# Podcast Listening Time Prediction

May 2, 2025

Elvis - Suguru - Dana

# Agenda

## Project Overview

Prediction of podcast listening time using a Kaggle dataset.

## Process

We cleaned the data, explored it, engineered features, and trained regression models.

## Findings

One feature had a dominant influence on predictions.

## Challenges

We faced communication gaps and production capacity limitations during the project.

# Project Overview

- **Goal** : Predict podcast episode listening time
- **Dataset** : Kaggle — Podcast Listening Time Prediction
- **Focus** : Understand which features influence user listening behavior
- **Approach** : Build a regression model to forecast listening time
- **Data includes:** episode metadata, publication info, sentiment, popularity scores, etc.

# Dataframe

*Shape: (750000, 12)*

Numerical Features | Categorical Features | Important Feature | Target Feature

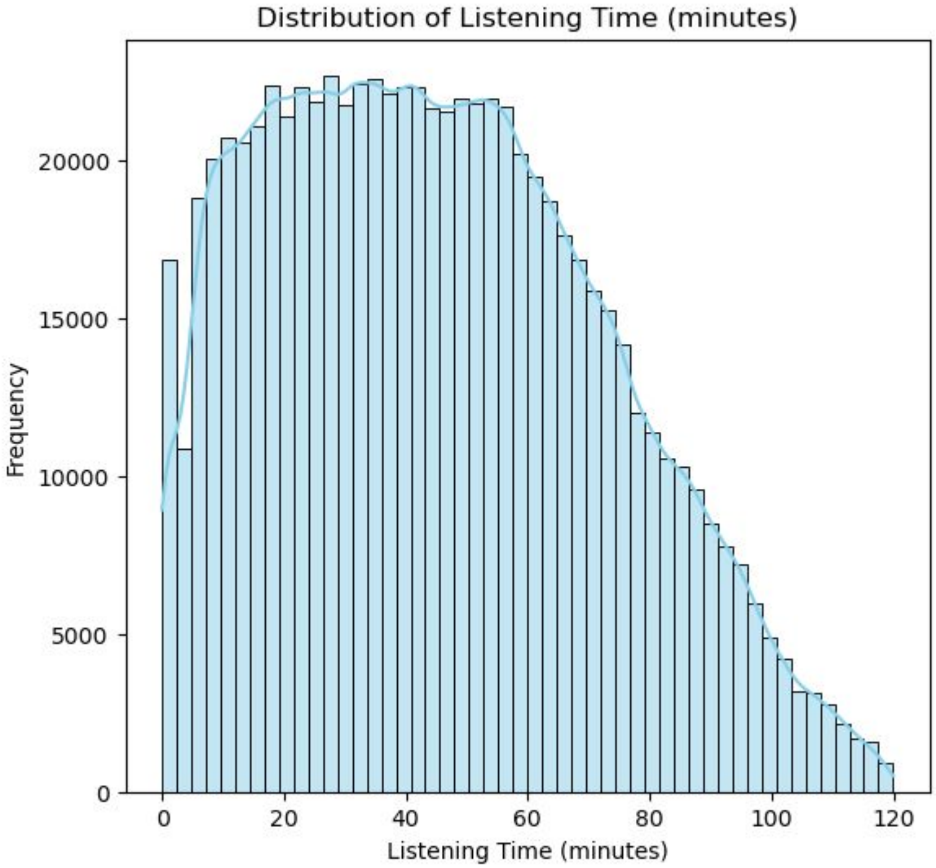
| id | Podcast_Name | Episode_Title | Episode_Length_minutes | Genre | Host_Popularity_percentage | Publication_Day | Publication_Time | Guest_Popularity_percentage | Number_of_Ads | Episode_Sentiment | Listening_Time_minutes |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | Mystery Matters | Episode 98 | NaN | True Crime | 74.81 | Thursday | Night | NaN | 0.0 | Positive | 31.41998 |
| 1 | Joke Junction | Episode 26 | 119.8 | Comedy | 66.95 | Saturday | Afternoon | 75.95 | 2.0 | Negative | 88.01241 |
| 2 | Study Sessions | Episode 16 | 73.9 | Education | 69.97 | Tuesday | Evening | 8.97 | 0.0 | Negative | 44.92531 |

