The Princess Diaries 2: Royal Engagement	Mia Thermopolis is now a college graduate and ...	8.2489

Figure 2.13: Step 1

2.5.2 Step 2

	genres	production_companies	production_countries	spoken_languages	Keywords
0	Comedy	4 60 8411	US	English	time_travel sequel hot_tub duringcreditsstinger
1	Comedy Drama Family Romance	2	US	English	coronation duty marriage falling_in_love
2	Drama	2266 3172 32157	US	English	jazz obsession conservatory music_teacher new_...
3	Thriller Drama	NaN	IN	English हिन्दी	mystery bollywood police_corruption crime indi...
4	Action Thriller	NaN	KR	한국어/조선말	NaN

Figure 2.14: Step 2

Then, we will apply another predefined class "TextEncoder" to collect all the "strings" variables and convert them into "float" variables.

After applying this two classes, we can finally use our data. We obtain the following result :

2.5.3 Step 3

	budget	genres	original_language	original_title	overview	poster_path	production_companies	production_countries	runtime	spoken_languages
0	14000000	1.3649	0.7722	29.7698	86.2111	8.0070	10.7776	0.8390	93.0000	0.7634
1	40000000	6.8065	0.7722	32.2026	232.2292	8.0070	3.8840	0.8390	113.0000	0.7634
2	3300000	1.0846	0.7722	7.3139	75.4125	8.0070	17.8654	0.8390	105.0000	0.7634
3	1200000	2.6527	4.2594	7.3139	318.5386	8.0070	0.0000	3.6266	122.0000	4.7455
4	0	3.1861	4.9688	7.3139	99.7664	8.0070	0.0000	4.8785	118.0000	8.7627

Figure 2.15: Step 3

2.6 Machine learning algorithm for solving the task

I used a sequential model for the three models implemented, which is more appropriate for a pile of superficial dense layers.
I adapted it because our models have several entries.
Moreover, this model is multimodal due to the different types of our attributes: integer or float obtained through the pre-treatment phase.

I used Keras from TensorFlow to perform this experiment.

2.6.1 First experiment

The neural network is composed of an alternative of dense layers and dropout layers. The dropout layers permit to avoid the overfitting on the learning data. The dense layers connect each neuron of the defined layer to the neuron of the previous layer.

The loss is "MSE" (Mean squared error) for a quadratic error, and the optimizer used is "Adam."

We train the model over 100 epochs et we obtain a loss value around 20/30.

I also implemented another model working on the same principle: 30 epochs and three layers, which; the last one is a linear activation function. But, again, the results are the same.

First Model

```
Epoch 100/100
75/75 [==============================] - 0s 3ms/step - loss: 23.8808 - accuracy: 0.0000e+00 - val_loss: 21.7290 - val_accuracy: 0.0000e+00
```

Figure 2.16: Loss and accuracy

	id	original_title	popularity	popularity_predicted
0	3001	ディアルガVSパルキアVSダークライ	3.8515	2.8183
1	3002	Attack of the 50 Foot Woman	3.5598	4.2085
2	3003	Addicted to Love	8.0852	2.6045
3	3004	Incendies	8.5960	7.6022
4	3005	Inside Deep Throat	3.2177	0.5571
5	3006	SubUrbia	8.6793	4.0514
6	3007	Drei	4.8989	5.4300
7	3008	The Tigger Movie	7.0234	5.8173
8	3009	Becoming Jane	7.8297	6.7551
9	3010	Toy Story 2	17.5477	16.5054
10	3011	Cruel World	0.2624	3.5962
11	3012	Bande de filles	4.2203	4.5207
12	3013	The Gods Must Be Crazy	10.9735	5.7875
13	3014	Raising Victor Vargas	1.1787	1.7933
14	3015	The Brothers Bloom	7.9731	6.8127
15	3016	Beautiful Boy	2.1148	4.3460
16	3017	Hot Tub Time Machine	11.9677	7.0565
17	3018	Transcendence	9.7302	10.0890
18	3019	All That Jazz	5.6323	8.1964
19	3020	Titanic	26.8891	22.6493
20	3021	Very Bad Things	10.4323	6.7461
21	3022	My Best Friend's Girl	14.1141	7.4471
22	3023	Broken Bridges	1.1205	5.7611
23	3024	Suspect Zero	7.5509	4.3397
24	3025	The Adderall Diaries	4.1423	7.2769

