UNIVERSITY OF PISA

Department of Computer Science

Top Spotify Podcast Episodes Dataset

Distributed Data Mining Project

Adam Dohojda
Wiktoria Stęczna
Elif Yilmaz

Academic year 2024/2025

Contents

1 Introduction and Data Understanding

Leveraging Spotify's Podcast Data: Market Intelligence for the Growing Audio Streaming Industry

The podcast industry has experienced remarkable growth, with global listenership reaching 546 million in 2024. In the United States, podcast penetration increased from 9% in 2008 to 47% in 2024, while advertising revenue attained 4 billion dollars[1]. Among major music streaming services competing for market dominance, Spotify has emerged as a particularly innovative player in this sector.

The Swedish streaming platform has captured 24% of the US podcast market share, positioning itself between YouTube (33%) and Apple Podcasts (12%). Spotify's strategic investments, including the integration of video content and high-profile content acquisitions such as the $100 million Joe Rogan contract, have solidified its position as a leading podcast platform. With 40 million US listeners and projections indicating growth to 42 million by 2025, Spotify's market presence continues to strengthen [2].

Given Spotify's significant market position and innovative approach, its data presents an optimal source for identifying industry patterns and understanding market dynamics, potentially informing strategic decisions in this expanding sector.

1.1 Overview of the Spotify dataset

The dataset we are analyzing consists of 29 features with a total of 228,000 observations, focusing on the top Spotify podcasts during the date range: September - October 2024, and it details ranks, genres, and countries. In the table 1.1, we can see the variables, their type and a short description.

Variable Name	Description	Variable Type
date	the date when the data was collected	date type
rank	the daily rank of a podcast in the top 200 list	numerical
region	the ISO code of the country	nominal
chartRankMove	the change in rankings compared to the previous day	nominal
episodeUri	spotify unique episode ID	nominal
showUri	spotify unique show ID	nominal
episodeName	podcast's episode name	nominal
description	episode description (some of these doesn't have any)	nominal
show.name	the name of the show	nominal
show.description	the description of the show	nominal
show.publisher	The publisher of the show	nominal
duration_ms	The episode length in milliseconds	numerical
explicit	whether or not the episode has explicit content(yes, no, unknown)	binary
is_externally_hosted	true if the episode is hosted outside of Spotify's CDN	binary
is_playable	true if the episode is playable in the given market, otherwise false	binary
language	The language used in the episode, identified by an ISO 639 code	nominal
languages	a list of the languages used in the episode, identified by their ISO 639-1 code	nominal
release_date	the date the episode was first released	date type
release_date_precision	the precision with which release_date value is known	ordinal
show.copyrights	copyright information for the show	nominal
show.explicit	whether or not the show contains explicit content	nominal
show.href	a URL linking to the show's Spotify page	nominal
show.html_description	HTML description of the show	nominal
show.is_externally_hosted	true if the show is hosted outside of Spotify's CDN	nominal
show.languages	true if the show is hosted outside of Spotify's CDN	nominal
show.media_type	media type of the show	nominal
show.total_episodes	total number of episodes that show has realized on the moment of snapshot	numerical
show.type	show type	nominal
show.uri	spotify unique show ID	nominal

Table 1.1: Variables description

This dataset consists mostly of nominal features with some different type features. This fact constrains us in our analysis, as this is much different than classic datasets that are arrays of numerical features. Twelve of these variables are not useful for our analyses or are duplicates of other columns. For now, we will consider them to gain an overall overview of the dataset, and later, we will determine how to handle them during the analyses conducted to address the research questions.

2 General Data Understanding

In this section we will investigate the dataset to understand its nature and plan future feature engineering steps. The dataset itself is raw API data, so we assume that the quality is very good but it may need a lot of feature engineering to be useful.

2.1 Quality of data: missing values and duplicates

We expect the dataset to be high quality, with date indexed data so we expect low number of missing values and no duplicates. The plot below shows fraction of missing values in TOP 10 most emtpy features.

