MSIN0097: Predictive Analytics

Group Assignment

Wenhao Jiao
Pavlos Michaelides
Symeon Kokovidis
Katharina Wiedmann
Andreas Mourou
Tammy Michaeli

Word count: 1949

14th April 2020

Table of contents

N.B. Codes can be accessed from **Faculty** ("PA Group Coursework – Group D")

Table of figures

Introduction – Business Understanding

The emergence of music streaming has revolutionized the music industry; instead of promoting CDs and records, artists and publishers are shifting their focus to music streaming platforms such as Spotify (Yang et al., 2016), which incentivizes businesses to shift to a more data-driven operation. The Warner Brothers Group is one of the leading corporations in music entertainment, and their library of contents increases every year, making them one of the most valuable music conglomerates in the world. In this report, we aim to help WMG to predict whether an artist will succeed in 2017 based on streams from Spotify.

Theoretical Framework

Based on our existing knowledge from our individual projects we decided to follow CRISP-DM (Cross Industry Standard Process for Data Mining) framework (Wirth et al., 2000), conducting a peer review for our previous work and re-defining our business and data understanding. The goal was to synthesize but also extend our existing work for the problem.

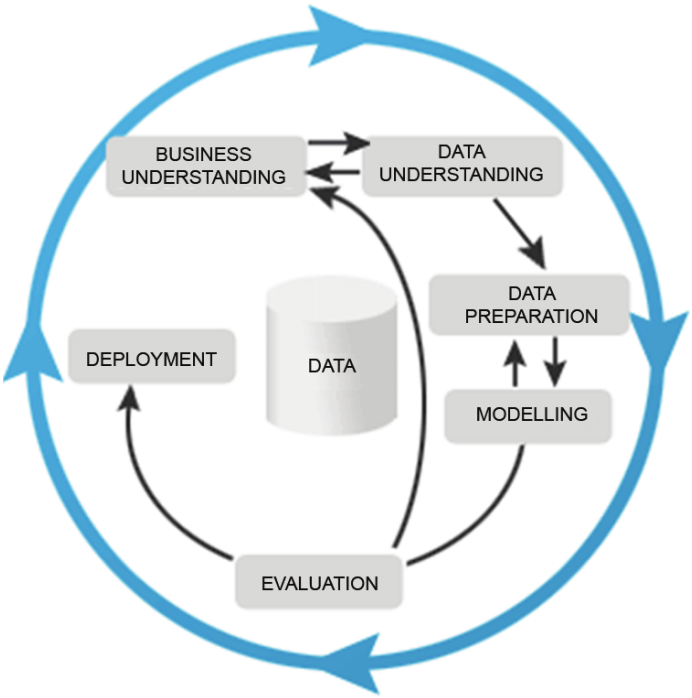

Figure 1: Data mining phases according to CRISP-DM (Wirth et al., 2000)

According to Raschka et al. (2020), there are five main steps of Machine Learning process (see Figure 2). In this report, these steps are represented by the stages of Data Preparation, Modelling and Evaluation according to CRISP-DM.

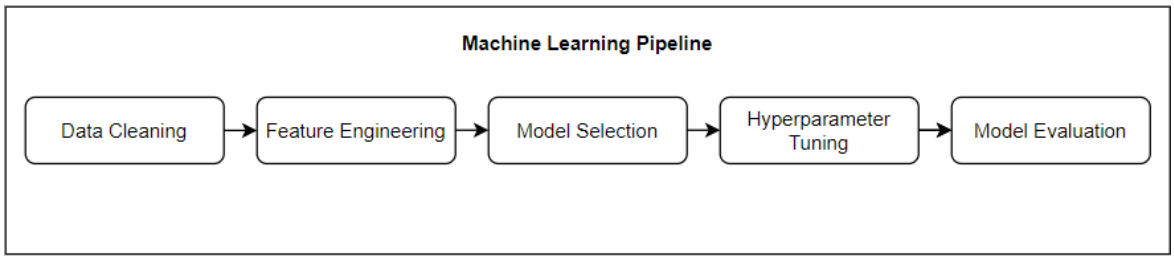

Figure 2: Machine Learning Pipeline

Data Understanding

Data Problem Formation

To model the problem, we defined a binary variable to classify successful artists: 1 denotes artists who have been featured on one of the four key playlists (Hot Hits UK, Massive Dance Hits, The Indie List, New Music Friday), and 0 denotes otherwise. We excluded the artists who were successful from 2014 to 2016 (see Figure 3). Moreover, we decided to keep transactions only before 1st Oct 2016 and leave a 3-month window period as these three months' transactions are timely close to the actual response (i.e. transactions in 2017), thus minimizing the potential biases of our prediction model.

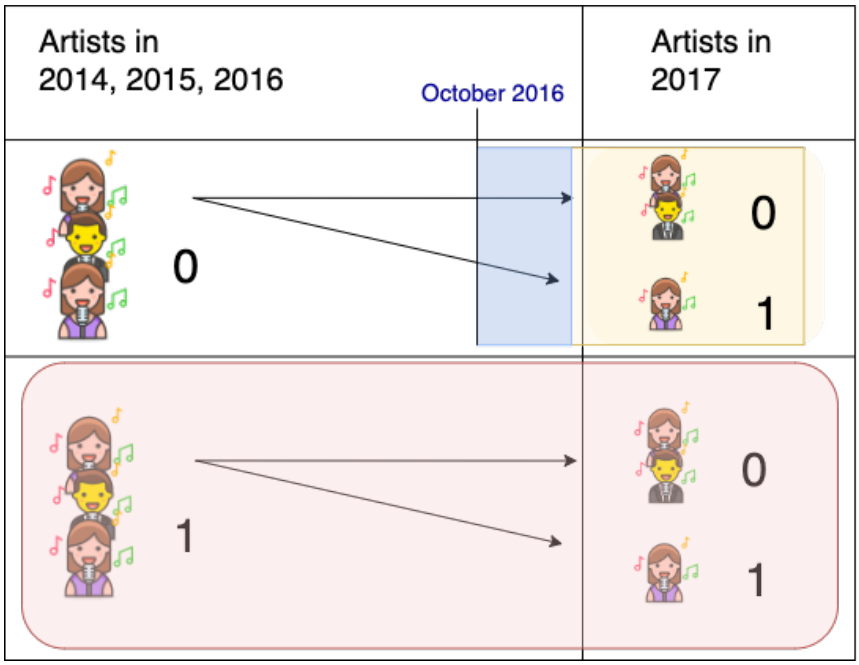

Figure 3: Problem formation

Exploratory Analysis

In our group project, we have decided to examine carefully the response (successful artist in 2017) but also its relationship with other attributes. In figure 4 we have the new successful artists per different date in 2017. Please note that in our dataset we have streams only from the 10th day of each month.