Figure 2.17: Results

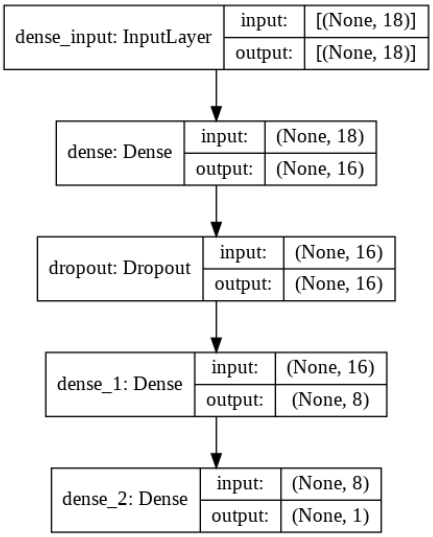

Figure 2.18: Layers

Second Model

```
Epoch 100/100
75/75 [==============================] - 0s 4ms/step - loss: 24.8019 - val_loss: 21.7725
```

Figure 2.19: Loss and accuracy

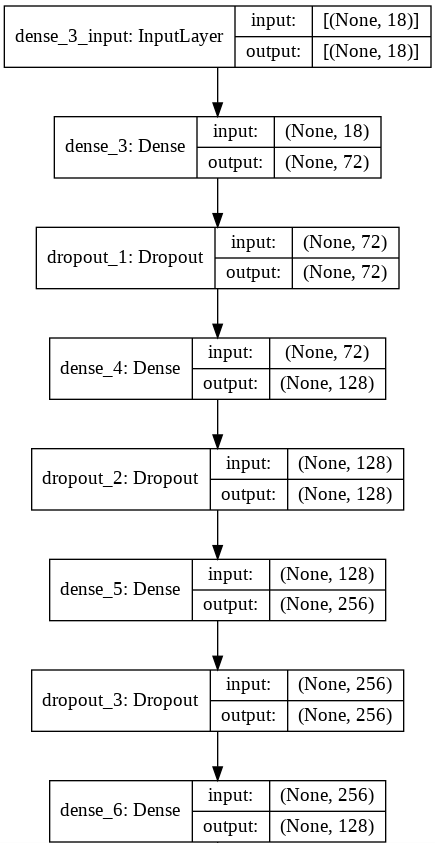

Figure 2.20: Layers

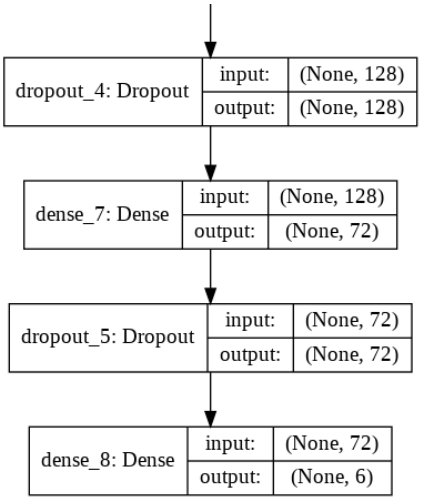

Figure 2.21: Layers

2.6.2 Second experiment

For the third model, I used seven dense layers with, for each, an activation function ReLu. Only the units number is changing, which means the layer output size. ReLu is essential for the convergence of our model.

The model is compiled with "mean square logarithmic error" as the loss function and with "SGD" (Stochastic gradient descent) as the optimizer. The model is trained on 100 epochs and allows us to obtain a loss a value around 0.23, which is much better!