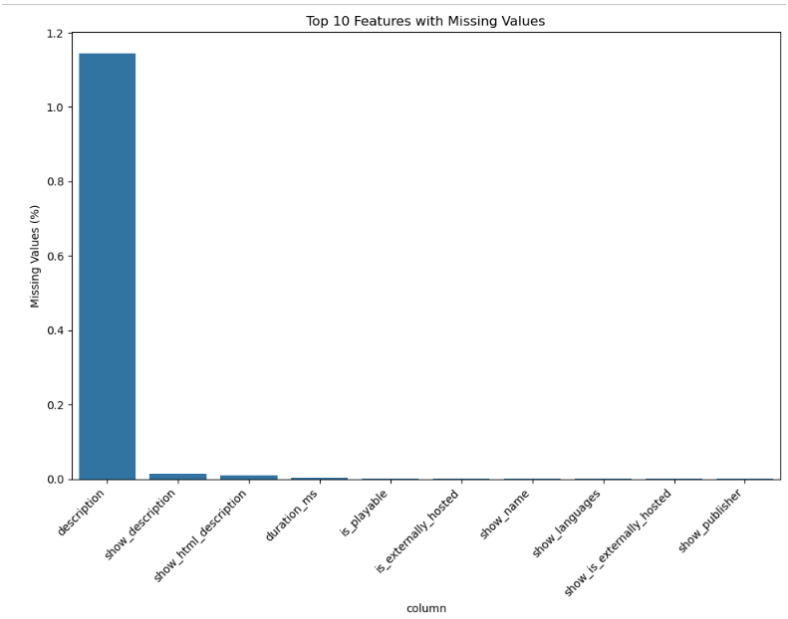


Figure 2.1: Missing values per feature

As seen we observed the most missing values in the `description` but still these observation accounted of less than 1.2% of the dataset. On the other note about data quality, we observed no duplicated rows in the dataset. These insights confirm our assumption about high quality of the data.

2.2 Visualization of distribution of selected features

The dataset contains equal number of records for each region which is visualized in Figure 2.2. This finding is important because we plan to perform analysis on features grouped by region.

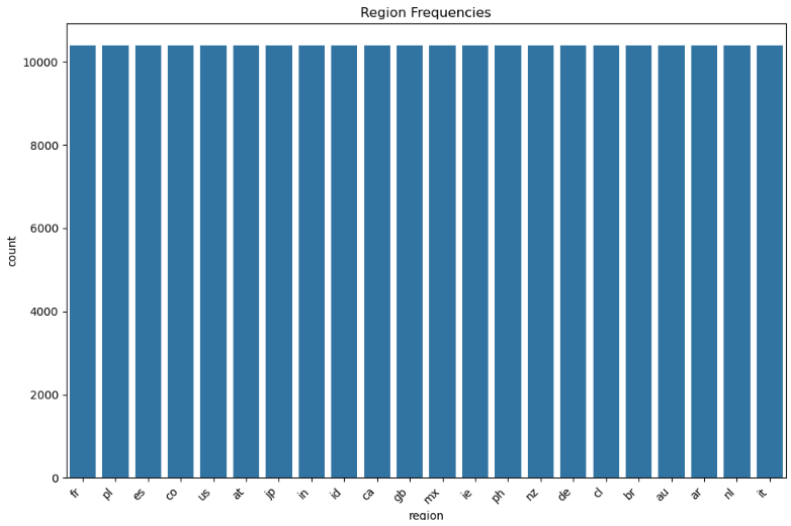


Figure 2.2: Distribution of Records Across Regions

We've also analyzed the durations of episodes as it might be useful in the upcoming tasks like classification or survival analysis. Mean duration of podcast was around 56 minutes. There were extreme cases of 5-second podcasts or 11-hour podcasts, but most podcasts had less than 10,000 seconds ($\sim$ 2h 48min).

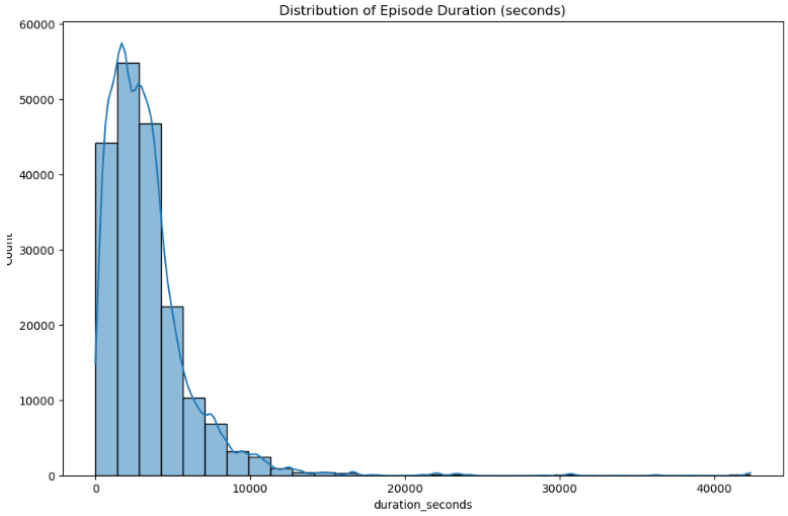


Figure 2.3: Episode Duration Distribution

The dataset also gave us insight into the most popular shows. With *The Joe Rogan Experience* having the most episodes and most days in the TOP 200 charts, plot 2.4 ensures us that the data quality is good, since these results are consistent with our domain knowledge.