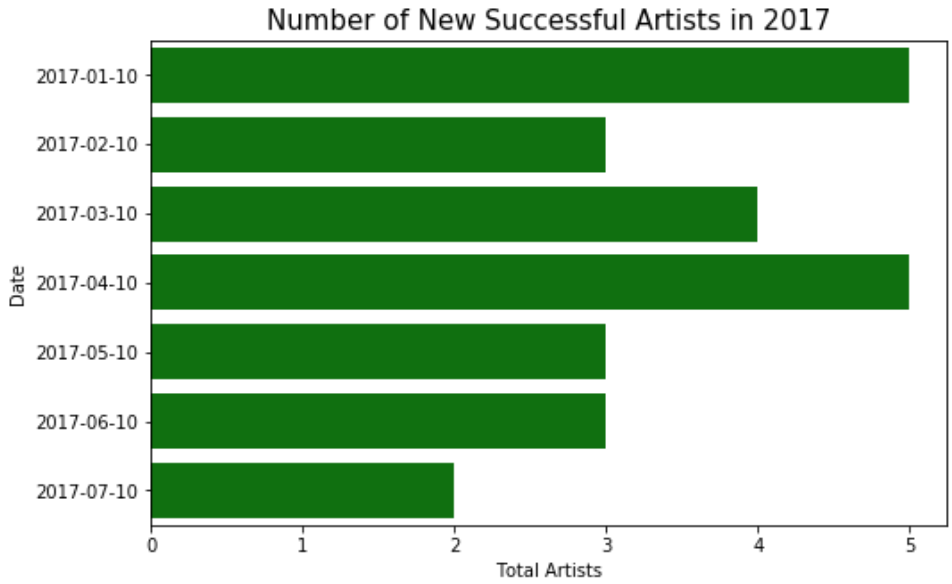

Figure 4: Number of New Successful Artists per date in 2017

From our individual work, we have seen that the gender, the age and the region of the streamers play a pivotal role in the success of an artist. In our exploratory analysis, we developed visualizations which compare the streamers of successful and unsuccessful artists. In Figure 5 we can see that successful artists have progressively more streams closer to 2017. Note that the total number of streams for successful and unsuccessful artists are different.

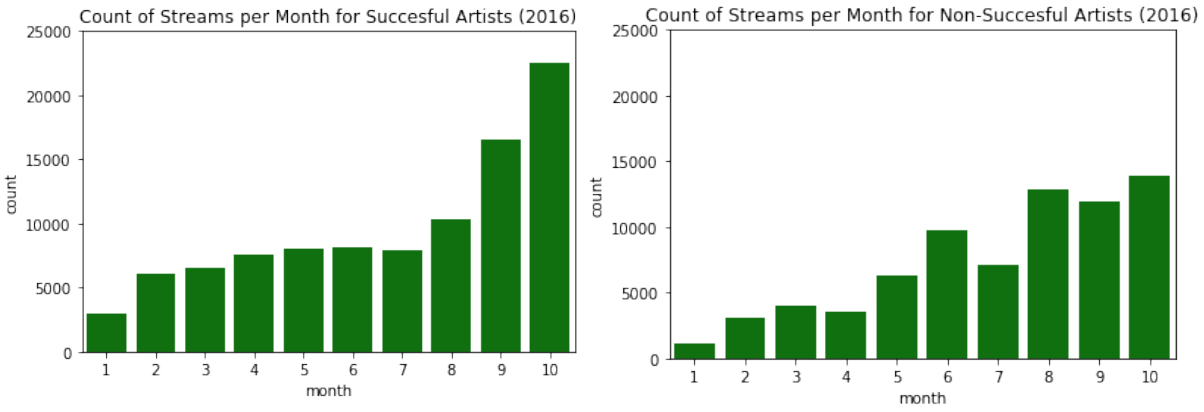

Figure 5: Count of Streams per Month for Successful and Unsuccessful Artists (2016)

For the Age Distribution, we notice that in both cases the median value is equal to 26, where the mean for successful artists is lower than the one for unsuccessful artists.

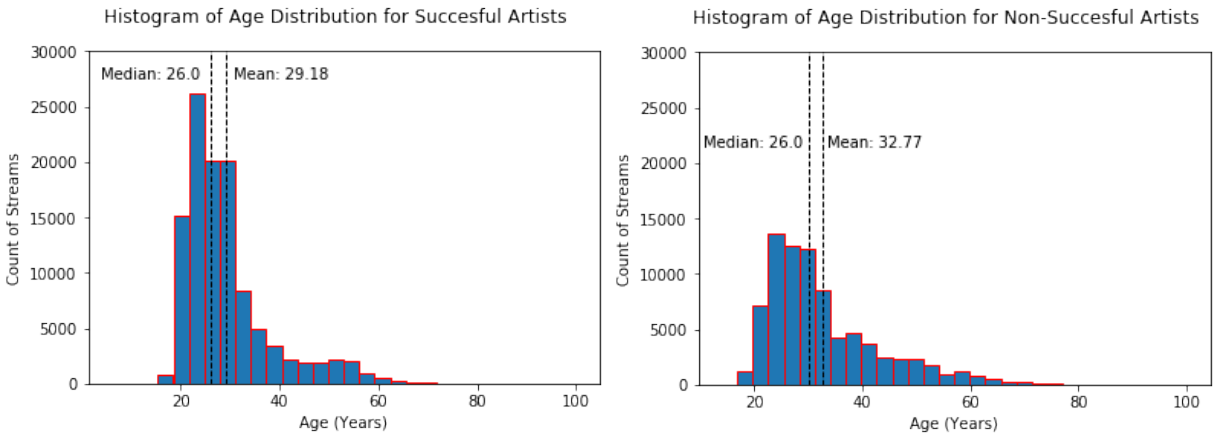

Figure 6: Age Distribution of streamers for Successful and Non-Successful Artists

We can see that their Inter Quantile Range differs. Successful artists tend to have more young followers. Besides, the range is bigger for unsuccessful artists.