Third Model

```
Epoch 100/100
75/75 [==============================] - 0s 6ms/step - loss: 0.2316 - val_loss: 0.3382
```

Figure 2.22: Loss and accuracy

	id	original_title	popularity	popularity_predicted
0	3001	ディアルガVSパルキアVSダークライ	3.8515	2.8183
1	3002	Attack of the 50 Foot Woman	3.5598	4.2085
2	3003	Addicted to Love	8.0852	2.6045
3	3004	Incendies	8.5960	7.6022
4	3005	Inside Deep Throat	3.2177	0.5571
5	3006	SubUrbia	8.6793	4.0514
6	3007	Drei	4.8989	5.4300
7	3008	The Tigger Movie	7.0234	5.8173
8	3009	Becoming Jane	7.8297	6.7551
9	3010	Toy Story 2	17.5477	16.5054
10	3011	Cruel World	0.2624	3.5962
11	3012	Bande de filles	4.2203	4.5207
12	3013	The Gods Must Be Crazy	10.9735	5.7875
13	3014	Raising Victor Vargas	1.1787	1.7933
14	3015	The Brothers Bloom	7.9731	6.8127
15	3016	Beautiful Boy	2.1148	4.3460
16	3017	Hot Tub Time Machine	11.9677	7.0565
17	3018	Transcendence	9.7302	10.0890
18	3019	All That Jazz	5.6323	8.1964
19	3020	Titanic	26.8891	22.6493
20	3021	Very Bad Things	10.4323	6.7461
21	3022	My Best Friend's Girl	14.1141	7.4471
22	3023	Broken Bridges	1.1205	5.7611
23	3024	Suspect Zero	7.5509	4.3397
24	3025	The Adderall Diaries	4.1423	7.2769

Figure 2.23: Results

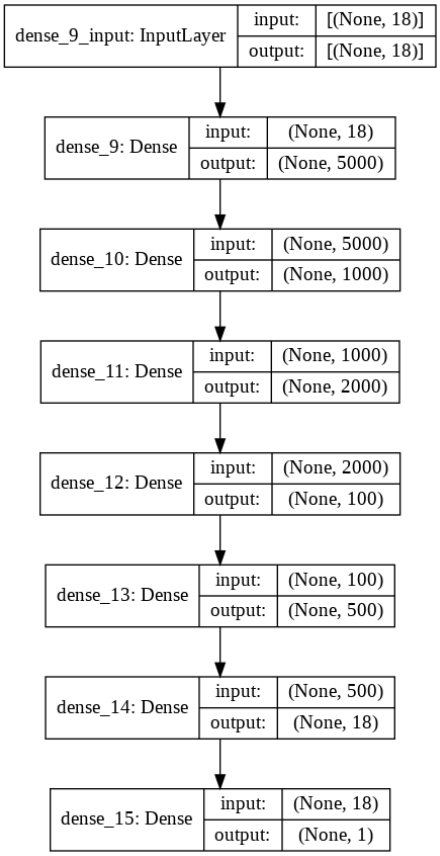

Figure 2.24: Layers

2.7 Comparison

To conclude, the second experiment is way more beneficial because the loss value is distinctly lower. The difference is due to the optimizer used, the loss, and the size of the output layers: for the third model, we were between 2000 and 5000, while for the first two, only 8 and 16.

On the other hand, this difference has no impact on the movies' prediction. In both experiments, the popularity predicted is around plus or minus two of the actual film's popularity.

This supervised learning allowed us to predict movies' popularity. The main difficulty was to select the proper parameters.

I could also have solved this problem with a classification model. However, in this case, it would have been necessary to predict a popularity class.

Chapter 3

Time analysis

3.1 Time depending on the number of processors/hosts

In order to perform this analysis, I took into consideration the last model, model three.
Indeed, it is the model with the lowest loss value.

My goal is to calculate the time to perform the training task with and without several workers.