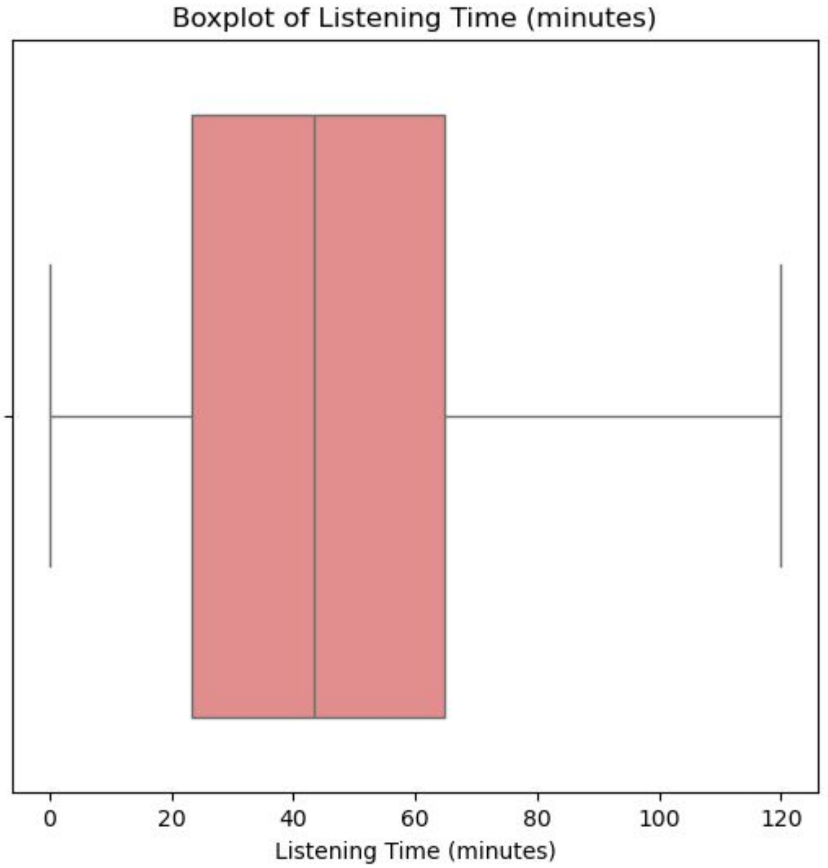
# EDA & Data Preprocessing

Exploratory Data Analysis

- Initial dataset exploration and variable assessment
- Correction of outliers that did not align with expected ranges
- Imputation of missing values to ensure data completeness