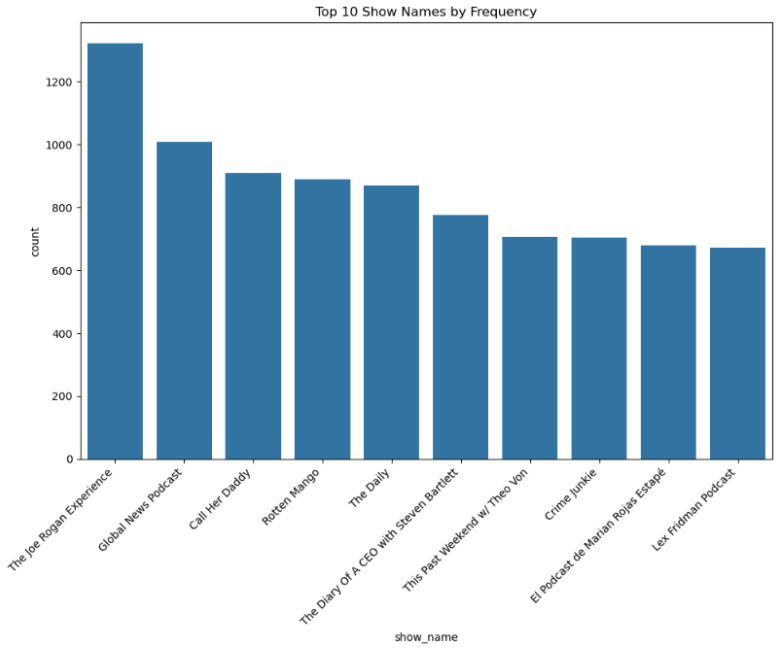


Figure 2.4: Most Popular Podcast Shows

2.3 Selecting useful features

We performed automatic feature analysis based on the type of the feature from the table 1.1 in order to select features that might be useful for our analyses.

Several features can be safely removed from our analysis without losing any valuable information. Some features contain only one value (like `show_type` is always "show"), such features are: `show_type`, `show_copyrights`, `release_date_precision`.

Some features were redundant (like `language` and `languages`), such pairs of features were: `language` and `languages`, `show_href` and `show_uri`, `show_html_description` and `show_description`. For these features we decided to just go with one of these features.

Since each task followed different pipeline we can't say we discarded all these features, these insights however were useful during planning phase of each activity. This part allowed us to understand that we are working with high quality dataset, that does not need a lot of missing values imputation. We understood how the features are distributed which in result allowed us to make informed decisions on what to do in the next steps. The examples of such decisions can be:

1. do not use `show_description`, use `description` for regions clusterization as it has more unique values
2. we can have very old podcasts in the dataset
3. podcasts vary in duration
4. each region has the same number of podcasts, so there won't be a problem with class imbalance

The other interesting facts will be investigated in the next phases. Now we will move on to our tasks.

3 Supervised Task

In the attempt of creating a model to predict whether a given episode will go up or down in the chart for the next day, we initially started with feature selection. We dropped columns that did not add any logical value, either because they were descriptive without added value (such as show URLs) or didn't have almost any variance (like `show_type`). We ended up dropping the following columns:

```
"episodeUri", "showUri", "show_uri", "description", "show_description",
"show_html_description", "show_href", "show_copyrights",
"show_is_externally_hosted", "is_externally_hosted", "is_playable",
"release_date_precision", "show_type"
```

This was done before the start of the process.

Preprocessing continued by targeting three columns related to languages. After confirming they contained the same information, we decided to keep only one column. Languages were accompanied by the abbreviation of their regions. To simplify the analysis, we dropped the region information from the language column and stored only languages in a separate column named `base_language`. This way, a language would not be redundantly separated into different regions (since we already have information on the regions in a separate column).

For the feature engineering part, to extract more meaningful information, we decided to add the following features:

- **released_since_date**: Calculating the difference between the current date and release date.
- **prev_rank**: Rank on the previous day.
- **rank_movement**: Our target column for ML models, showing if the podcast has moved up (`1`) or went down/stayed still (`0`) on a given day.
- **times_in_top_200_episode**: Per episode, counting how many times an episode appeared in the top 200.
- **total_times_in_top_200**: Per show, counting how many times a show appeared in the top 200 in total (independent of time).
- **avg_times_in_top_200_per_show**: Using `total_times_in_top_200` and the total episodes per show, calculating the average number of times a show stays in the top 200.

As needed, a `StringIndexer` and a `VectorAssembler` were applied to the dataset to enable ML model training.