In Keras, it is possible to distribute training across multiple machines.
In order to perform that, we use

```
tf.distribute.MultiWorkerMirroredStrategy
```

which implements a synchronous CPU/GPU multi-worker solution to work with Keras-style model building and training loop, using synchronous reduction of gradients across the replicas.
Additionallly, I added two more parameters in the

```
model.fit()
```

function:

- workers: Integer. Used for generator or keras.utils.Sequence input only. Maximum number of processes to spin up when using process-based threading. If unspecified, workers will default to 1.
- use_multiprocessing: Boolean. Used for generator or keras.utils.Sequence input only. If True, use process-based threading. If unspecified, use_multiprocessing will default to False.

With this new configuration, I ran several experiments. Following is a summary of my research.

	Single worker	Multiprocessing
Description	Same as before	I added several parameters as describe above
Function used	`model_3.fit( X_train_pp, y_train, epochs=100, batch_size=32, validation_split=0.2, callbacks=[cb])`	`model_3.fit( X_train_pp, y_train, epochs=100, batch_size=32, validation_split=0.2, callbacks=[cb], workers=8, use_multiprocessing= True)`
Time for training	54.683 s	130 s

I changed the number of workers in order to observe the behavior of the time for the training task.
I analyzed that the training time is increasing if we add more workers. Indeed for 8 workers, the time is 196s, while for 20 workers, it is 261s.
This result surprised me, I expected a lowest execution time with more workers.[1]

The following two graphs show the loss throughout the process.
We can notice that for the test set, the loss/epoch is better at the end for the single worker model.

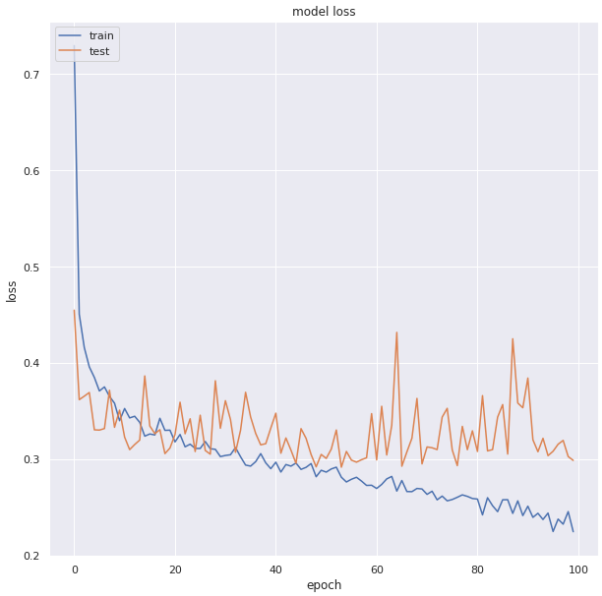

Figure 3.1: Single Worker

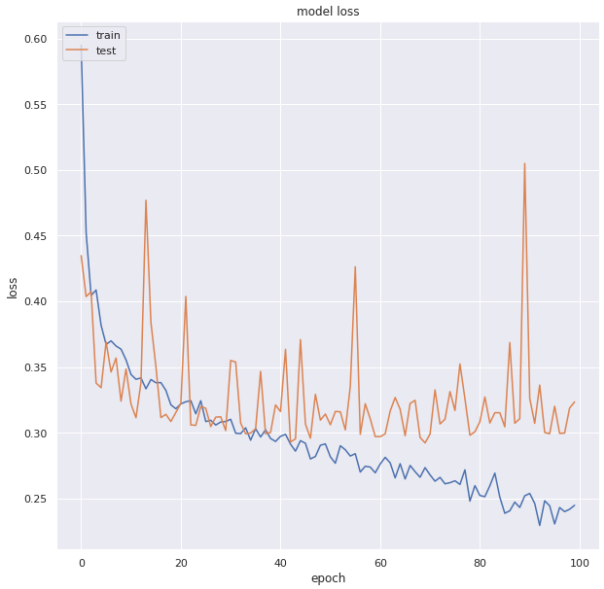

Figure 3.2: 4 workers

[1] I can provide a full demonstration of the algorithm.

