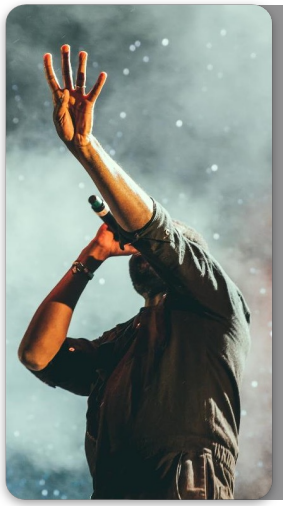
INSY662 Data Mining & Visualization

M^2

Mood-Match Playlist Generator

Adrian Alarcon, Arham Anwar, Chiara Lu,
Kazuya Hayashi & Rodrigo Castro

Executive Summary



Imagine a music platform that could understand your feelings: "feeling down" or "can't wait to go home." M² generates the perfect playlist for each user based on their sentiment through a text query.

Steps

Data Extraction from Spotify

Exploratory data Analysis

Pre-processing & Sentiment Score

Modeling & Testing

Front End

Analytics Problem Statement

Limitations in Music Streaming Services

- Primary focus on genre and artist recommendation
- Lack personalized, mood-based playlist generation
- Manual playlist generation

Bridging the Emotional Gap

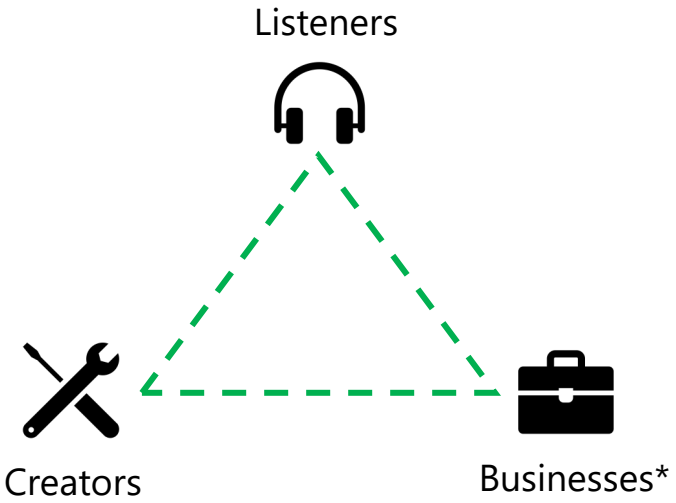
- Enhanced user experience through mood-based playlists.
- M^2 matches user emotions with music.

Analytics Problem Statement

Business Value

- Improve user engagement and satisfaction
- Increase user retention
- Track user moods for future products
- Increase revenue through more streams
- Lay the groundwork for further innovation

Key Stakeholders



*(Businesses refer to marketing team, investors, PMs)

Data Source Extraction

1. Obtained Spotify **API credentials**

2. Set up/**authenticate** project

3. Searched & **retrieved** playlist details

4. Extracted **track information** from playlists

5. Fetch lyrics (**integrated lyrics service**)

6. **Tested** with various playlists & implemented logging

Data Exploration – Steps

Load_data (path)

remove_duplicates(df):

dataprofiling(df)

create_scatterplots(df)

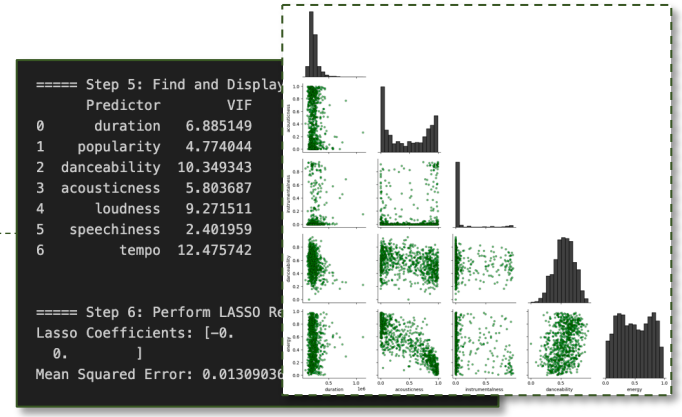
process_data(df)

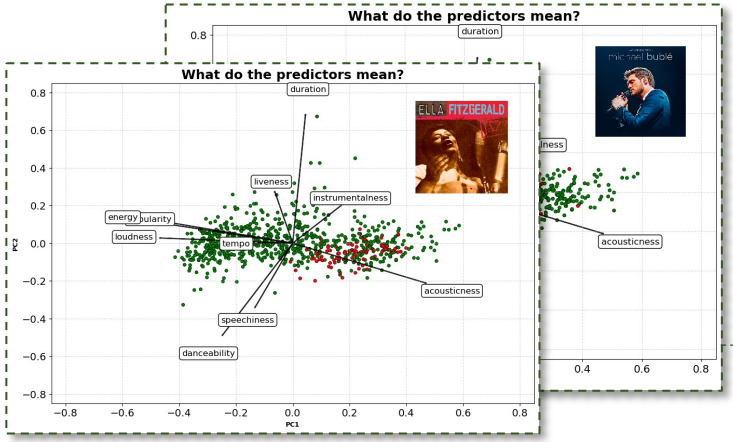
Mcollinearity(df)

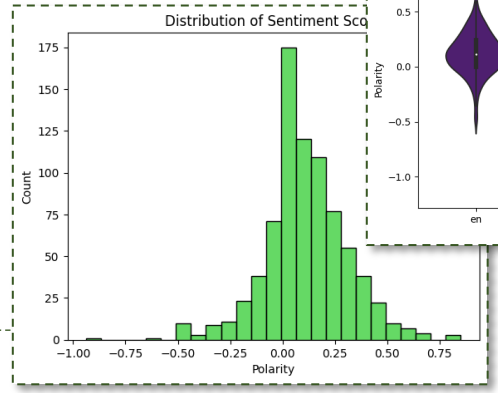
pca(df)