3.1 Logistic Regression

We used Logistic Regression as a baseline model to evaluate how well our prepared dataset performed. Since this model is sensitive to feature magnitudes, `long` and `double` datatypes were scaled using a standard scaler. For our dataset, which was almost completely balanced, the results were slightly above average:

- **Accuracy**: 0.6556
- **F1-Score**: 0.6554
- **Precision**: 0.6558
- **Recall**: 0.6556
- **Area Under ROC (AUC)**: 0.7044

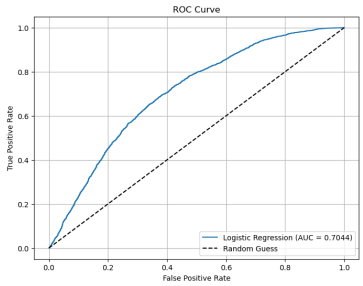


Figure 3.1: ROC Curve for Logistic Regression

3.2 Decision Tree

Our initial model without hyperparameter tuning performed as follows:

- **Accuracy**: 0.7469
- **F1-Score**: 0.7469
- **Precision**: 0.7469
- **Recall**: 0.7469

We also observed the feature importance according to this model, where 95% of the importance was attributed to `prev_rank` (0.82) and `times_in_top_200_episode` (0.13).

3.2.1 Hyperparameter Search

We implemented hyperparameter tuning using grid search combined with cross-validation (numFolds=5). The search ranges included:

- `maxDepth`: [3, 5, 10, 15]
- `maxBins`: [32, 64, 128]
- `minInstancesPerNode`: [1, 5, 10]
- `minInfoGain`: [0.0, 0.01, 0.1]

Best parameters:

```
maxDepth: 15
maxBins: 128
minInstancesPerNode: 5
minInfoGain: 0.0
```

Results with the best model:

- **Accuracy**: 0.7947
- **F1-Score**: 0.7947
- **Precision**: 0.7950
- **Recall**: 0.7947

As observed, the results improved slightly compared to the base Decision Tree. Another important observation was the change in feature importance: after using the best model, the importance of `prev_rank` decreased to 0.47, and many other columns gained importance compared to the base model.

3.3 Random Forest

The initial implementation of the Random Forest, with `numTrees`=50 and `maxDepth`=10, gave results similar to the Decision Tree after hyperparameter tuning.

- **Accuracy**: 0.7880
- **F1-Score**: 0.7947
- **Precision**: 0.7950
- **Recall**: 0.7947

However, after setting the `maxDepth`, `maxBins`, and `minInstancesPerNode` parameters to the values found optimal for Decision Tree grid search, our model's performance improved to:

- **Accuracy**: 0.8025
- **F1-Score**: 0.8024
- **Precision**: 0.8029
- **Recall**: 0.8025

This model performed the best so far, as expected since it is the most sophisticated model.

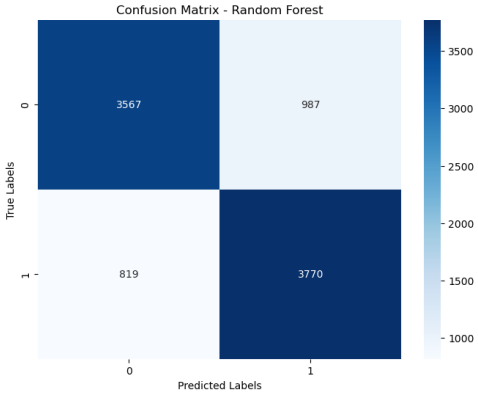


Figure 3.2: Confusion Matrix for Best Performing Model

Table 3.1: Comparison of Model Performance Metrics

Model	**Accuracy**	**F1-Score**	**Precision**	**Recall**
Logistic Regression	0.6556	0.6554	0.6558	0.6556
Decision Tree (Base)	0.7469	0.7469	0.7469	0.7469
Decision Tree (Tuned)	0.7947	0.7947	0.7950	0.7947
Random Forest (Base)	0.7880	0.7947	0.7950	0.7947
Random Forest (Tuned from DT)	0.8025	0.8024	0.8029	0.8025

4 Unsupervised Task

As our dataset contains mostly nominal features we decided to perform clustering based on the description of the episodes and shows. What we are interested in is establishing well defined clusters of regions based on podcasts' episodes descriptions and show descriptions. We know that there are 22 regions, each with equal number of episodes, so we expect to split those 22 regions into clusters based on podcast audience thematic preferences. To do this task we need to leverage PySpark NLP library: NLP Spark. The challenge was that we had 58 languages in the dataset, so we had to leverage multilanguage sentence embedding model. We decided to choose XLM-RoBERTa Base Sentence Embeddings model that was trained on over 100 languages. We had to use single model to compare different embeddings. We split the pipeline into two parts:

1. deriving lists of sentence embeddings for descriptions
2. averaging embeddings for descriptions → averaging by region → clustering based on these average embeddings

Details of these pipelines can be investigated on schemas below:

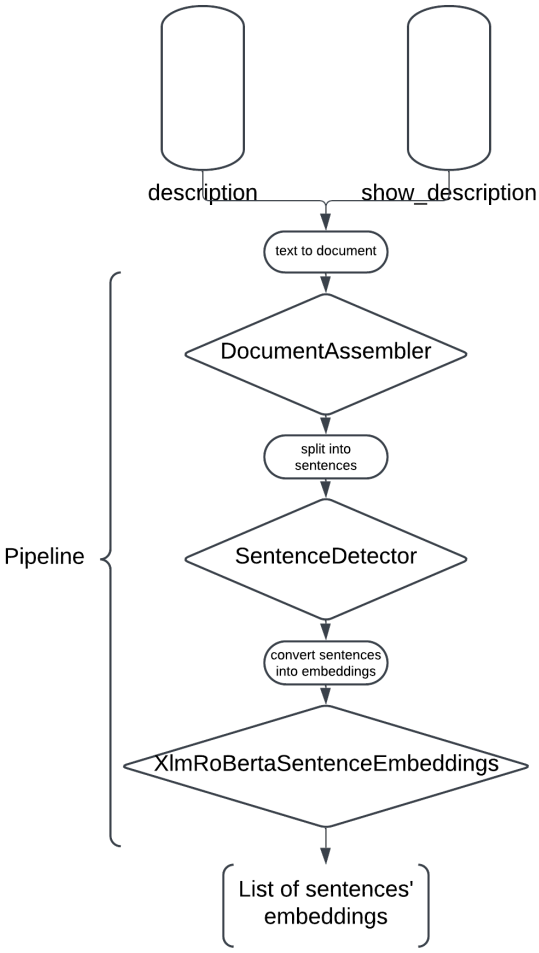


Figure 4.1: Pipeline Diagram 1

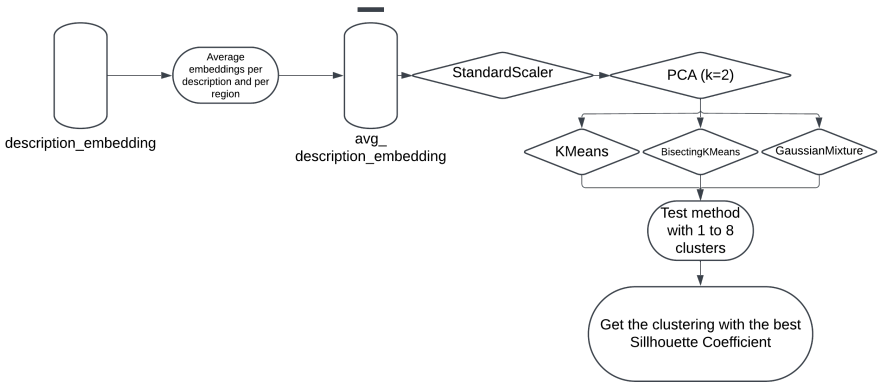


Figure 4.2: Pipeline Diagram 2

We decided to use only `description` column for clustering, as it had more unique values and we were limited by computation power. The Sillhouette Coefficients of various methods and with different clustering in decreasing order.

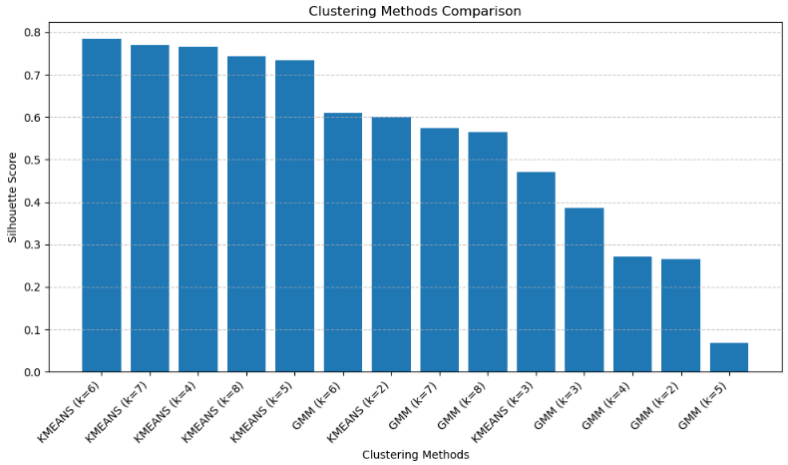


Figure 4.3: Sillhouette Scores comparisons

We've chosen the KMEANS with 6 clusters. This particular clustering received 0.7843 Silhouette Coefficient which is really high, with 1 meaning perfect clustering.