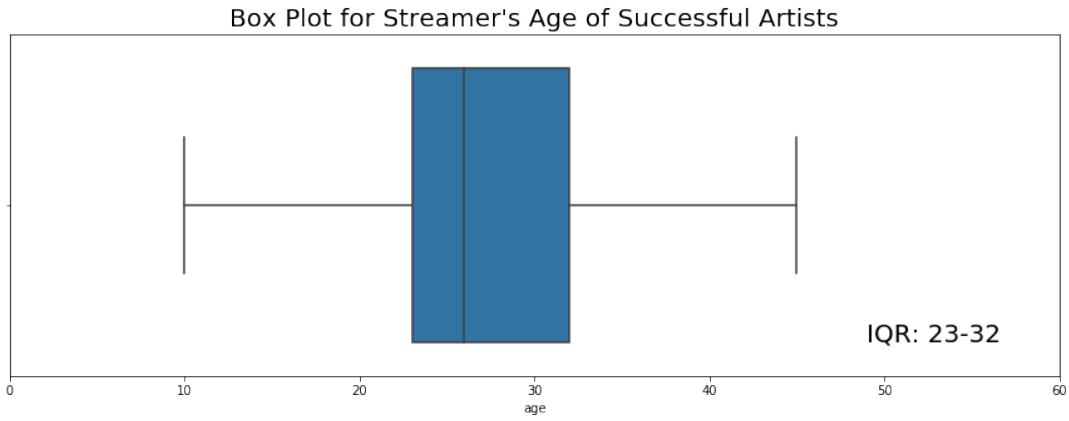

Figure 7: Box Plot for Streamer's Age of Successful Artists

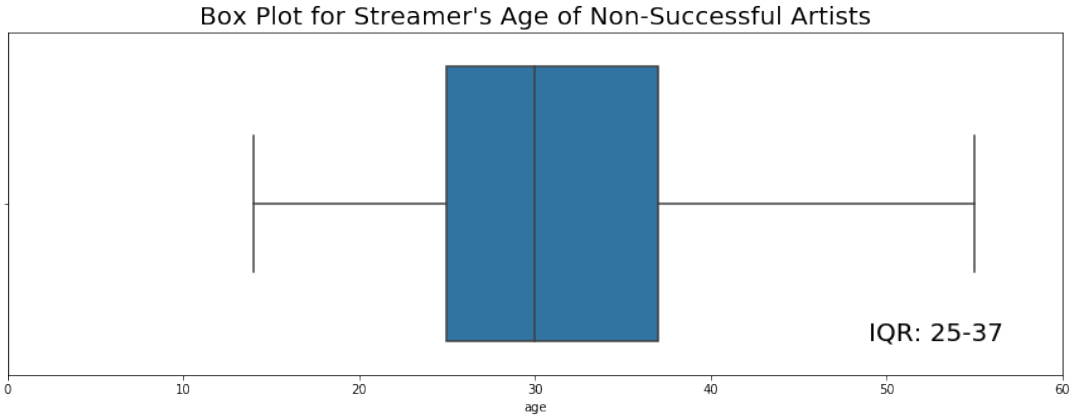

Figure 8: Box Plot for Streamer's Age of Non-Successful Artists

To further extend our analysis, we created different age bins. We notice that young listeners (20-25) listen more to successful artists (compared to other age groups).

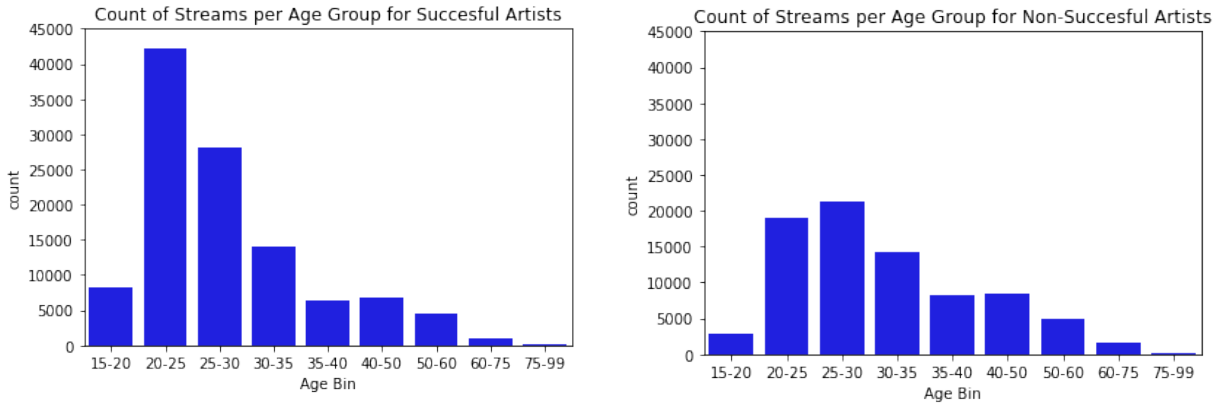

Figure 9: Count of Streams per different Age Groups for Successful and Unsuccessful artists

Regarding gender, we see that females have a slightly higher percentage (4%) in successful artists' userbase.

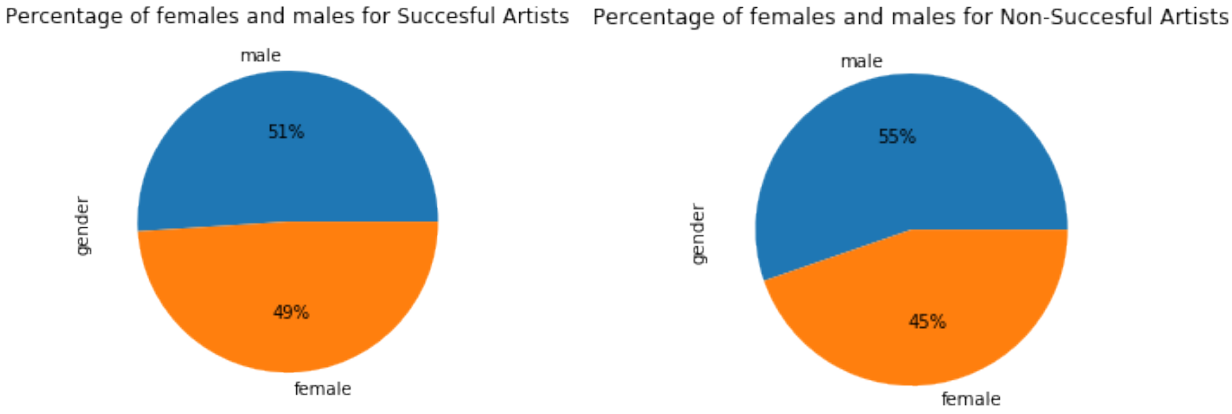

Figure 10: Percentage of Female and Males for Successful and Unsuccessful Artists

Regarding young people of different gender, young females seem to listen more to successful artists compared to males. Besides, male streamers who are 30 years old are substantially higher for both successful and unsuccessful artists.