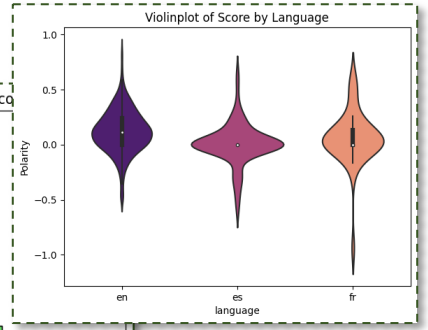
artist_profiles(df)

Data_Analysis(df)



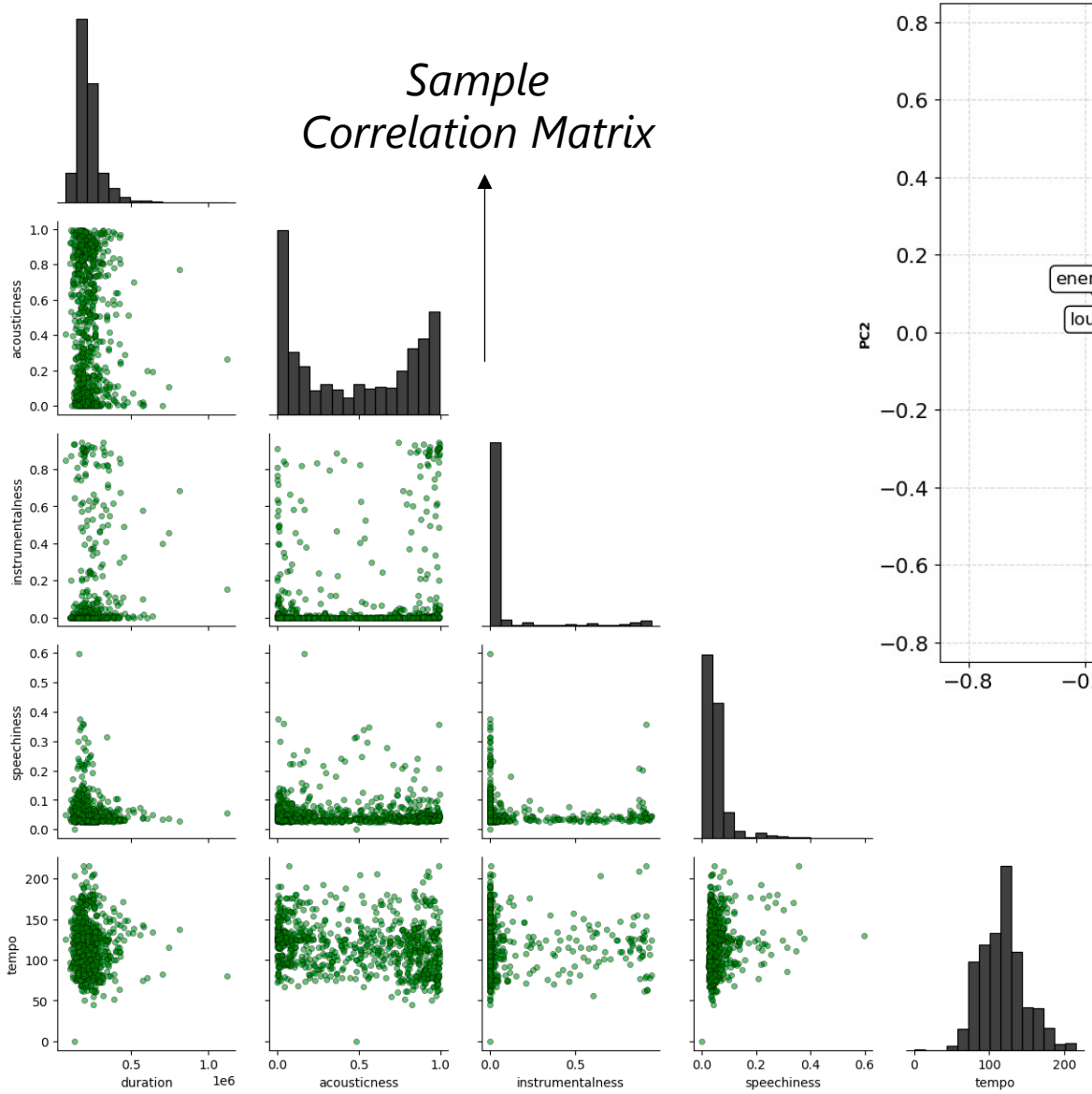


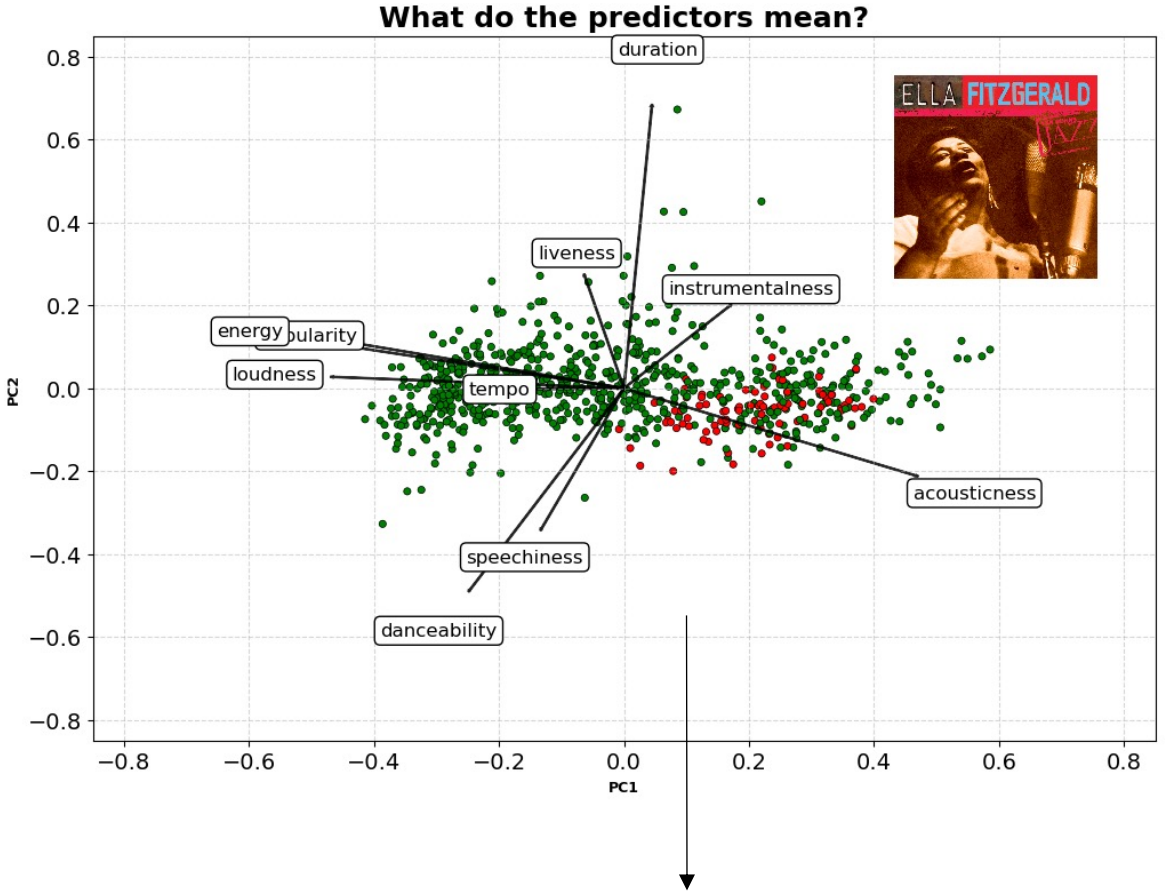




Data Exploration

Sample Correlation Matrix





PCA was used to understand music repository profile

Data Pre-processing summary

1. **Import Libraries**: Import necessary libraries for data pre-processing, including pandas, langdetect, spacy, nltk, and others.

2. **Import Dataset**: Read the dataset from an Excel file and display the first few rows.

3. **Remove Irrelevant Columns:** Drop unnecessary columns for analysis.

4. **Identify and Remove Duplicates**: Check for and remove duplicate rows, keeping the first occurrence.

5. **Handle Missing Values**: Check for missing values and handle them.

6. **Reset Index**: Reset the DataFrame index.

7. **Filter Out Songs Without Lyrics**: Identify and remove songs without available lyrics.