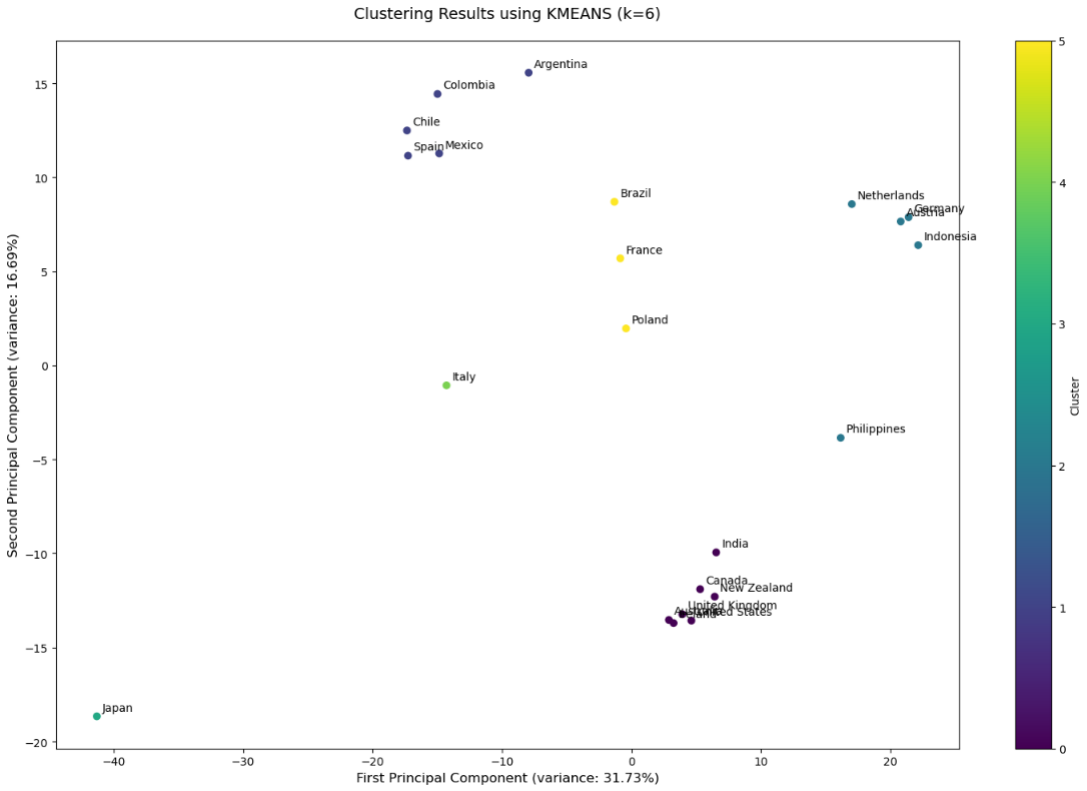


Figure 4.4: Effects of best clustering

We can see that clusters seem to make sense, proving that our clustering method can be sensible.

Clusters that place similar thematically podcasts in TOP 200 are:

- Anglosphere
- Spanish-speaking countries
- German speaking countries and Indonesia which is really interesting
- Japan, that is thematically away from all other countries
- Italy is seperated from all other countries
- Brazil, France and Poland

This pretty much makes sense to us. Similar culturally countries are together. There are of course some surprises like Indonesia with countries like Germany, Austria and Netherlands. However Japan distanced from all other countries makes perfect sense, when analysing this from cultural point of view. The question is if the clustering is effect of the semantic similarity in preferences or just based on language. Because Language Models are black-box models, we can't diagnose this problem with full explainability. To check it, we could try to cluster the descriptions, not based on the regions but based on shows. Unfortunately, due to limited time of the delivery of the project we wouldn't be able to finish computations before the deadline.

5 Survival Analysis

The goal of survival analysis for our dataset was finding the answer to 3 following questions:

1. How much time passes from release date of podcast episode to enter top200 (assuming it reaches top200)?
2. What is the probability that the podcast episode which entered to top200 will remain in that ranking for the 1st week, 2nd week and so on?
3. What are the factors that influence this survival probability, what causes podcast episode to remain in the ranking for a longer period of time?

To perform survival analysis we narrowed down the dataset to the following columns: `episodeUri`, `region`, `rank`, `chartRankMove`, `date`, `releasedate`, `durationms`, `showtotalepisodes`, `language`, `showmediatype`, `texplicit`, `isexternallyhosted`, `isplayable`. We basically got rid off the columns connected with podcast show itself - in survival analysis we focus on the specific episodes, not on the shows. Moreover, most of the show attributes are long string attributes, which are not very easy to handle.

In order to answer the first question we filtered podcast episodes throughout the regions to make sure that we do not take into consideration the same podcast episode twice. Then, we calculated the average number of days between release date and first date in top200 in each region.