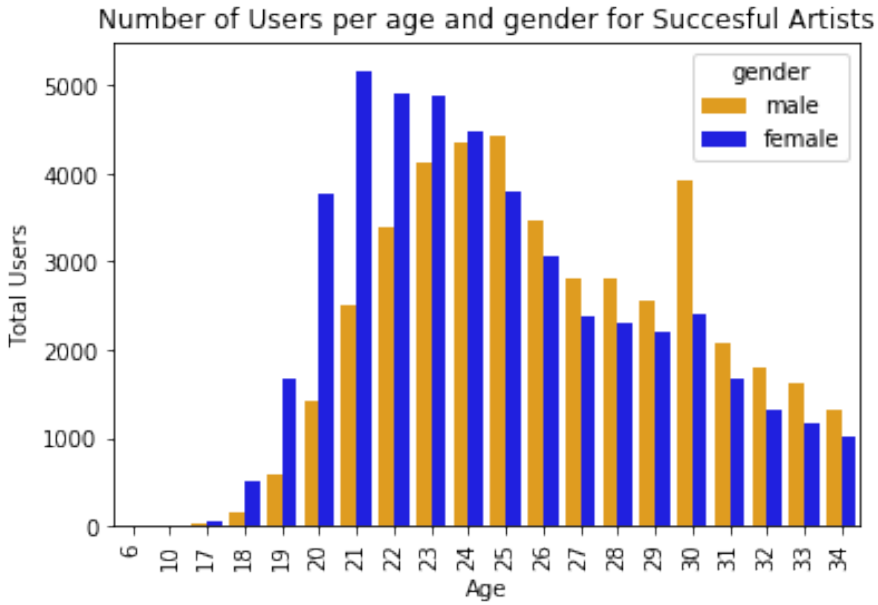

Figure 11: Number of Streamers per Age and Gender for Successful Artists

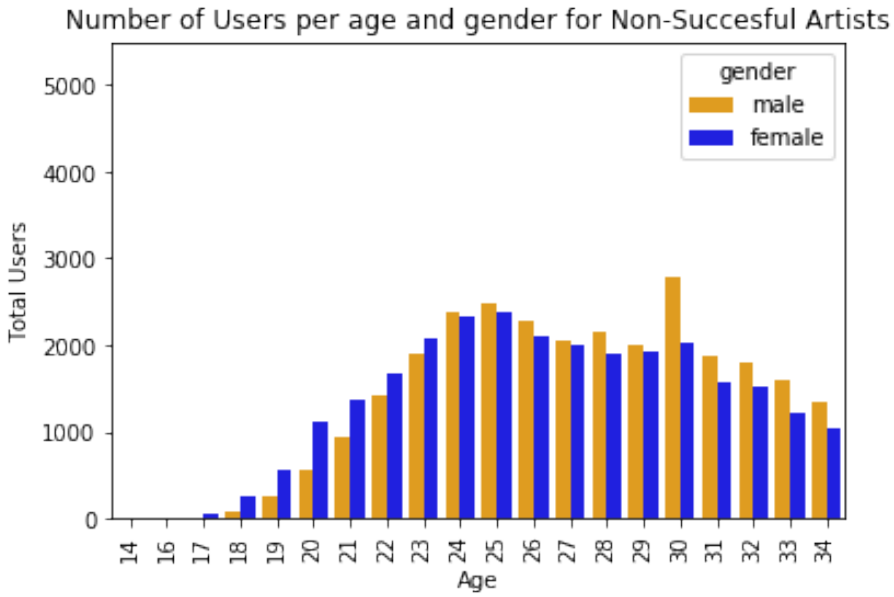

Figure 12: Number of Streamers per Age and Gender for Unsuccessful Artists

Regarding the location of streamers, successful artists have around 18% of their streams from London; for unsuccessful artists, this percentage equals to 23%.

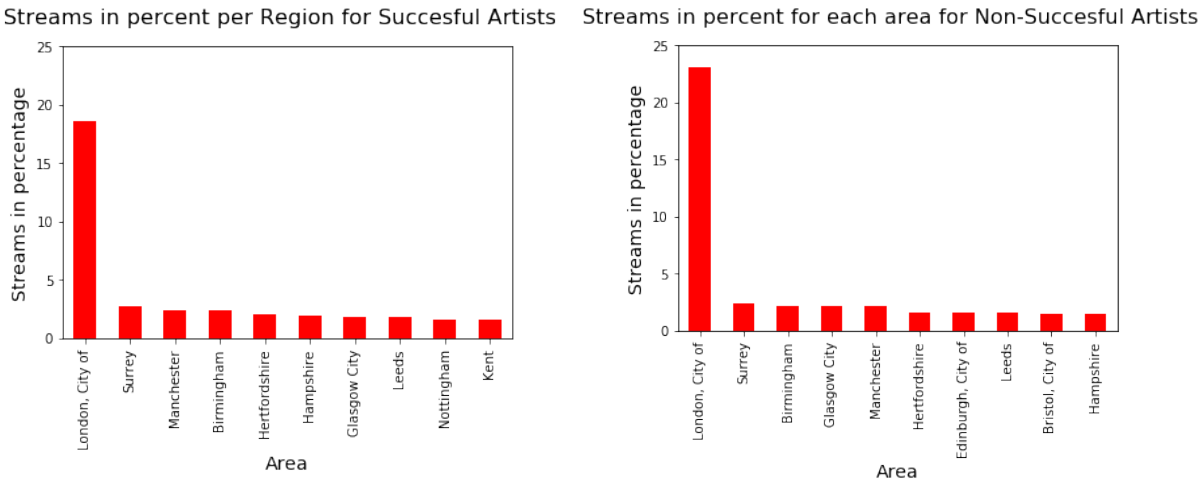

Figure 13: Percentage of streams per region for successful artists

Finally, we created a map that illustrates the percentage of successful streams by region in the UK (By region: $\frac{Streams\ from\ succesful\ artist}{Total\ \ streams}$). What we observe is a clear distinction between regions. For example, London has the highest percentage of successful streams, which implies that the region could be an important feature in predicting success.

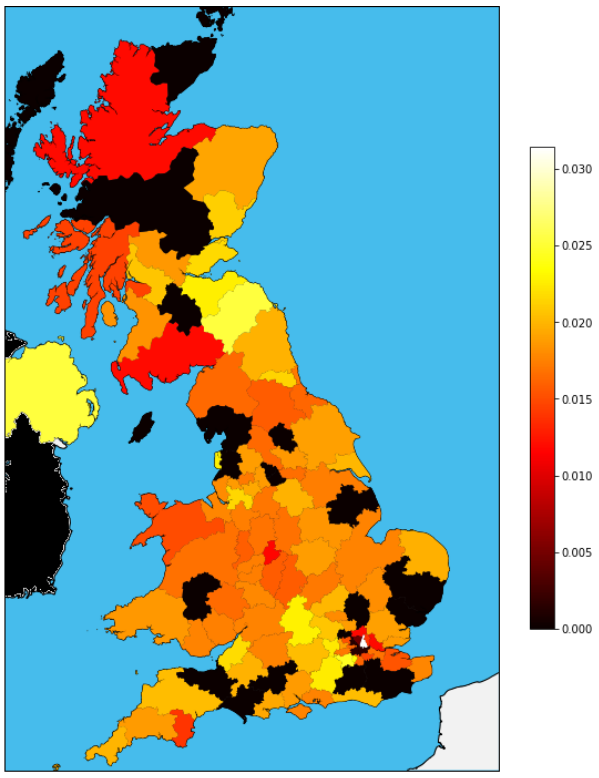

Figure 14: Regional Percentage of Successful Streams

Feature Engineering

Artist Features

We first calculated the passion score (Stream Count/ Users per artist) for each artist. Furthermore, we created a feature ("art_streamlength_ratio_mean"), which captures the average time listened (out of full length) for all the songs of an artist (See Appendix A).

