Trendify

Analyzing Cross-Platform Impact on Spotify Popularity

Gabe Compton, Henok Gelan, Abdur Islam, Osaze Ogieriakhi, Justin Moos, Aden Athar, Jaeger Nelson, Kaleb Kedebe

Motivation

- Music success is now driven by more than just ra
  and albums
- TikTok trends are reviving old popular songs
- Understanding which platforms give the biggest
  push

Problem Statement

How do external platforms like Tik Tok and YouTube influence Spotify Track Score?

Real-World Importance: Artists and labels need insight into cross-platform impact

For example:

🎵 Song: "Bloody Mary" – Lady Gaga
📱 TikTok Trend: November–December 2022
📈 Spotify Spike: Massive streaming boost in late 2022 / early 2023

The song was originally released in 2011, not a new track.

It went viral in late 2022 due to Wednesday Addams dance.

After going viral on TikTok, it entered the Spotify Top 200 in multiple countries, 12 years after release.

Lady Gaga even released an official music video for it in response to the renewed popularity.

Challenges in Analyzing Cross-Platform Influence

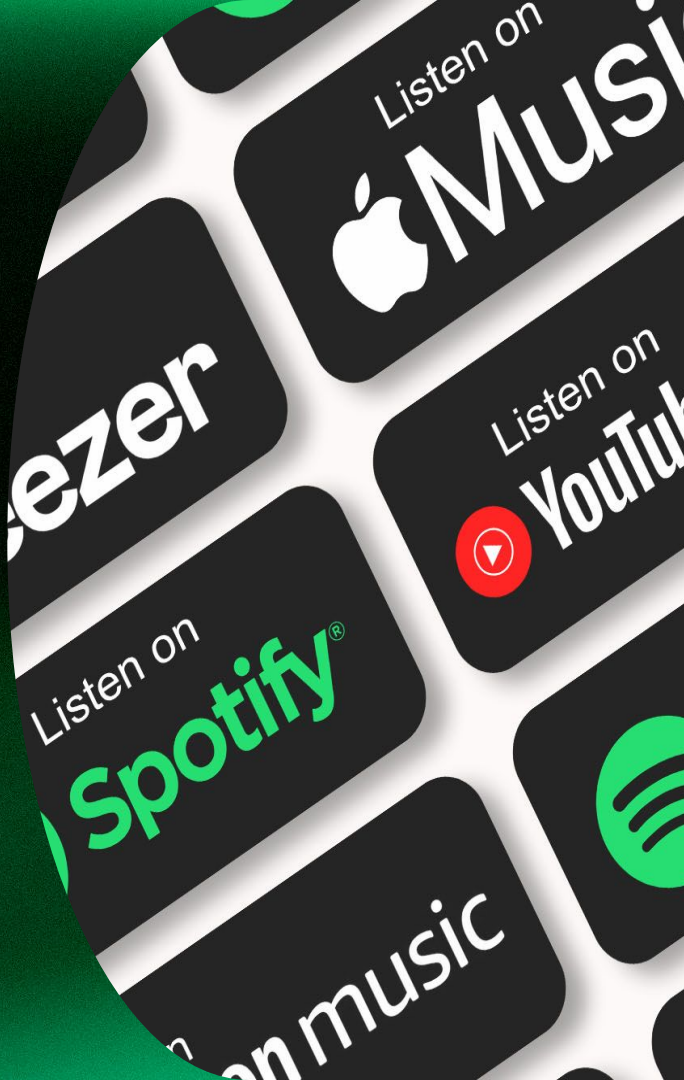
🔧 Challenge	🧠 Description
💾 Data Aggregation	Platforms use different metrics
🧹 Data Cleaning	Nulls, duplicates, formats
⏳ Time Lag	Trend ≠ instant chart change
📊 Measuring Impact	Distinguishing viral spikes
⚖️ Normalization	Controlling for overall popularity

Data Mining Questions

#1	**Which platform has the highest correlation with Track Score?**	3:45
#2	**Do older songs regain popularity due to TikTok/YouTube trends?**	5:00
#3	**Can a spike in YouTube views predict an increase in Spotify streams?**	3:45
#4	**Does recency impact a song's success?**	5:00

Dataset Overview

- Spotify top songs 2024 dataset
- Kaggle
- Track score, streams, playlist count, views/likes from YouTube, TikTok metrics



Preprocessing/cleaning

- Dropped duplicates
- Cleaned strings
- Converted release date into datetime, extracted release month
- Removed column with > 1000 nulls
- Scaled all features using StandardScaler

Technical Approach

#01	Data Collection & Preprocessing	Removed duplicates and irrelevant columns Cleaned string-formatted numbers and handled missing values by dropping columns with >1000 nulls and rows with critical NaNs.
#02	Feature Selection	Focused on the features that had better potential to reflect external influence: Spotify Streams, YouTube Views, TikTok Views, Apple Music Playlist Count, AirPlay Spins Shazam Counts
#03	Normalization Strategy	All features were normalized by Spotify Streams. This allows analysis to focus on the proportionate impact rather than absolute numbers, controlling for overall popularity.