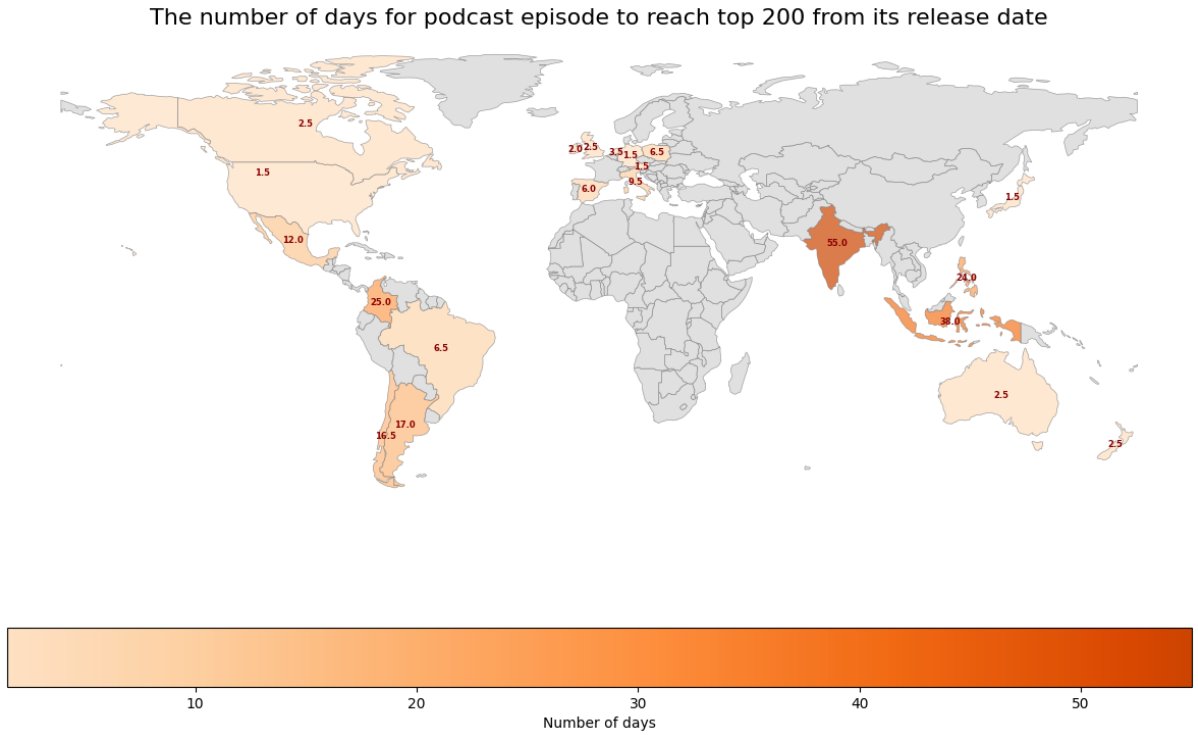


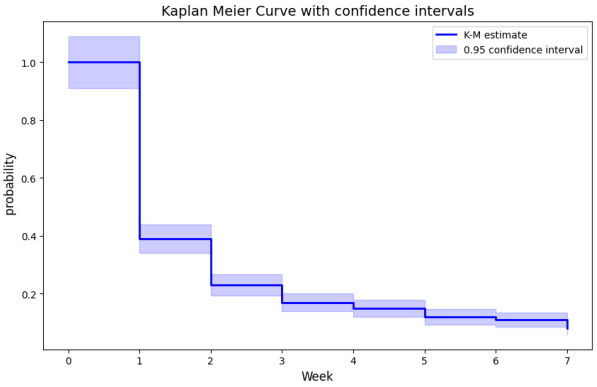
Figure 5.1: The number of days to reach top200

Country with the longest wait to enter top200 is India - which intuitively makes sense. This is the country with huge population, therefore reaching a top200 is a success that means a podcast episode has to be really good to reach that level.

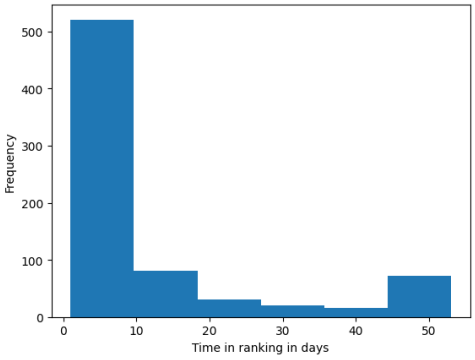
For the second question we decided to create Kaplan-Meier curve. A Kaplan-Meier curve is a statistical tool used to estimate the survival function from time-to-event data. In case of our dataset we would like to know what is the survival probability for a podcast episode to remain in the following week - 1st, 2nd and so on. In order to do that we had to do calculations step by step, due to the fact, that PySpark does not support calculating Kaplan Meier curve as quickly as lifelines library does.

To create Kaplan-Meier curve we need a life table. Life table is a useful tool to summarize the survival patterns over time. It includes attributes such N_t - number at risk for the beginning of a period, D_t - number of deaths during interval, q_t - proportion of dying during interval, p_t - proportion surviving interval and S_t - survival probability.

We narrowed down our dataset only to these podcast episodes, that entered the ranking at the beginning of observed period which is 02.09.2024 - so their first date in ranking has to be like that and also their chartRankMove has to be NEW. Thanks to that our results are more interpretable, otherwise we would take into consideration episodes that were at 'the end' of their time in chart.