artist_name	stream_count	users_per_artist	passion_score	art_streamlength_ratio_mean
The Hunna	25746	13933	1.848	0.866
Coasts	23168	17232	1.344	0.824
RAT BOY	11005	9035	1.218	0.806
dvsn	10018	7661	1.308	0.737
A R I Z O N A	9874	9040	1.092	0.804

Figure 15: Artist features

Playlist features

For the playlist features, we kept the top 20 playlists (in terms of streams) that each artist has featured on before eventually ending up on one of the top 4 key playlists. For each of these playlists we calculated the (a) Playlist Stream Counts; (b) Playlist Unique Users (Reach); (c) Prior Playlist Passion Score. As we did this on a per artist-playlist basis, we aggregated these metrics by taking the average of each artist.

	artist_name	playlist_stream_count	users_per_playlist	playlist_passion_score
0	99 Percent	11.417	10.417	1.013
1	A Boogie Wit Da Hoodie	252.538	241.846	1.139
2	A R I Z O N A	414.700	387.100	1.094
3	AGWA	1.000	1.000	1.000
4	Adan Carmona	11.000	7.000	1.571

Figure 16: Playlist Features

User features

In terms of user features, we decided to explore the gender percentage breakdown by creating a male to total streamers ratio (Figure 17).

	artist_name	MaleRatio
0	99 Percent	31.400
1	A Boogie Wit Da Hoodie	77.900
2	A R I Z O N A	49.000
3	AGWA	100.000
4	ALMA	100.000

Figure 17: User feature: Male Ratio

API

To enrich our feature-frame, we decided to include information obtained through the Spotify API (***Spotipy***[1]**,** Appendix B). We obtained the average track characteristics at the artist level (Figure 18), which were merged with the basis feature-frame.

artist	danceability	energy	key	loudness	mode	speechiness	acousticness	instrumentalness	liveness	valence	tempo
99 Percent	0.702933	0.785288	6.105769	-5.310769	0.451923	0.113213	0.061183	0.006985	0.191172	0.514827	121.102625
A Boogie Wit Da Hoodie	0.691098	0.603951	5.460784	-6.666588	0.441176	0.250470	0.284661	0.000007	0.128052	0.470767	125.335000
A R I Z O N A	0.673333	0.658667	3.190476	-6.950095	0.904762	0.050443	0.148927	0.087732	0.137614	0.356848	112.628381
AGWA	0.729000	0.412875	3.375000	-9.187375	0.750000	0.154537	0.461350	0.000016	0.164875	0.637375	118.620875
ALMA	0.532050	0.502528	6.299652	-7.526021	0.960511	0.080033	0.619858	0.005185	0.188826	0.868913	155.439139

Figure 18: Features from Spotipy API

Dealing with multi-collinearity

Decision trees are immune to multicollinearity. For example, if you have two features which are 99% correlated, only one of them will be chosen when deciding upon a split the tree (Piramuthu et al., 2008). Our approach uses a boosted tree algorithm, hence, rectifies potential correlation issues between the features.

Age and Region features using Principal Component Analysis

Based on our previous work, we have seen that both the age and location of the streamers have a major impact on predicting the success of an artist. Thus, we decided to create five different feature-sets based on them (Appendix C). For some of them, we have applied a Principal Component Analysis (PCA), an algorithm first defined by Person (1901), to reduce their size (columns). According to Abdi and Williams (2010), it is a multivariate technique that can extract important information from a data table and represent as a set of new orthogonal variables.

These five feature-sets have been joined independently with the basis feature-frame. In the next step we are going to present their predictive ability.

[1] https://spotipy.readthedocs.io/en/2.9.0/#

Model Selection

With these five feature-sets, we created different train-test subsets. Also, we used SMOTE, which over-samples the minority class by generating random "synthetic" examples (Chawla et al., 2002).

To determine which of the 5 different datasets had the most predictive power for predicting the successful artists, we used XGBoost, a scalable machine learning system for tree boosting (Chen and Guestrin, 2016, Appendix D).

We have created a universal train/test split before applying any algorithm (Min-Max Scaler, PCA, SMOTE, XGBoost) to the 5 different datasets, thus fitting all the different algorithms only on the train set and then transformed/predicted on the test and train set separately. This way we can use train and test sets in the same space, without including any information of the test data inside the fitted algorithms; otherwise, it cannot be considered as unseen data and bias would be introduced. The results of the different scenarios are presented in figure 19.

Dataset	AUC
1. Bins Counts	0.830
2. Bins Percentages	0.860
3. All Ages Counts	0.789
4. All Ages Percentages	0.753
5. Region - Age Bins Counts	0.749

Figure 19: Comparison of the five feature-frames after using XGBoost

When looking at the AUC scores, which is a measure for evaluating the predictive ability of learning algorithms (Huang and Ling, 2005), we can see that the feature-set with the Age Bins percentage (No PCA) and region code PCA performs best.

Hyper-parameter Tuning

For our final model, we considered the features of the best performing feature-set. Based on this, we used kPCA instead of PCA for the age and region features as kPCA captures the variance of the data better with the same data as the relationship between the data might be non-linear (Cao et al., 2003). To select the appropriate hyperparameters for kPCA and XGBoost algorithms, we used the RandomizedSearchCV (see appendix E) algorithm. The list of parameters that we have examined can be found in Appendix F.

Model Evaluation

To evaluate our final model, we had a look at two different metrics: the ROC curve and the precision-recall score based on different thresholds for the predicted probability of the test observations.

According to Géron (2017), we know that the receiver operating characteristic (ROC) plots the true-positive rate against the false-positive rate, whereas the dotted line represents the ROC curve of a purely random classifier.