Technical Approach

#04	Reduction with PCA	PCA was performed to reduce dimensionality and uncover latent structures in the normalized data.
#05	Conclusion & Visualization	PCA biplot was used to visualize song distribution & feature influence directions. Loadings were plotted to identify which platforms contribute most to each component.
#06	Top Tracks	Identified top-performing tracks along PC1 and PC2 axes to explore different types of success

PCA

- **What was dropped to prevent overfitting?**
  - Removed redundant metrics to ensure consistency and avoid multicollinearity.
  - PCA inherently drops components with negligible variance (minimal impact on Track Score)

- **What was standardized?**
  - We standardized numeric values such as stream counts, Youtube views, and playlist reach.
  - This is because PCA is sensitive to scale. Standardization ensures equal weighting.

Evaluation Methodology

Data Sets

- Spotify Top Songs 2024 from Kaggle with track score and platform metrics.

Metric Used

- Pearson correlation coefficients to identify influential platform metrics.
- PCA explained variance ratios to uncover important feature combinations.
- Manual inspection of trend spikes (e.g., TikTok or YouTube) aligned with Spotify jumps.

Challenges

- Missing or sparse TikTok data across many tracks.
- No timestamps for viral trend events, affecting temporal alignment.
- Differences in how platforms measure popularity (views vs streams vs likes).

Data set and Cleaning

- Remove commas in the numbers
- Log transform data to reduce skewing
- Remove null values
- Filter the release years with less than 5 songs
- Normalize the values to allow for ML models

```
# Clean and convert all relevant columns to numeric
for col in feature_cols:
    df[col] = df[col].astype(str).str.replace(',', '', regex=False)
    df[col] = pd.to_numeric(df[col], errors='coerce')
```

```
# Log-transform skewed features
skewed_cols = [
    'Spotify Streams', 'YouTube Views', 'TikTok Views',
    'Apple Music Playlist Count', 'AirPlay Spins', 'Shazam Counts'
]
for col in skewed_cols:
    df[col] = np.log1p(df[col])
```

```
# Drop NA
df = df.dropna(subset=['Track Score'] + feature_cols)
```

```
# Filter out years with too little data
year_counts = df['Release Year'].value_counts()
df = df[df['Release Year'].isin(year_counts[year_counts >= 5].index)]
```

Release Date vs Track Score

- Track score is directly tied with amount of Spotify listens in the year of 2024
- We want to know how streams on other platforms affects this score
- We want to also know how release date affects this score and we can see hints of how trends can impact this.

Objective

- Using the different track score as our output
- Using streams/views on platforms and release date as our input
- We are trying to predict the track score based on the release date using all the stream/view metrics as predictors
- Classification
- Used classification to give “popularity” value to tracks scores based on
- We used bins of 0-30 : Low, 30-60 : Medium, 60-750 : High,.

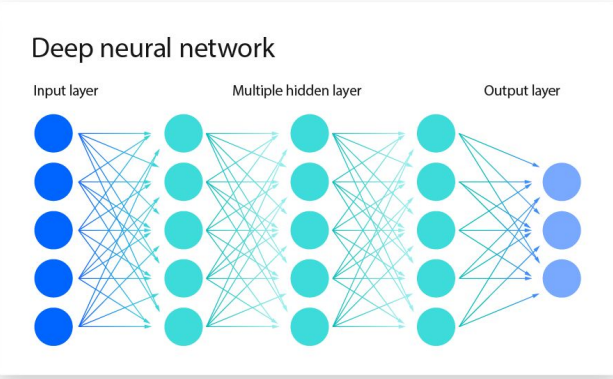
3 Machine Learning Models

- Feedforward Neural Network (FNN)
  - Great for modeling complex nonlinear relationships
  - Requires more tuning and training time
- Random Forest Regression Model
  - Robust with outliers
  - Slower and more expensive
- XGBoost - Extreme Gradient Boosting
  - Great for nonlinear and non scaled data
  - Harder to get correct parameters