# Target Feature Behaviour

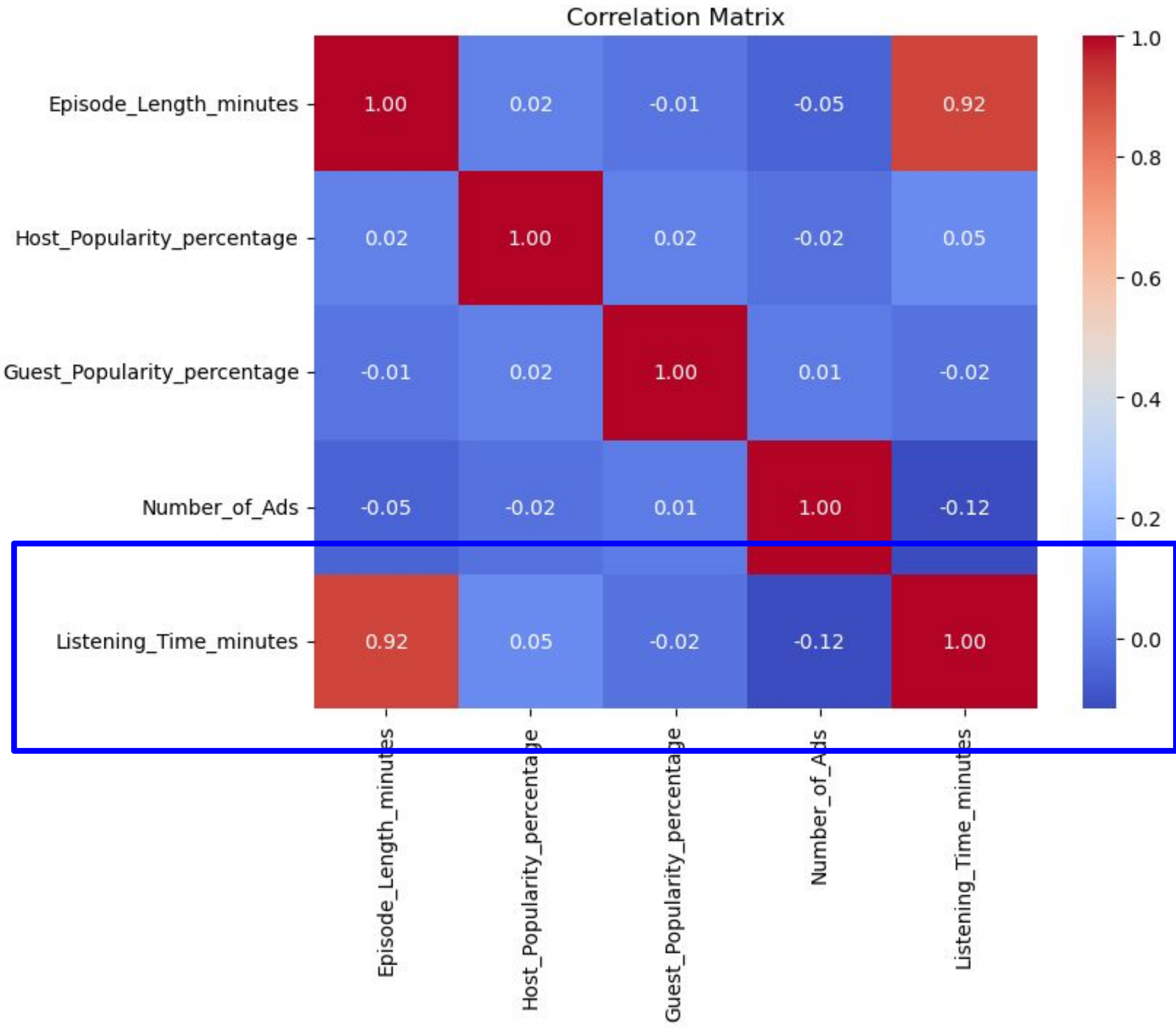


# Numerical Features

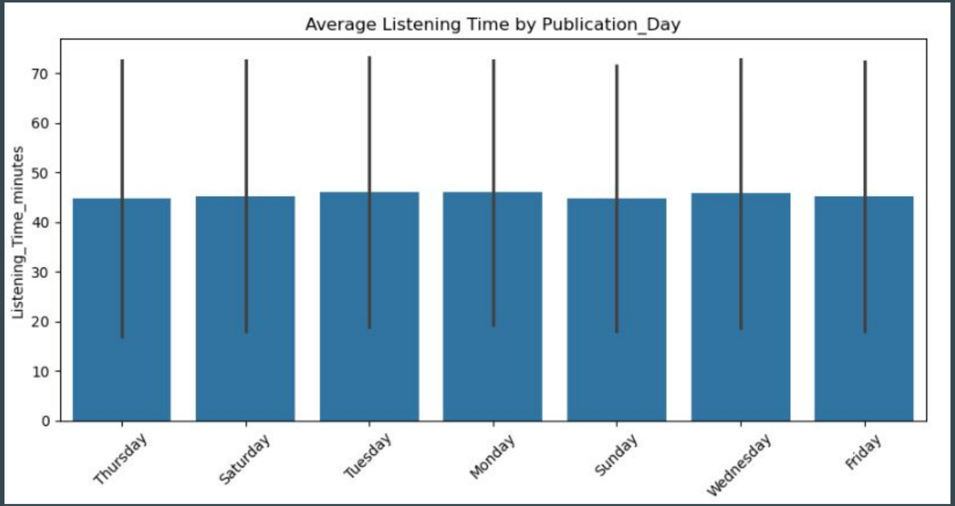
We noticed that the main feature influencing the target is *Episode_Length_minutes,* showing a 92% correlation.