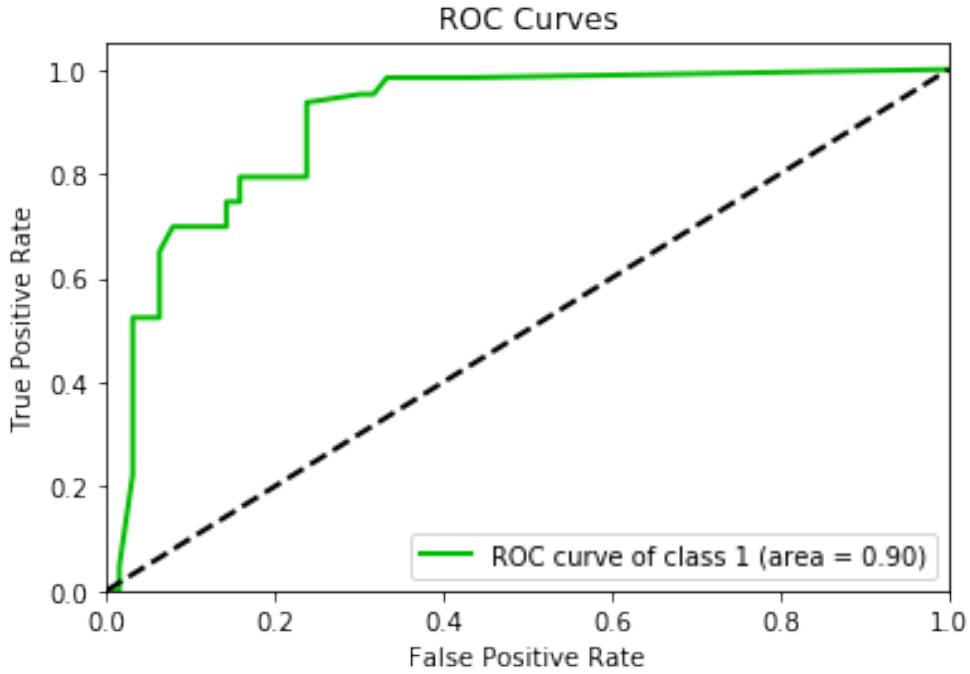

Figure 20: ROC Curve of final model

From (Géron, 2017, pp.79-101) we know that the precision is evaluated as:

$$\frac{True\ positive}{True\ positive + False\ positive}$$

Therefore, it shows a ratio which describes how many artists we have correctly labelled as successful out of all predicted successful artists, whereas the Recall is calculated as:

$$\frac{True\ positives}{True\ positives + False\ Negatives}$$

It shows how many artists we have labelled as successful out of all successful artists. Naturally, there is a trade-off between these two ratios.

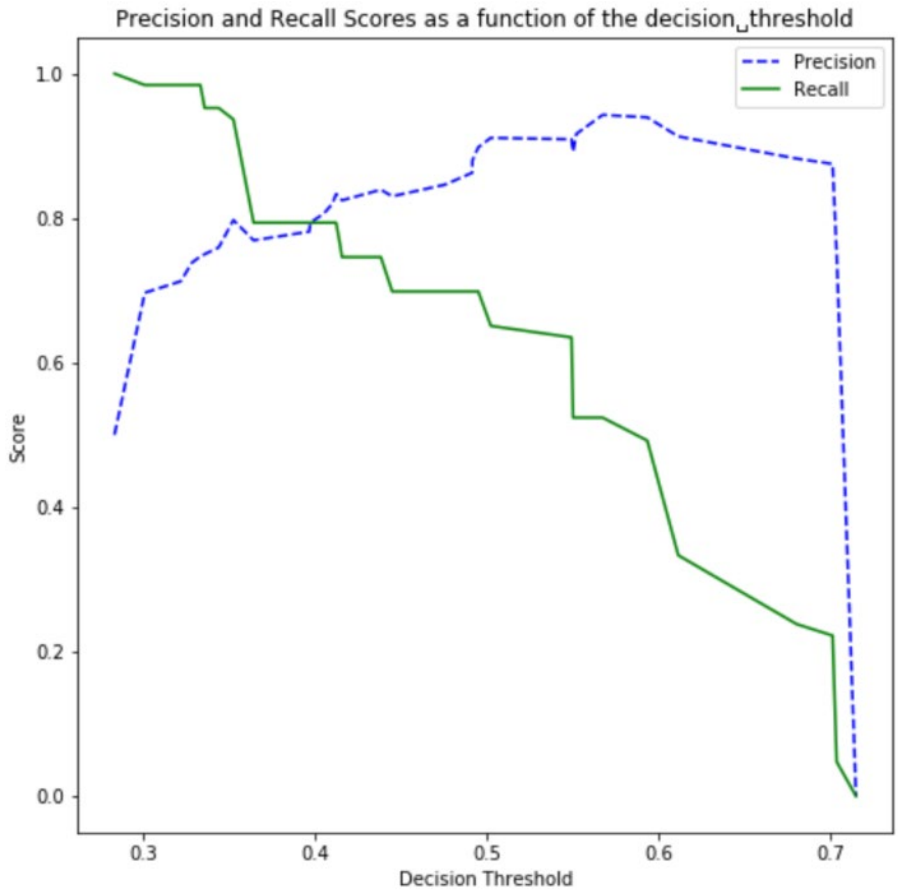

Figure 21: Precision and Recall

For our problem, we have decided not to use the standard threshold (0.50) but instead use a threshold of 0.35 so to classify most of the successful artists correctly (Recall = 0.94). Therefore, our model classified an artist as follows:

- unsuccessful if p(success) <= 0.35
- as successful if p(success) > 0.35

With this threshold, we obtained the following confusion matrix:

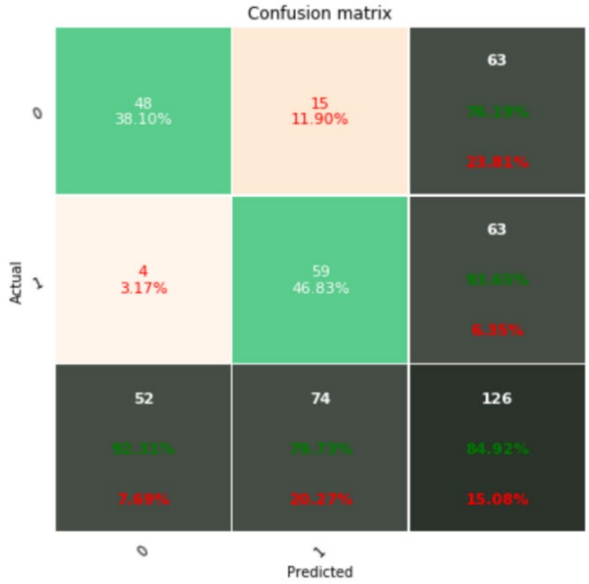

Figure 22: Confusion Matrix

Most importantly here we missed only 3.17% of successful artists. The high recall (0.94) yielded with the cost of 15 False Positives (i.e. classified as successful even though they are not). However, in the long-run, the cost of False Positives is likely to be more than covered from the revenue that will be generated from the successful artist that have been correctly classified.

Feature Importance

The feature importance shows that age and region were among the top twenty ones.