8. **Clean Lyrics Text**: Create a lambda function to delete unnecessary text before the actual lyrics.

9. **Detect Language of Lyrics**: Use the langdetect library to detect the language of each lyric.

10. **Filter by Language**: Keep songs only in English, Spanish, or French.

11. **Divide into Language-specific DataFrames**: Create separate DataFrames for English, Spanish, and French lyrics.

12. **Process Lyrics with NLP Models**: Use pre-trained language models (Spacy) to process lyrics for each language.

13. **Remove Unwanted Words**: Define and remove unwanted words from processed lyrics.

14. **Sentiment Analysis**: Use TextBlob to analyze sentiment scores for each lyric.

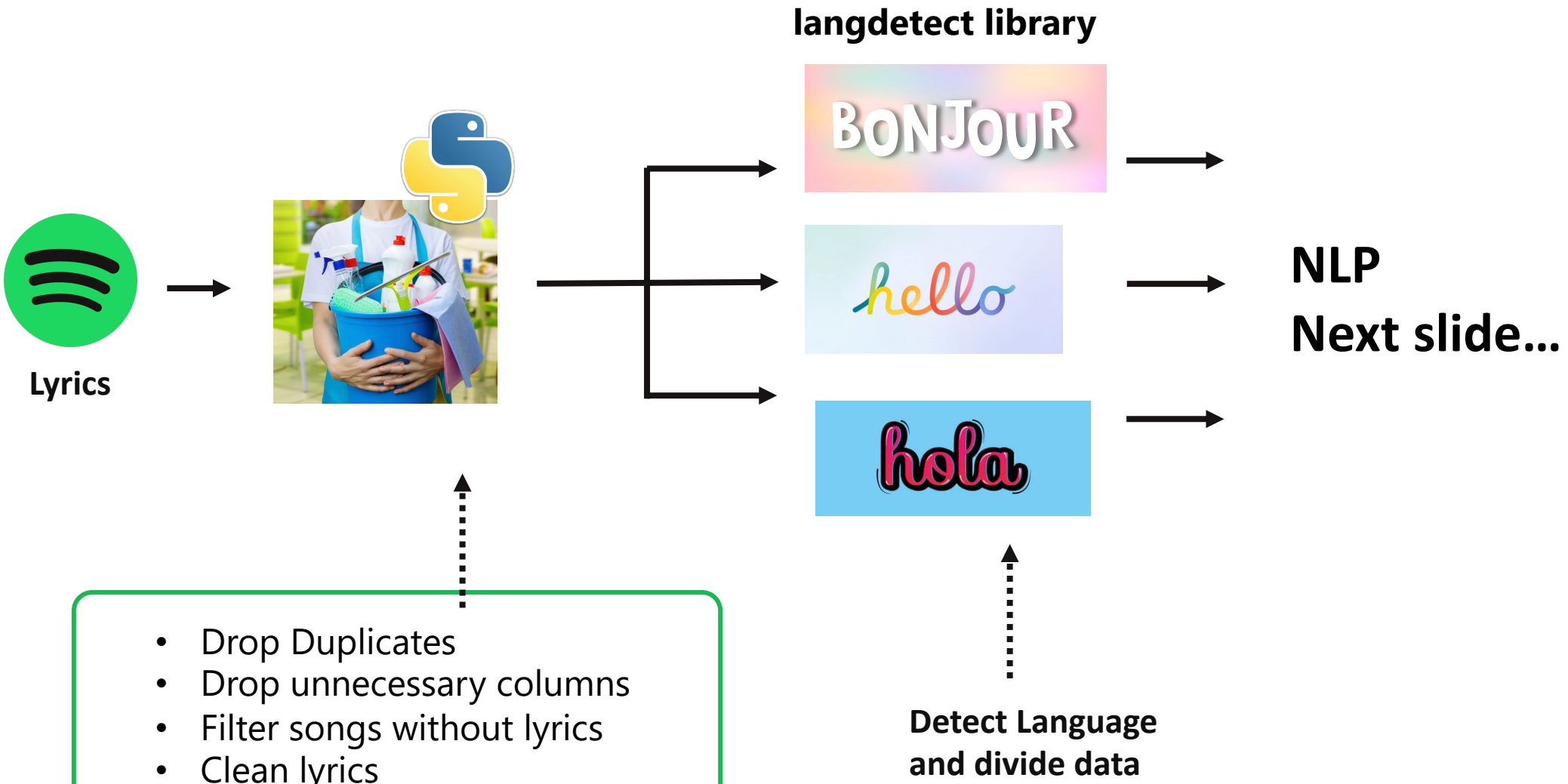
15. **Extract Polarity and Subjectivity**: Split sentiment scores into polarity and subjectivity columns.