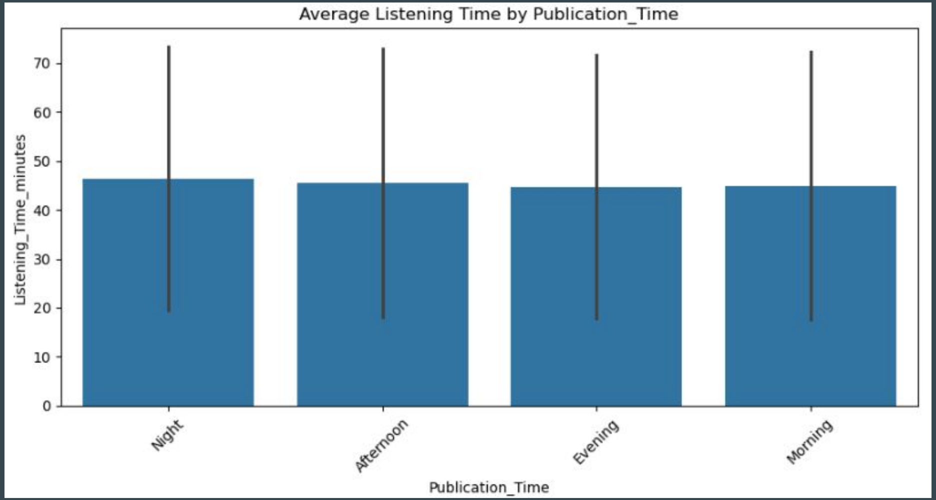
This makes sense, as the total listening time largely depends on the length of each episode.



# Publication Day & Time

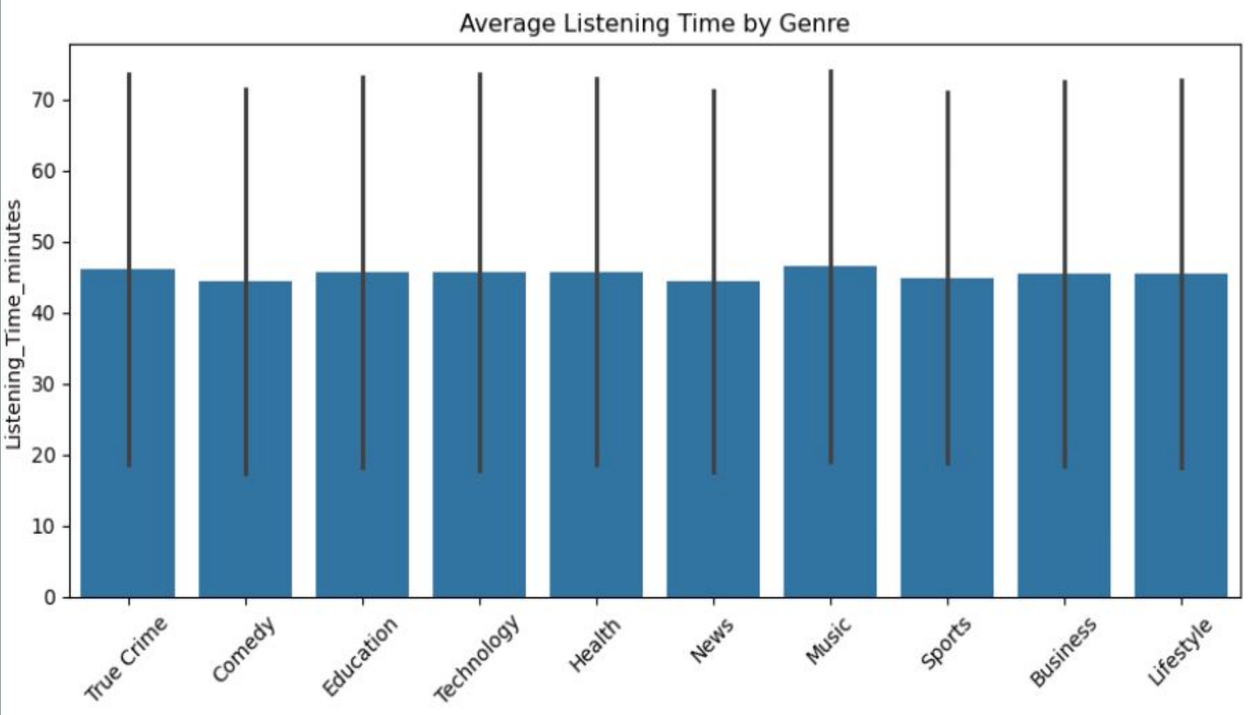
- ➔ No clear trend was observed across days or times of publication.
- ➔ These features did not significantly contribute to predicting listening time.
- ➔ Therefore, they were not prioritized during feature selection.