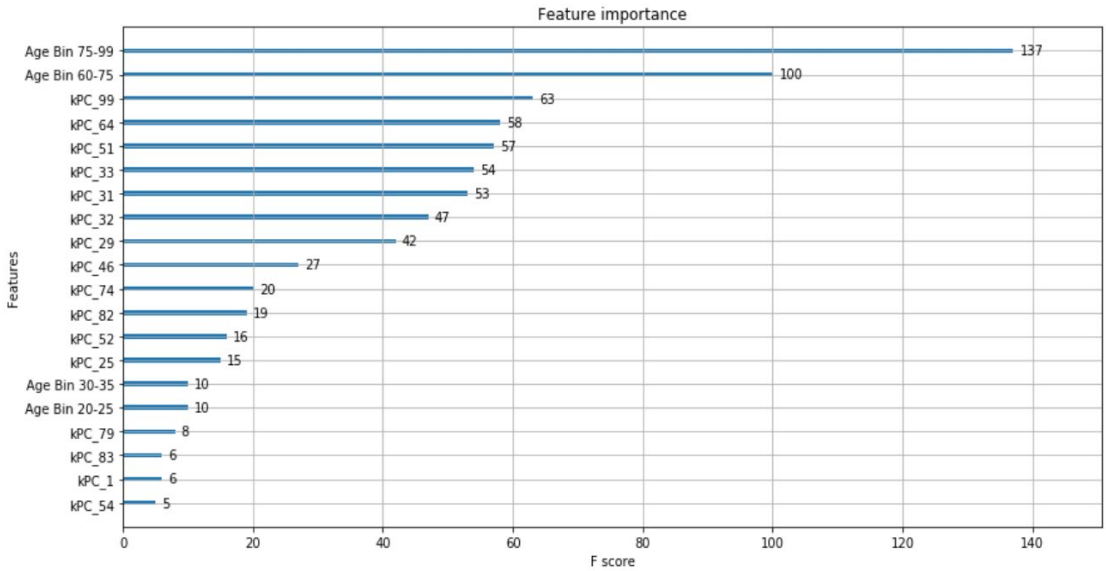

Figure 23: Feature Importance

Summary

In this report, we aimed to help WMG to predict whether an artist will succeed in 2017 based on streams from previous years on Spotify by investigating the factors which have predictive power on the success of an artist.

Firstly, we created basis-features regarding artists, users and playlists. To extend our model, we extracted features through the Spotify API. Then, we generated different combinations of features for the region and age of the streamers and reduced their dimension with the PCA.

With the best performing set of features (e.g. AUC) we created an XGBoost model which takes advantage of the kPCA for the features of region and age. Finally, we used a non-exhaustive method (RandomizedSearchCV) to find well-performing hyperparameters.

Results

Our final model evaluates to an AUC=0.90 by adjusting the decision threshold of the classifier; we ended up with a model that predicts most of the successful artists correctly (recall=0.94) at a cost of 15 False Positives. Finally, from the feature importance of the model, we have seen that the age and the region of the streamers were among the most important features to form our predictive model.

References

1. Wirth, R., 2000. CRISP-DM: Towards a standard process model for data mining. [online] Proceedings of the Fourth International Conference on the Practical Application of Knowledge Discovery and Data Mining, pp.29-39. Available at: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.198.5133>

2. A decision-tree-based model for evaluating the thermal comfort of horses - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/A-Data-mining-phases-according-to-CRISP-DM-Chapman-et-al-2000-B-Understanding-the_fig2_259595137

3. Bornmann, L., 2011. Scientific peer review. Annual Review of Information Science and Technology, 45(1), pp.197-245.

4. Abdi, H. and Williams, L., 2010. Principal component analysis. Wiley Interdisciplinary Reviews: Computational Statistics, [online] 2(4), pp.433-459. Available at: <https://doi.org/10.1002/wics.101>.

5. Benslimane, D., Dustdar, S. and Sheth, A., 2008. Services Mashups: The New Generation of Web Applications. *IEEE Internet Computing*, [online] 12(5), pp.13-15. Available at: <https://doi.org/10.1109/mic.2008.110>.

6. Cao, L., Chua, K., Chong, W., Lee, H. and Gu, Q., 2003. A comparison of PCA, KPCA and ICA for dimensionality reduction in support vector machine. Neurocomputing, [online] 55(1-2), pp.321-336. Available at: <https://doi.org/10.1016/S0925-2312(03)00433-8>.

7. Chen, T. and Guestrin, C., 2016. XGBoost: A Scalable Tree Boosting System. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16), [online] pp.785–794. Available at: <https://doi.org/10.1145/2939672.2939785>

8. En.wikipedia.org. 2020. Kaggle. [online] Available at: <https://en.wikipedia.org/wiki/Kaggle> [Accessed 10 April 2020].

9. Chawla, N., Bowyer, K., Hall, L. and Kegelmeyer, W., 2002. SMOTE: Synthetic Minority Over-sampling Technique. Journal of Artificial Intelligence Research, [online] 16, pp.321-357. Available at: <https://doi.org/10.1613/jair.953>.

10. Developer.spotify.com. 2020. Web API | Spotify For Developers. [online] Available at: <https://developer.spotify.com/documentation/web-api/> [Accessed 6 April 2020].

11. Géron, A., 2017. *Hands-On Machine Learning With Scikit-Learn And Tensorflow*. 1st ed. O'Reilly Media, Inc., pp.79-101.

12. Huang, J. and Ling, C., 2005. Using AUC and accuracy in evaluating learning algorithms. *IEEE Transactions on Knowledge and Data Engineering*, [online] 17(3), pp.299-310. Available at: <https://doi.org/10.1109/tkde.2005.50>.

13. Paper, D., 2020. *Hands-On Scikit-Learn For Machine Learning Applications*. Berkeley, CA: Apress, pp.165-188.

14. Pearson, K., 1901. LIII. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, [online] 2(11), pp.559-572. Available at: <https://doi.org/10.1080/14786440109462720>.

15. Raschka, S., Patterson, J. and Nole, C., 2020. Machine Learning in Python: Main developments and technology trends in data science, machine learning, and artificial intelligence. [online] pp.5-29. Available at: <https://arxiv.org/pdf/2002.04803.pdf>.

16. Yang, L., Terry, H., Sugiyama, M., Jankowski, S. and Bellini, H., 2016. Music In The Air. [online] The Goldman Sachs Group. Available at: <https://www.goldmansachs.com/insights/ pages/infographics/music-streaming/stairway-to-heaven.pdf>.

17. Wagner Cipriano, 2018, pretty-print-confusion-matrix, GitHub Repository. Available at: <https://github.com/wcipriano/pretty-print-confusion-matrix/blob/master/ confusion_matrix_pretty_print.py>

18. Géron, A., 2017. Hands-On Machine Learning With Scikit-Learn And Tensorflow. Chapter 8 – Dimensionality Reduction, kPCA, 2nd ed. O'Reilly Media, Inc., pp.228-231.