16. **Drop Unnecessary Columns**: Remove the original lyrics column.

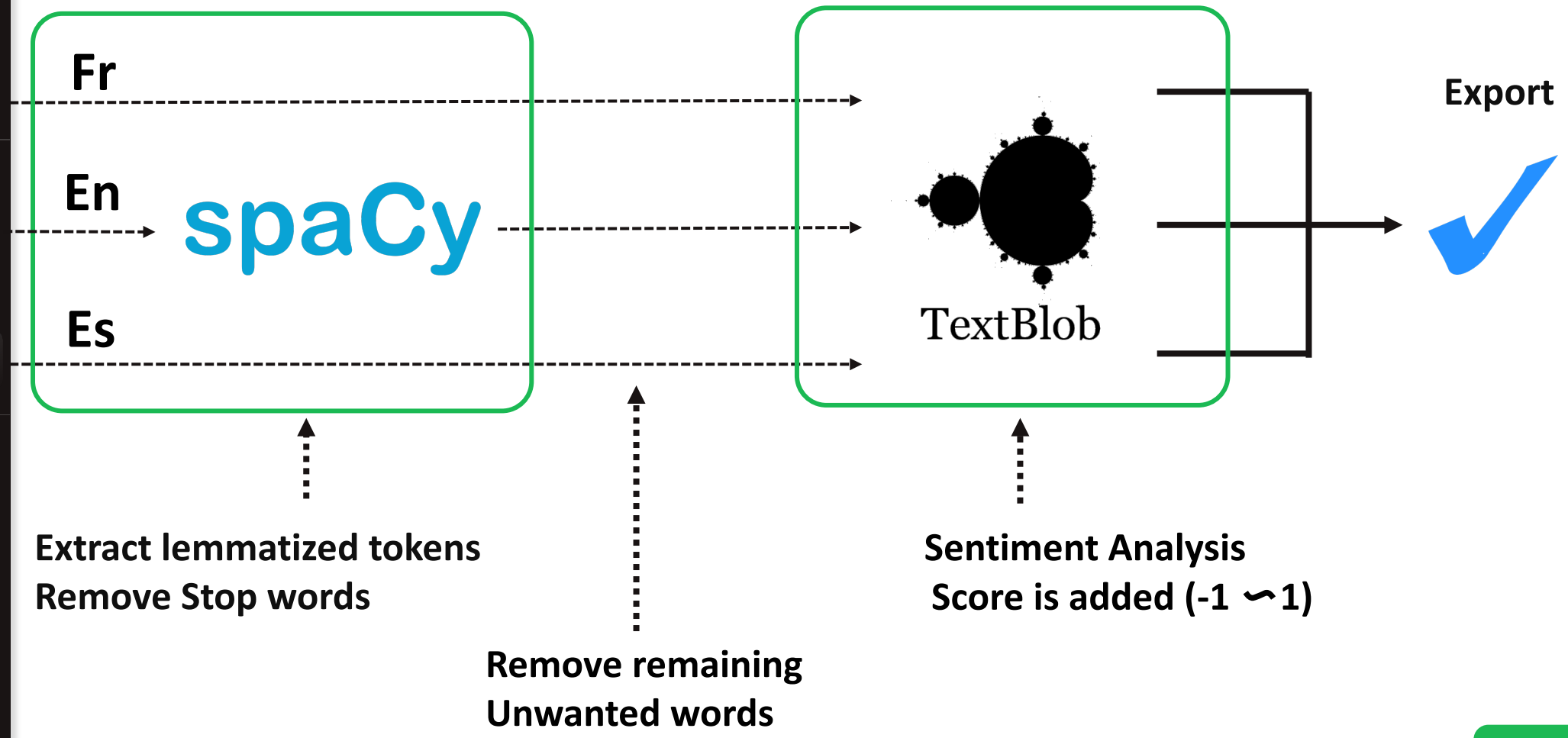
17. **Combine DataFrames**: Concatenate language-specific DataFrames into a single DataFrame.

18. **Export Cleaned Data**: Save the cleaned data to an Excel file.