FNN

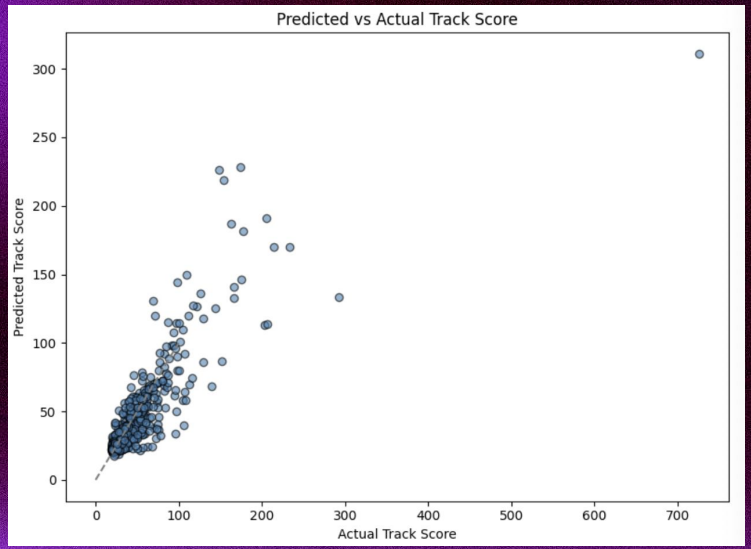
Implementation:

- 3 layers 128 -> 64 -> 1 neuron
- Batch normalization
- Dropout 30% for regularization
- Optimizer: Adam, lr=0.001
- Epochs: 1000
- Train/Test split: 80/20



FNN Results

- $R^2 = 0.7027$
- MAE = 9.31
- Accuracy 82.95%



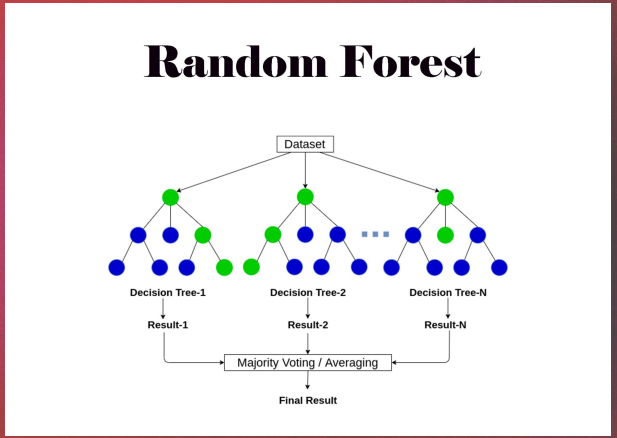

```
Neural Network R²: 0.7027, MAE: 9.31, Accuracy: 82.95%
```

```
=== Classification Report for FNN ===
              precision    recall  f1-score   support

        High      0.844     0.743     0.790       109
         Low      0.768     0.896     0.827       280
      Medium      0.712     0.604     0.654       225

    accuracy                          0.762       614
   macro avg      0.774     0.748     0.757       614
weighted avg      0.761     0.762     0.757       614
```

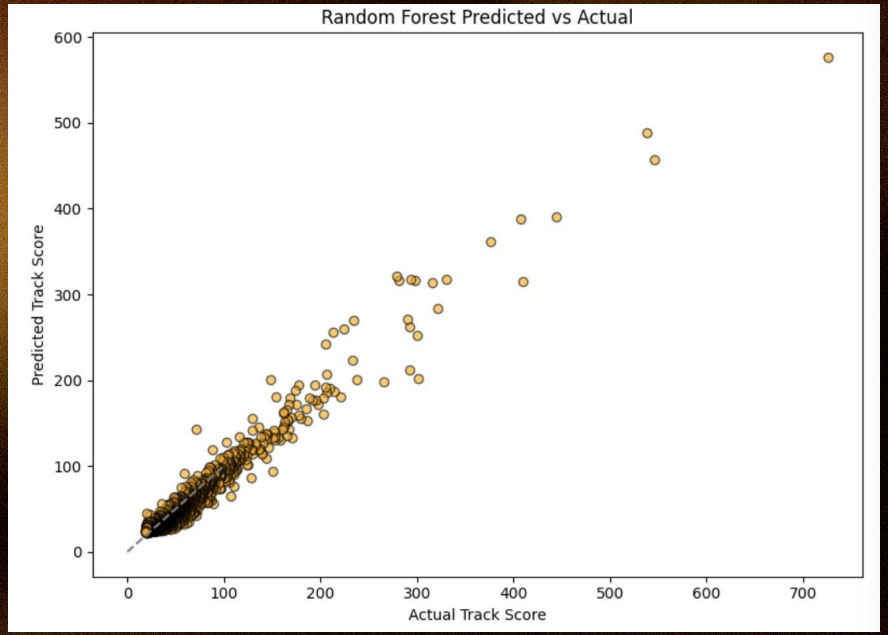
Random Forest

Implementation:

- 300 decision trees
- Max depth 12
- Random state = 42



Random Forest



- R^2: 0.9501
- MAE: 5.52
- Accuracy: 85.66%

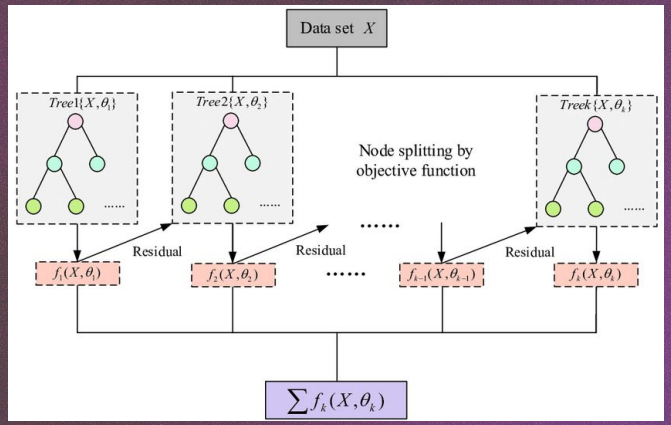
```
Random Forest R²: 0.9501, MAE: 5.52, Accuracy: 85.66%
```

```
Random Forest Classification Report:
              precision    recall  f1-score   support

        High       0.93      0.86      0.89       470
         Low       0.88      0.83      0.86      1478
      Medium       0.75      0.82      0.78      1118

    accuracy                           0.83      3066
   macro avg       0.85      0.84      0.84      3066
weighted avg       0.84      0.83      0.83      3066
```

XGBoost

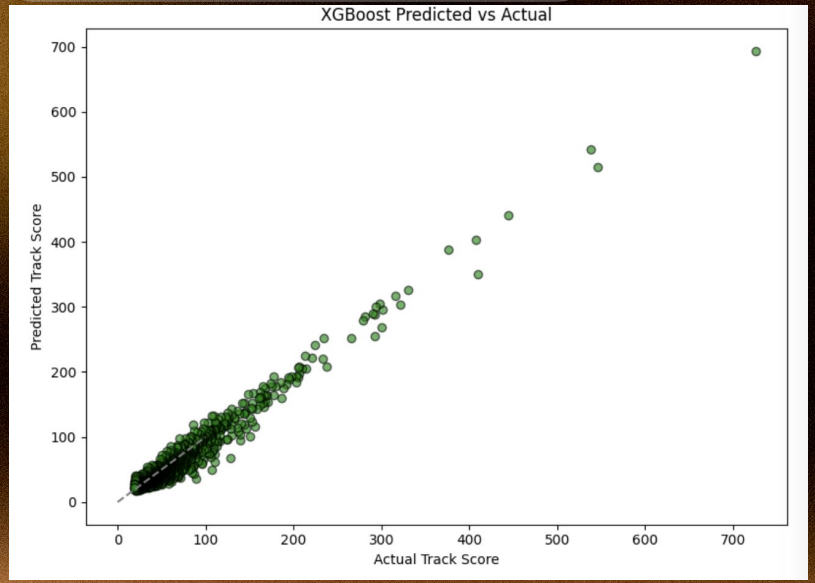
Implementation:

- Parameters
  - Number of Trees: 100, 300 trees
  - Learning rate 0.01, 0.05, 0.1
  - Max depth 3, 5, 7
- GridSearchCV
- Cv = 5
- Regularization: Built-in
- Optimization: Gradient Boosting



XGBoost Results

- R^2: 0.9569
- MAE: 5.69
- Accuracy: 85.06%



XGBoost R^2: 0.9569, MAE: 5.69, Accuracy: 85.06%

XGBoost Classification Report:

	precision	recall	f1-score	support
High	0.90	0.82	0.86	470
Low	0.87	0.78	0.82	1478
Medium	0.69	0.81	0.74	1118
accuracy			0.80	3066
macro avg	0.82	0.80	0.81	3066
weighted avg	0.81	0.80	0.80	3066

Model Comparison

Model	R^2	MAE	Accuracy
FNN	0.7027	9.31	83.85%
Random Forest	0.9501	5.52	85.66%
XGBoost	0.9569	5.59	85.06%

Future Work

- Go back to the FNN model and explore different parameterization/tuning
- For XGBoost give more parameters for GridSearchCV due to the relatively low set of 18 combinations in the current implementation
- Fine turn current models and/or explore other models that would work best with our dataset
- Do analysis and predictions of more datasets to see how good our models are

References

N. Elgiriyewithana, "Most Streamed Spotify Songs 2024," Kaggle, 2024. [Online]. Available: https://www.kaggle.com/datasets/nelgiriyewithana/most-streamed-spotify-songs-2024