19. Aurélien Geron, 2017, Hands-on Machine Learning with Scikit-Learn, Keras and TensorFlow, Dimensionality Reduction Jupyter Notebook GitHub Repository. Available at: <https://github.com/ageron/handson-ml2/blob/master/08_dimensionality_reduction.ipynb>

20. Xgboost.readthedocs.io. 2020. Python API Reference — Xgboost 1.1.0-SNAPSHOT Documentation. [online] Available at: <https://xgboost.readthedocs.io/en/latest/python/python_api.html> [Accessed 13 April 2020].

21. Scikit-learn.org. 2020. Sklearn.Decomposition.Kernelpca — Scikit-Learn 0.22.2 Documentation. [online] Available at: <https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.KernelPCA.html> [Accessed 13 April 2020].

22. Python, C., 2020. Complete Guide To Parameter Tuning In Xgboost (With Codes In Python). [online] Analytics Vidhya. Available at: <https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/> [Accessed 13 April 2020].

23. Piramuthu, S., 2008. Input data for decision trees. Expert Systems with applications, 34(2), pp.1220-1226.

Appendix

A) Methodology for the feature of average time listened for all of the songs of an artist (art_streamlength_ratio_mean)

This feature is based on all streams of the users towards the songs of an artist.
The methodology is comprised of the following steps:

1. For each song of an artist, we have extracted the maximum stream length (in seconds) out of all streamers. We assume that this is the actual full length of the track.
2. Based on this metric, we calculate a ratio for which has as the numerator the stream length of a user towards a track. In the denominator, we have the maximum stream length. When the ratio is equal to 1, the streamer has listened to all the song in full. For lower values, the streamer has listened to the song partially.
3. We aggregate this feature by grouping by the artist. We get the mean of this aggregation. A high ratio indicates that users tend to listen to the tracks of an artist in full, without skipping them

B) Spotipy API

As retrieving the data from the API takes a long time, we decided to add in an additional folder ("spotipy") in our faculty project which contains a file "Spotipy2.ipynb". The latter connects to the API and gathers the information which we then convert into a CSV file and add into our final dataframe. The process for obtaining the information from the API is as follows: We first need to connect to the API and then get on a per track basis all information (artist, album name, album uri, track, release date, id, song uri, track href, analysis url, type, danceability, energy, key, loudness, mode, speechiness, acousticness, instrumentalness, liveness, valence, tempo, duration ms, time signature) of artists who have an album. Following, we will do the same for tracks which were not featured on an album. We added these two frames together and manually imputed missing artists and their tracks by looking up their specific Spotify URIs. After obtaining a complete dataframe with all artists and the features on a per track basis, we aggregate them onto a per artist basis by taking the mean for each artist per feature. We hence obtain a dataframe with additional information on the average characteristics of their songs – i.e. how danceable are their songs on average.

C) The five feature-sets

In the beginning, we calculated:

1. "Age_bins_count": A feature set for each artist with the total number of streams per age group[2]
2. "Age_bins_perc": Same as above, but with the percentage of listeners for each artist per group

[2] Age groups are divided into groups for 15-20, 20-25, 25-30, 30-35, 35-40, 40-50, 50-60, 60-75, 75-99 year olds

3. "Ages_count": For each artist the total number of streams for each individual age
4. "Ages_perc": Same as above, but with the percentage of listeners for each artist per individual age
5. "Region-Age PCA": PCA performed on a matrix that has the stream counts for every age bin in every region. (The two columns combined)

Note: Whenever we used PCA for these scenarios, the number of principal components was always 10, as it was giving satisfactory results in terms of variance explained.

Final Basis DataFrame and additionally:	Train & Test set
Age Bins Count (No PCA) and region codes PCA	• age_bins_count_train • age_bins_count_test
Age Bins Percentages (No PCA) and region codes PCA	• age_bins_perc_train • age_bins_perc_test
All Ages Counts (PCA) and region codes PCA	• ages_count_train • ages_count_test
All Ages Percentages (PCA) AND region codes PCA	• ages_perc_train • ages_perc_test
Region-Age Count PCA	• ageregion_count_train • ageregion_count_test

Figure App. C-1: Overview of final data frames

All these scenarios were concatenated on the dataframe with the features that we have previously created (manually and through the API). Thus, this is how the 5 different datasets were created in order to compare their predictive power (AUC) through the XGBoost algorithm and choose the one with the highest AUC.

D) Advantages of XGBoost

The advantages of XGBoost are manifold, but they are verified by it being one of the most popular and successful machine learning algorithms – Chen and Guestrin (2016) mentioned that among the 29 challenge winning solutions published at Kaggle's[3] blog during 2015, 17 solutions used XGBoost. The authors of that paper base the popularity of XGBoost on its performance in a vast variety of problems, scalability, and to its ability for handling sparse data, (through a novel tree learning algorithm) which is enhanced by parallel and distributed computing that makes efficient use of every available resource.

E) RandomizedSearchCV

Randomized search is a technique where random combinations of the hyperparameters are used to find the best solution for the model by performing multiple random iterations. It is more time and space efficient compared to an exhaustive grid search, and it has been proven to yield excellent results, without sacrificing optimization significantly (Paper, 2020, pp.166-188).

[3] Kaggle, a subsidiary of Google LLC, is an online community of data scientists […] which allows users to find and publish data sets, explore and build models […] enter competitions to solve data science challenges.

F) Hyperparameters of kPCA & XGBoost

Below you can find the description of the parameters for kPCA & XGBoost accompanied with their selected value after the hyperparameter tuning. The parameters have been selected according to "Hands-on Machine Learning with Scikit-Learn, Keras and TensorFlow" book for kPCA and popular online guides for XGBoost (ref. 20-24)

kPCA Hyperparameters per documentation

- kernel ('sigmoid') : type of kernel
- n_components (118) : Number of principal components
- gamma (0.0475) : Kernel coefficient
- alpha (0.2611) : Hyperparameter of the ridge regression that learns the inverse transform

XGBoost Hyperparameters per documentation

- eval_metric ('auc') : Evaluation metrics for validation data, a default metric will be assigned according to objective. In our case we used 'auc' (Area Under the Curve)
- nthread (-1) : Number of parallel threads used to run XGBoost. When it has value equal to -1, uses all of the CPU cores. This lead to better computation time
- eta (0.7) : Step size shrinkage used in update to prevents overfitting. After each boosting step, we can directly get the weights of new features, and eta shrinks the feature weights to make the boosting process more conservative.
- max_depth (5) : Maximum depth of a tree. Increasing this value will make the model more complex and more likely to overfit. 0 is only accepted in lossguided growing policy when tree_method is set as hist and it indicates no limit on depth. Beware that XGBoost aggressively consumes memory when training a deep tree.
- max_leaf_nodes (1750) : The maximum number of terminal nodes or leaves in a tree.
- estimators (4833) : Number of gradient boosted trees. Equivalent to number of boosting rounds.
- learning_rate (0.00583) : Boosting learning rate