Data Pre-processing



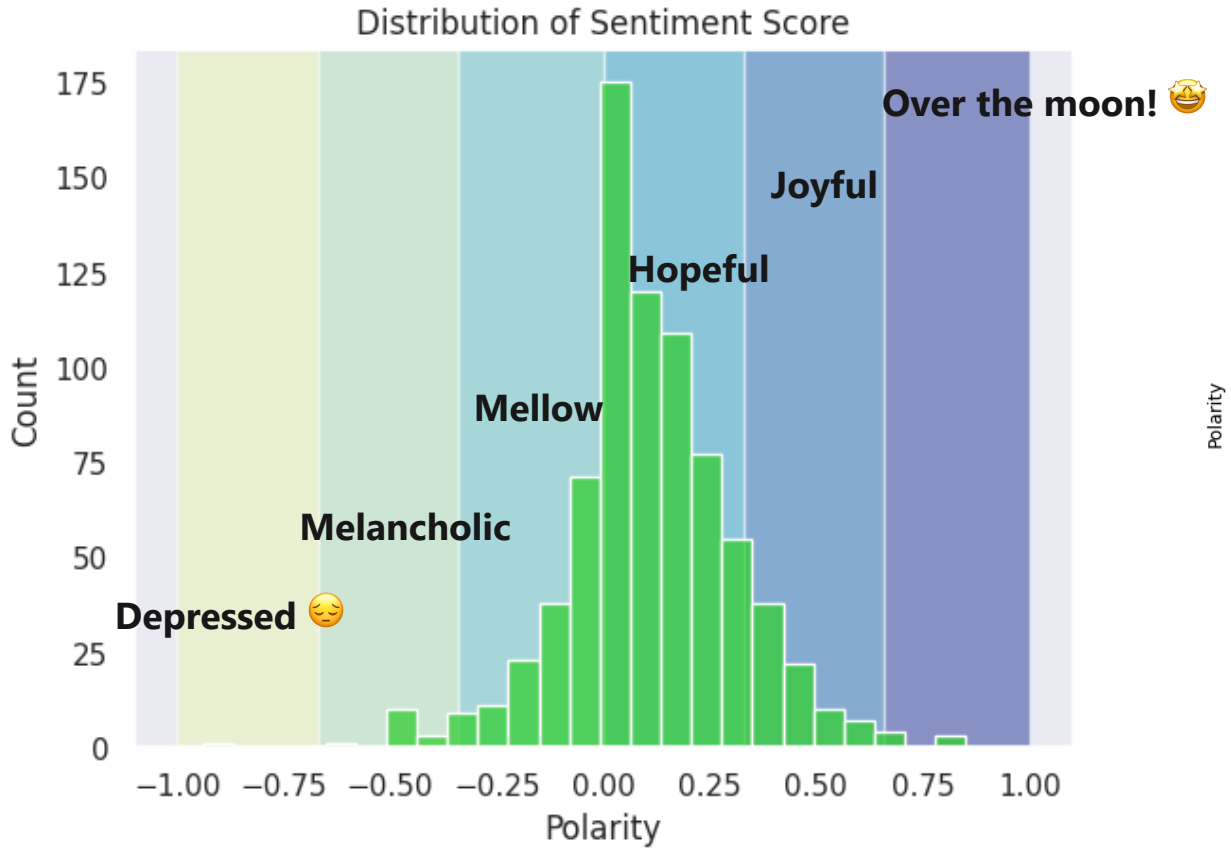
Data Pre-processing



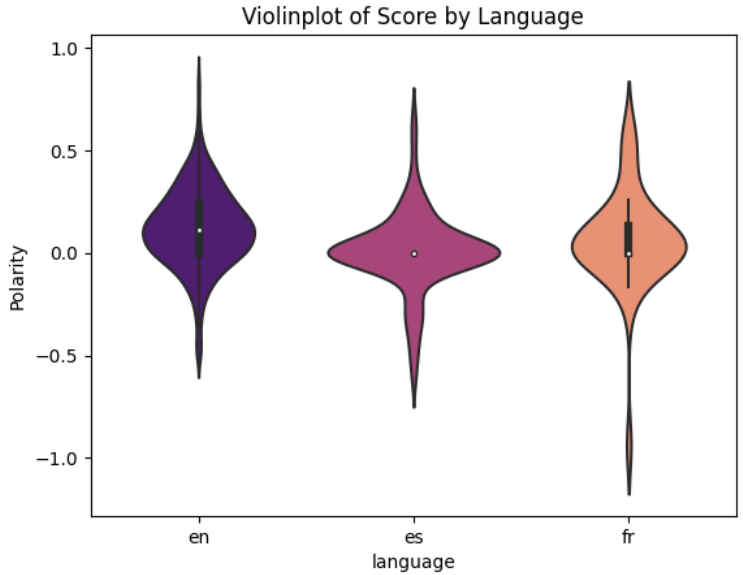
Sentiment Exploration

Sentiment Score Distribution

Overall

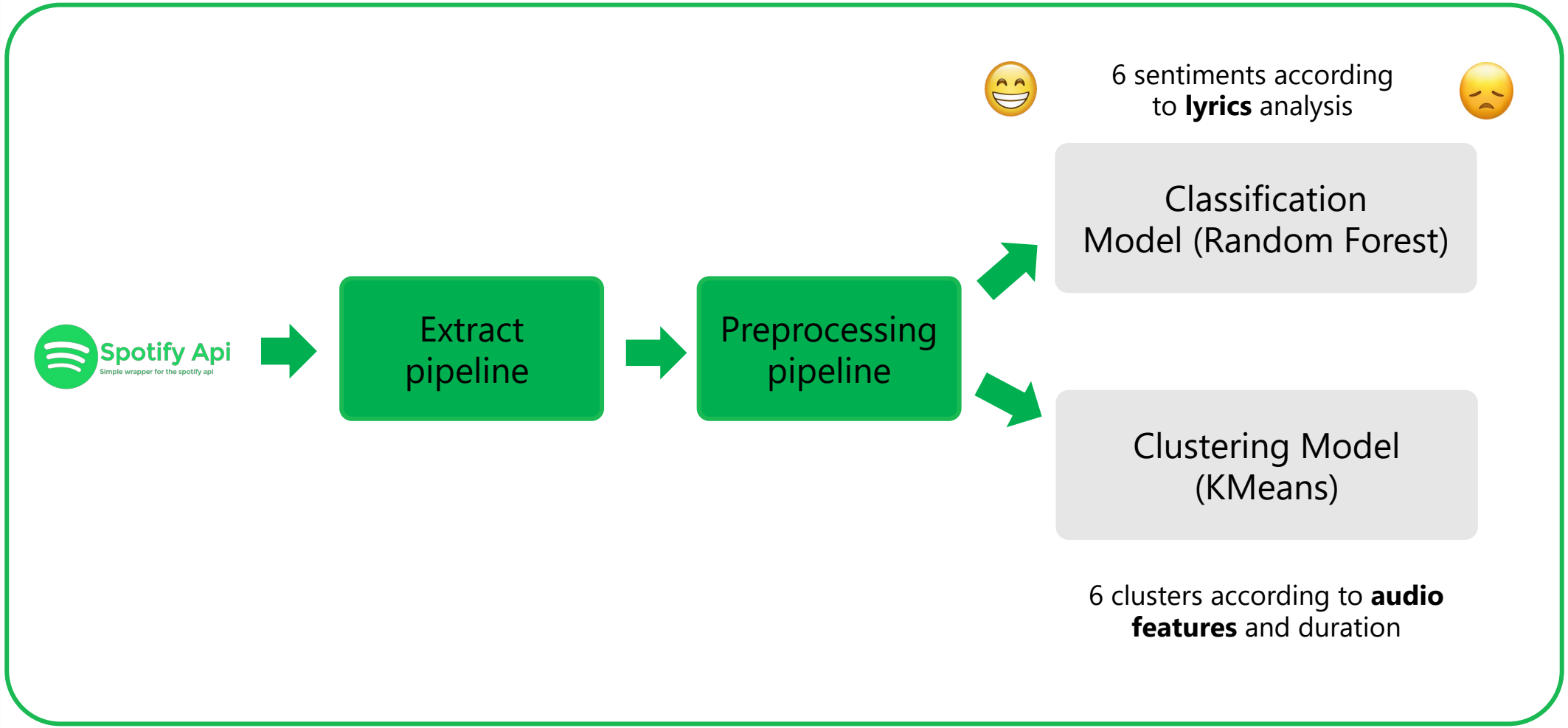


By Language



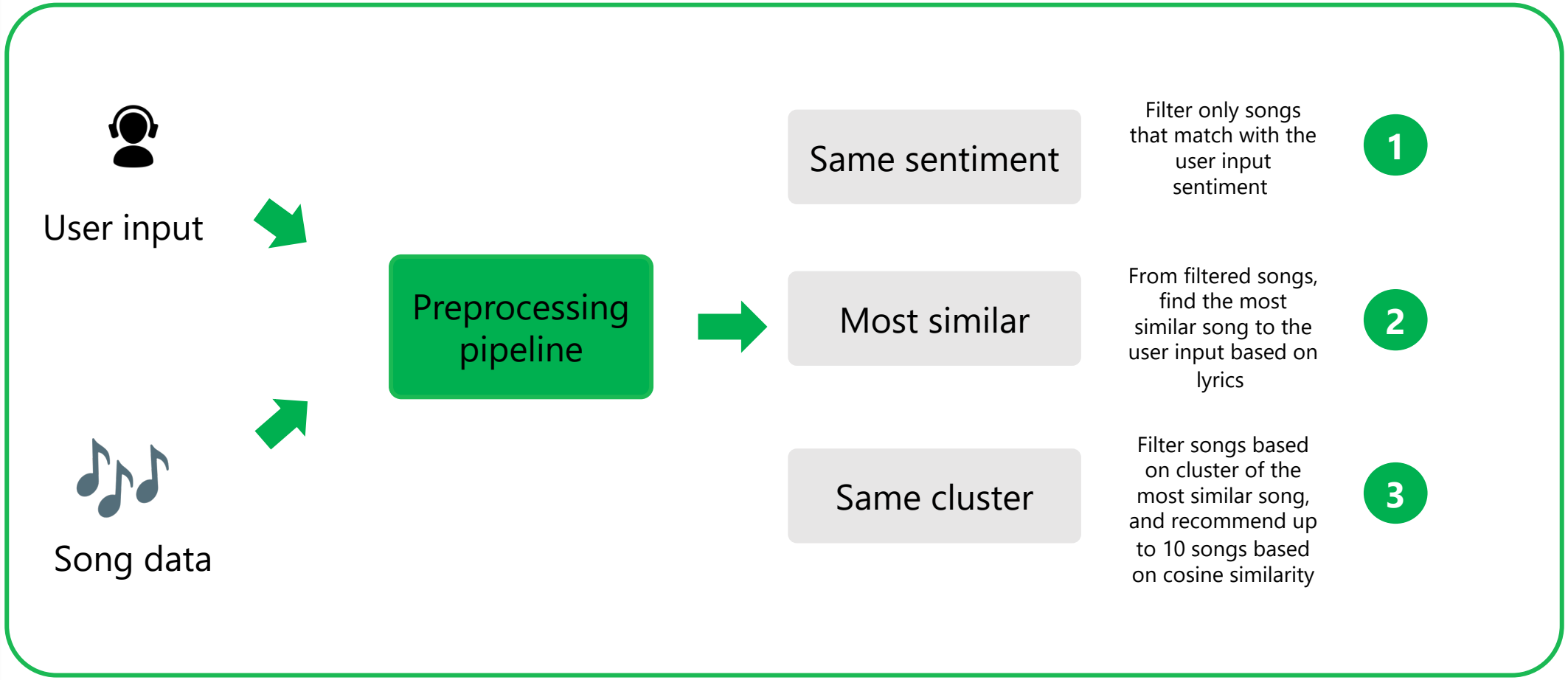
Model Description

Training Architecture

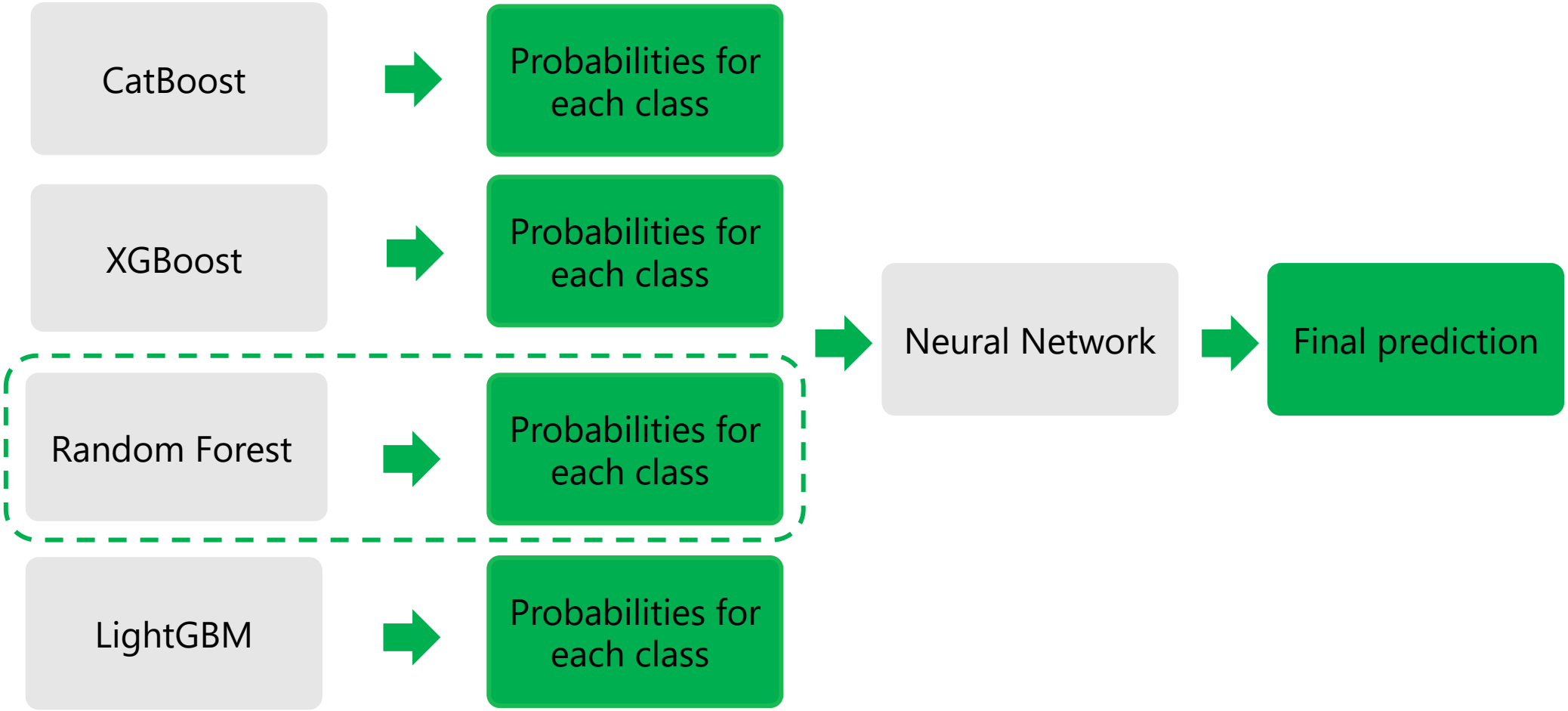


Model Description

Predict Architecture



Stacked Model



Model Selection

	Workings	Pros	Cons	Accuracy
CatBoost	Handles categorical data	Handles categorical values automatically	Long training time and low interpretability	0.7121
LightGBM	Uses leaf-wise growth strategy	High efficiency and speed	Prone to overfitting and limited interpretability	0.7121
XGBoost	Implements parallel tree boosting	Regularization prevents overfitting	Sensitive to hyperparameter tuning	0.6919