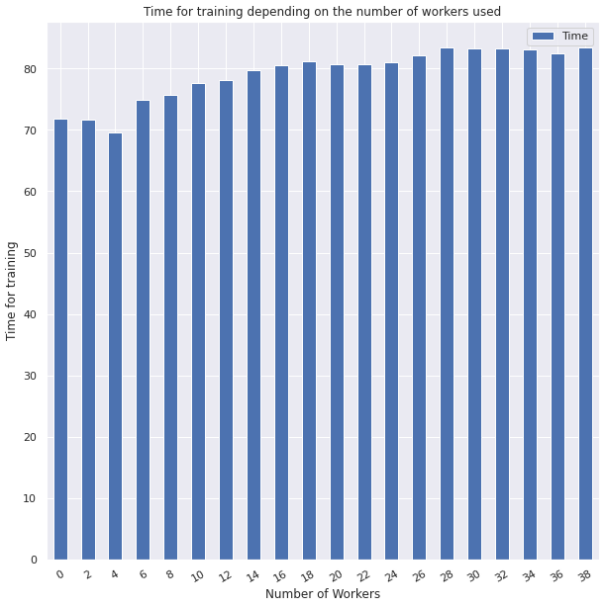

Figure 3.3: Time for training depending on the number of workers used

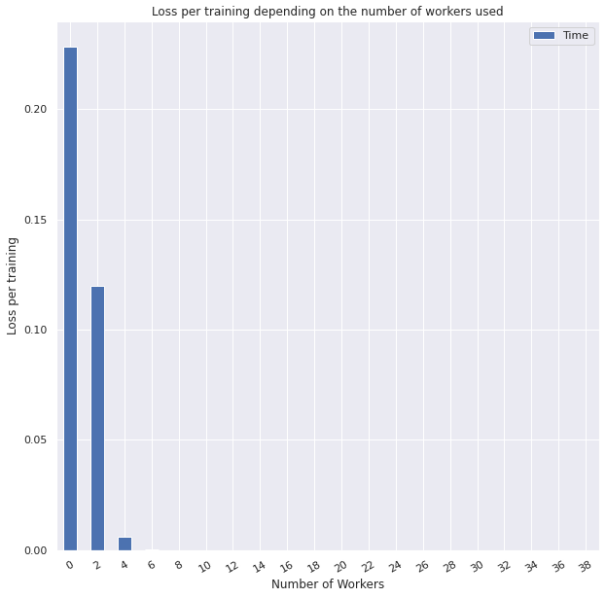

Figure 3.4: Loss per training depending on the number of workers used

3.2 Time depending on the number of data

As for the experiment based on the number of workers, I considered the third model.