(a) Kaplan-Meier curve



(b) Time in ranking - distribution

From the Kaplan - Meier curve on the left we can see that the probability that podcast episode will remain in top200 during its first week is 39%. Later on, this probability decrease to 8% in the 7th week.It means that a small fraction of podcast episode can stay in top200 for such a long time.

On the right side we have a histogram plotting the time in ranking in the dataset. The plot looks like gamma distribution with low shape parameter - the most of the podcast episodes drop out of the ranking under 10 days of their 'life' in it.

To obtain the answer for the third question we created an AFT model. The Accelerated Failure Time model is a survival analysis technique used to predict the time until an event occurs. It can also help identify the covariates that contribute to calculated survival probability. PySpark supports AFT model calculation with AFTSurvivalRegression class and computes AFT model according to the Weibull distribution. In order to compute the model we had to use OneHotEncoder for `language` and `region` columns to make it suitable format to VectorAssembler. Independent variables to AFT PySpark model are required in assembled format.

Our AFT model has 55 coefficients due to onehotencoding. The most interesting coefficients are:

- -0.005227821454609564 for `rank` means that unit increase in rank indicates that the

average survival time will be reduced by factor of 0.99 (exponent of coefficient). It is important to remember that the lower the rank value, the higher the position of podcast episode in the ranking. Therefore, the interpretation of the coefficient makes sense - the lower the rank of the episode, the lower survival probability.

- 6.599901366224987e-08 for `durationms` means that unit increase in duration of the podcast episode indicates that the average survival time will be increased by factor of 1 - time doesn't change it's survival chances in ranking.

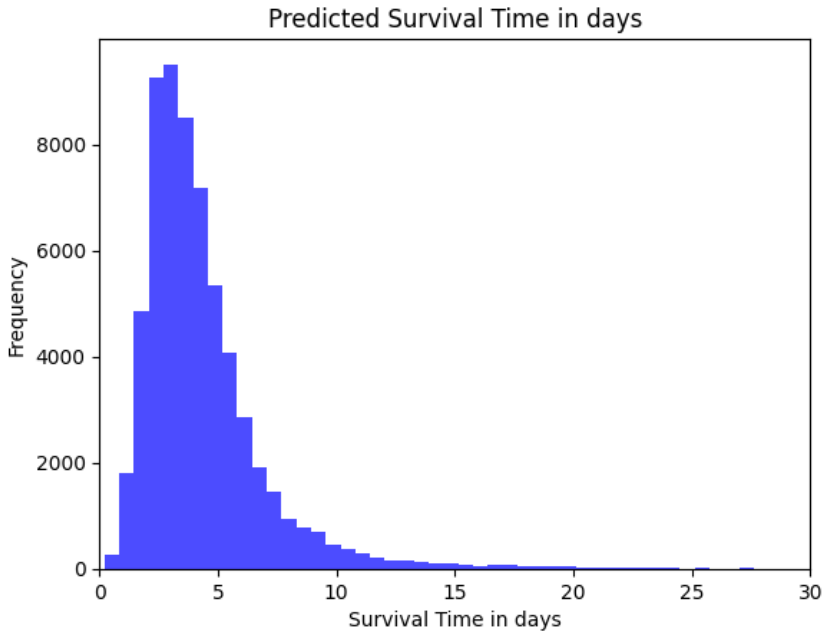


Figure 5.3: Histogram of predicted survival days obtained from AFT model

Figure 5.3 shows distribution of survival time for studied podcast episodes. A we can see the most podcasts survive less than 5 days in the ranking, with mean survival time equal to $\sim$ 4 days.

To sum up this part, the dataset contains crucial information to detect and assess time dependent patterns. We can see that the average time of the podcast episode release to reach top200 in most countries is not very long, around 3-5 days to get there. Also the time in the ranking for the podcast episode is quite short - around 5 days. Only small fraction of podcast episode can remain in top200 for a long time - almost 40% of podcast episodes will drop out of the top200 in their first week in the ranking.

The survival analysis helped us understanding the fact that being in top200 podcast in the region is a hard achievement. It is also in most cases a short-term success and the probability is not in the favour of podcast episode.

Bibliography

[1] *Podcast Statistics You Need To Know.* Backlinko. https://backlinko.com/podcast-stats

[2] Statista Research Department. (2024). *Apple Podcasts vs. Spotify podcast listener numbers in the United States from 2022 to 2025.* Statista https://www.statista.com/statistics/1303252/apple-spotify-podcast-listeners-united-states/

[3] Wayne W. LaMorte, MD, PhD, MPH (2016). *Estimating the Survival Function.* Boston University School of Public Health https://sphweb.bumc.bu.edu/otlt/mph-modules/bs/bs704_survival/bs704_survival4.html