Random Forest	Ensemble of decision trees	Robust to overfitting	Computationally expensive	0.7071
MLP	ANN with multiple nodes	Learns complex and non-linear patterns	Sensitive to hyperparameter tuning	0.6616
Stacked Model	Multiple models stacked	Better accuracy	Low interpretability	0.7010

Model summary

The solution has two parts. First, it includes a classification model that predicts the sentiment of a song (6 classes). For this, many models have been tested. Here a summary of each of them:

- Catboost with default hyperparams
- LigthGBM with default hyperparams
- XGBoost with default hyperparams
- Neural Network
- RandomForest with default hyparams
- Stacked model

The purpose of the stacked model was to combine many predictors and ensamble them with a non linear model to try to create a more robust model, so the last non linear model regularizes the errors of the previous models. This brings a better accuracy on prediction but it also supposes more maintainance and no interpretability.

For the purpose of this work, the group selected RandomForest with hyperparams using Optuna. This is a framework that allows to save time when optimizing hyperparams using a Bayesian approach.

Recommendation model:

The infraestructure for the recommendation model includes two parts. First a clustering model, and then, a cosine similarity match with lyrics.

1. Clustering model:

To identify songs with similar audio features, a KMeans model with K = 6 has been trained using only audio features. The number of clusters have been selected using Silhouette criteria.

2. Recommendation engine

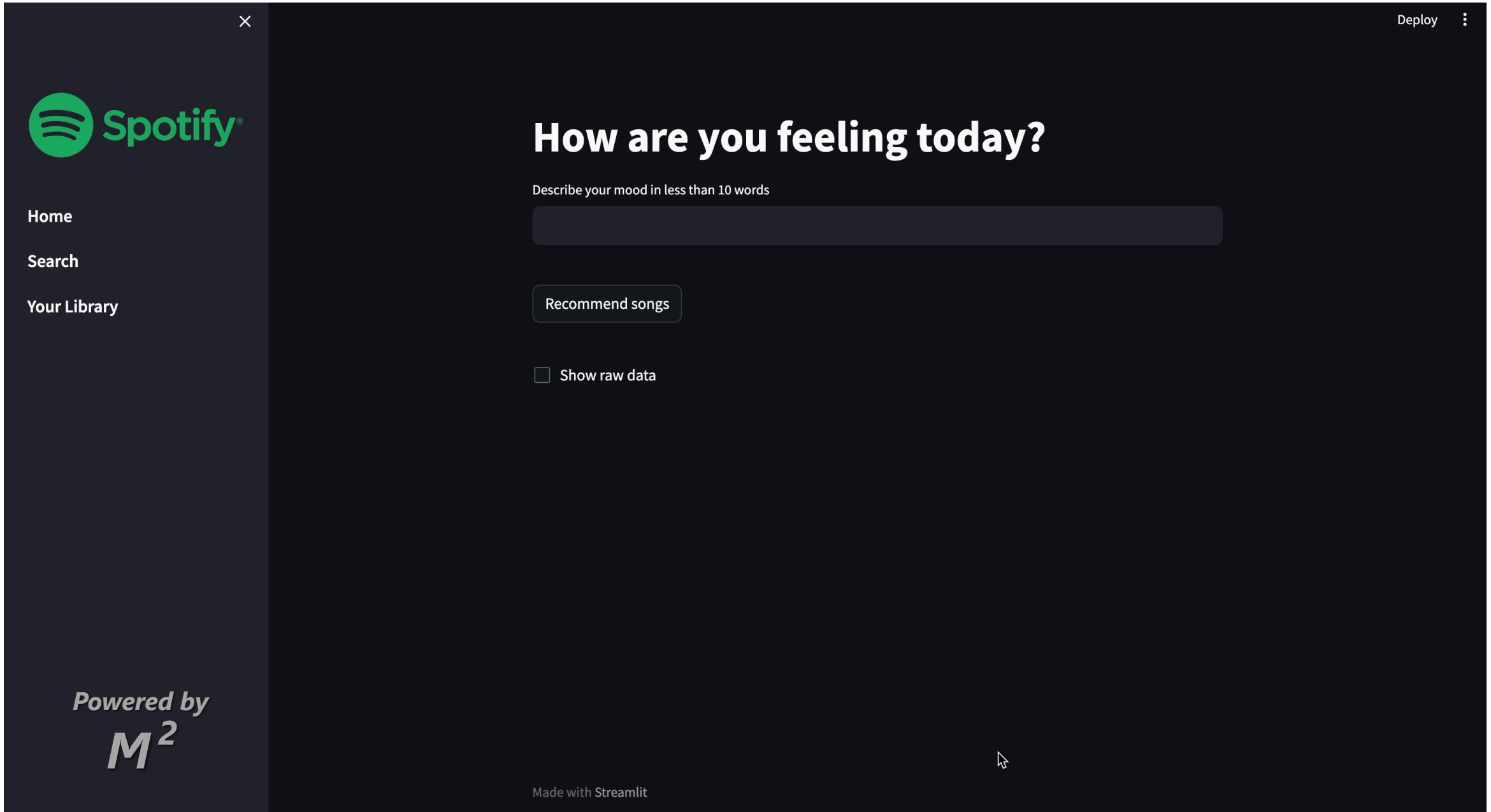
To generate the recommendation, we considered only songs with the same predicted sentiment, using the classification algorithm.

