Nikoloz Chadunelli
Matthieu Schlienger
Paul Messéant
Paul Renoux

PTML Report

Python & Machine Learning Project

Part 1 : Supervised Learning

1.1) Choosing a dataset

To perform a supervised learning, we chose the MovieLens Dataset, which is a dataset of 25 million ratings and one million tags applied to 62 000 movies. This dataset was released in December 2019.

The whole data was split into 6 data-frames, each giving interesting and useful information.
After looking at what each data-frame contained, we merged some of them together in order to get the best information. This left us with a first dataframe having movie Ids, title and genres, a second one showing users Id with different tags and rating-scored they assigned to movies, and finally, a third one assessing, for each existing tag, the relevance it had in regards to the movie.

We used datas from each of those 3 dataframes and built a dataframe containing the movie id, its genres, the average rating given by users, the most commonly given tag, the most relevant tag, the number of ratings and finally the number of tags.

Our goal was to use that dataframe to train a model to predict ratings of movies.

1.2) Processing and analysis

The last thing we had to do, as our dataframe was almost ready, was to encode the “Genre” column, as the genres of a movie were listed as a single string, with each genre separated by a “|’. We split the different genres and made a column for each of the 20 different existing genres, each filled with either 1 or 0, according to whether the movie was of that genre or not.
Once that dataframe was completed and was meeting the requirements, we made a general analysis of the dataset.
We visualized the data to see if there were any outliers and after finding a few, we chose to remove them.
We also made a correlation matrix which showed that tags seemed to have very little impact on the rating, and the columns that had the strongest correlation with the ratings were the number of ratings and the number of tags.
We also studied the most occurring tags and it turned out that it was “based on a book” and “criterion” was the most relevant one.

1.3) Linear Regression

We chose to use a linear regression algorithm in order to try to predict ratings of movies.

1.4) Results

We split our data into a training dataset and a test dataset, and we then trained our model with the training dataset. Once the training was done, we used the test dataset to see if our model could make proper predictions. We then visualized the result of the prediction using a plot, which showed that our predicted ratings, although not too far from actual values, were still very inaccurate.

The scoring might be a bit too harsh with our predictions as the average rating has lots of decimal points.

After evaluating our algorithm, it turned out that our r^2 score was 0,3.
This means our regression was not very accurate, and the predictions were mostly off.

In order to fix those bad results, we tried several things:

- Tweaking the hyper-parameters to improve our score
- Removing the movies with either too few ratings or too few tags
- Removing the tags entirely, as they were given by Id and it might have complicated things for the algorithm.
- A grid search to find the right hyper-parameters

The grid search yielded even worse results. Since we were not able to correct our results, we tried to understand what was wrong, and came up with several explanations:

- Maybe we made a bad use of the dataset, and should have focused on user ids instead of movie ids, and tried to predict the ratings a given user would have given to a certain movie for example. So the axis we chose maybe wasn't the most relevant.

- Maybe the dataset was lacking data that would have helped the algorithm, because none of our attributes had a very high correlation with the ratings. For instance, we could have used things like the Box Office Income and Movie Budget which are much more likely to be correlated with the ratings. Sadly, we didn't have access to such data.

Part 2 : Unsupervised Learning

2.1) Choosing a dataset

We chose to set up an unsupervised learning model with a tweet dataset. This is the Sentiment140 dataset. It contains 1,600,000 tweets extracted using the twitter api. We used different ways to cluster tweet content.

It contains the following 6 fields:

1. target: the polarity of the tweet (0 = negative, 2 = neutral, 4 = positive)
2. ids: The id of the tweet (2087)
3. date: the date of the tweet (Sat May 16 23:58:44 UTC 2009)
4. flag: The query (lyx). If there is no query, then this value is NO_QUERY.
5. user: the user that tweeted
6. text: the text of the tweet

2.2) Processing and analysis

We did some preprocessing to change data to make it usable :

- We cleaned data by removing stop words and punctuation
- To make data exploitable we had to encode the actual tweet into numbers

2.3) Clustering

To cluster tweets, we used the KMeans algorithm because it is one of the most used and efficient. We trained the KMeans with the vectors returned by the TF-IDF and we assigned groups to the cluster variables. The Elbow method determines the most efficient number of clusters. PCA is a technique which reduces the dimensionality of a data set to an arbitrary number while preserving most of the information contained in it.

2.4) Results

As we can see, the clustering activity worked : the algorithm found groups. If we look at the top keywords of each cluster, there are lexical fields that appear. For example,

bed,days,still,night,day,early,ready,ta,going,work are routine's words. Some of them are almost similar because of their close pattern. Tweets are very random and common terms among some of these tweets which, when vectorized, obtain equal values for some dimensions.