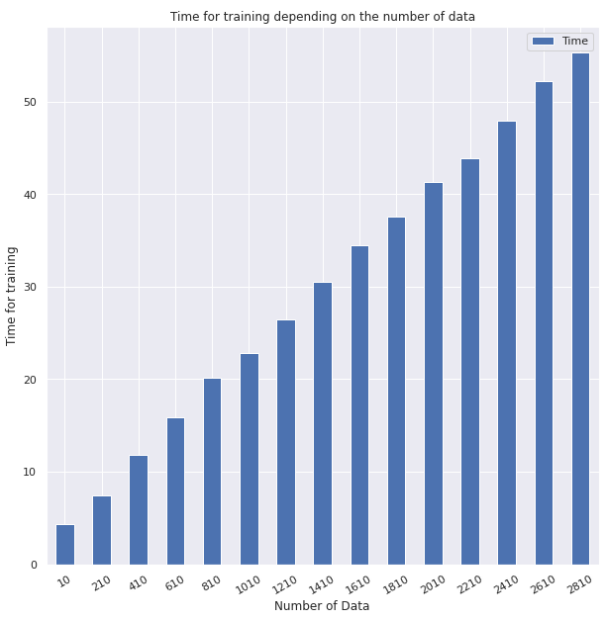

Figure 3.5: Time for training depending on the number of data used

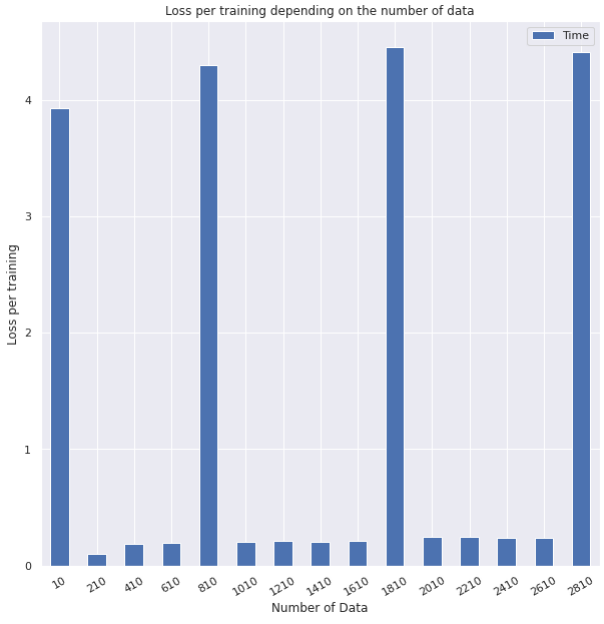

Figure 3.6: Loss per training depending on the number of data used

This two graphs show :

- the evolution of the time to train the model depending on the amount of data used

- the evolution of the loss

The initial amount of data is 3000. Let us consider only 1500 data.

With 1500 data, the training execution time is 30 s compared to 54.683 s with 3000 data.
I made some tests with a different amount of data. The conclusion is the following : the less data I have, the quicker is the execution of the training task.

Nevertheless, I wanted to analyze if this parameter has an impact on the prediction.
As shown in the picture below, the accuracy of the prediction was impacted.

	id	original_title	popularity	popularity_predicted
0	3001	ディアルガVSパルキアVSダークライ	3.8515	0.3077
1	3002	Attack of the 50 Foot Woman	3.5598	0.3375
2	3003	Addicted to Love	8.0852	0.3725
3	3004	Incendies	8.5960	0.3878
4	3005	Inside Deep Throat	3.2177	0.3013
5	3006	SubUrbia	8.6793	0.3494
6	3007	Drei	4.8989	0.3425
7	3008	The Tigger Movie	7.0234	0.3617
8	3009	Becoming Jane	7.8297	0.4029
9	3010	Toy Story 2	17.5477	4.0935
10	3011	Cruel World	0.2624	0.3098
11	3012	Bande de filles	4.2203	0.5442
12	3013	The Gods Must Be Crazy	10.9735	0.3664
13	3014	Raising Victor Vargas	1.1787	0.3319
14	3015	The Brothers Bloom	7.9731	0.3606
15	3016	Beautiful Boy	2.1148	0.3448
16	3017	Hot Tub Time Machine	11.9677	0.5365
17	3018	Transcendence	9.7302	4.3433
18	3019	All That Jazz	5.6323	0.4594
19	3020	Titanic	26.8891	21.6817
20	3021	Very Bad Things	10.4323	0.4134
21	3022	My Best Friend's Girl	14.1141	0.5376
22	3023	Broken Bridges	1.1205	0.3557

Figure 3.7: Prediction with 1500 data

Chapter 4

Conclusion

The experiments carried out have been conclusive.
Nevertheless, in the end, I cannot predict if a movie will be able to reimburse the cost of its production or if it will be the next icon of pop culture.

To interpret and use these results, we could perhaps predict the probability of a movie being the next prominent movie.