Then, among the filtered songs, we found the most similar song between user input and song's lyrics.

Finally, we only considered songs that are in the same cluster of the most similar and the recommendation output is sorted by cosine similarity between user input and song's lyrics.

Model Demo

https://youtu.be/_IJkxZAUFSg



P&L Analysis

Revenue= Songs played x Net earnings
per stream

$ 167,440,000

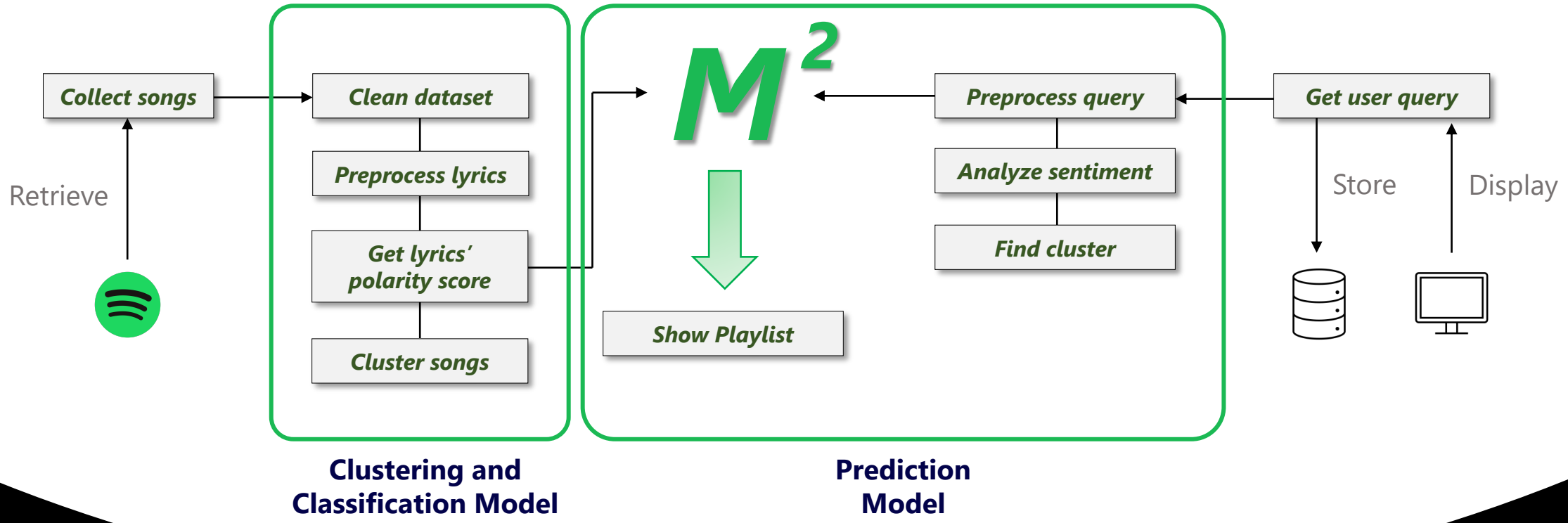
Benchmark settings:
- 20% of 182 million subscribers on Spotify. Each subscriber listens to 1000 songs
- $0.005 earnings per stream (Spotify and creators)
- Cost of vectorizing and matching one song is $0.0004
- Currency in USD

Business Benefits

	Listeners	Creators	Businesses
Personalization	Personalized playlist based on sentiment	Creating and remixing songs using polarity scores	Automated song and creator classification
Experience	Easier search prompts	Equality of opportunity	Enhanced user engagement & loyalty
v2.0 Sentiment Analytics	Track personal mood and rate how well playlists suited their sentiment	Visualize trends on listeners' moods	Implement Personalized marketing

Summary

Enhanced Personalization & Experience through Mood-matching Songs Recommendation System.



Keep Vibin’

Thank you for listening
We’re open for questions

References

Carter, G. (2023, August 23). How much does Spotify pay per stream? Music Gateway.

Çano, E. (2018). Text-based sentiment analysis and music emotion recognition. *arXiv (Cornell University)*. https://doi.org/10.6092/polito/porto/2709436

Knibbe, J. (2021, September). *Are music recommendation algorithms fair to emerging artists? Music Tomorrow Blog*. Music Tomorrow. Retrieved November 21, 2023, from https://www.music-tomorrow.com/blog/fairness-and-diversity-in-music-recommendation-algorithms

Software Used