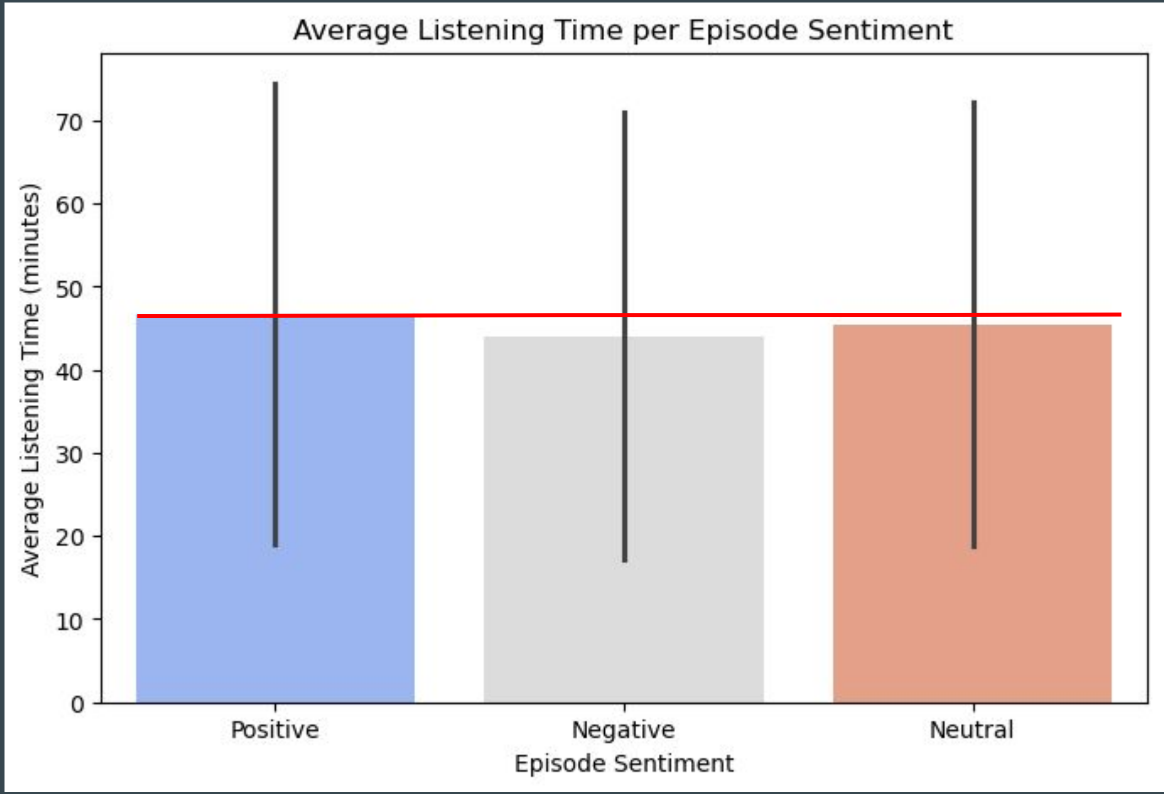
# Listening Time by Genre

- ➔ The average listening time is <u>relatively consistent</u> across genres.
- ➔ <u>No specific genre stands out</u> as significantly more or less listened to.
- ➔ This suggests *genre does not strongly influence listening time predictions.*



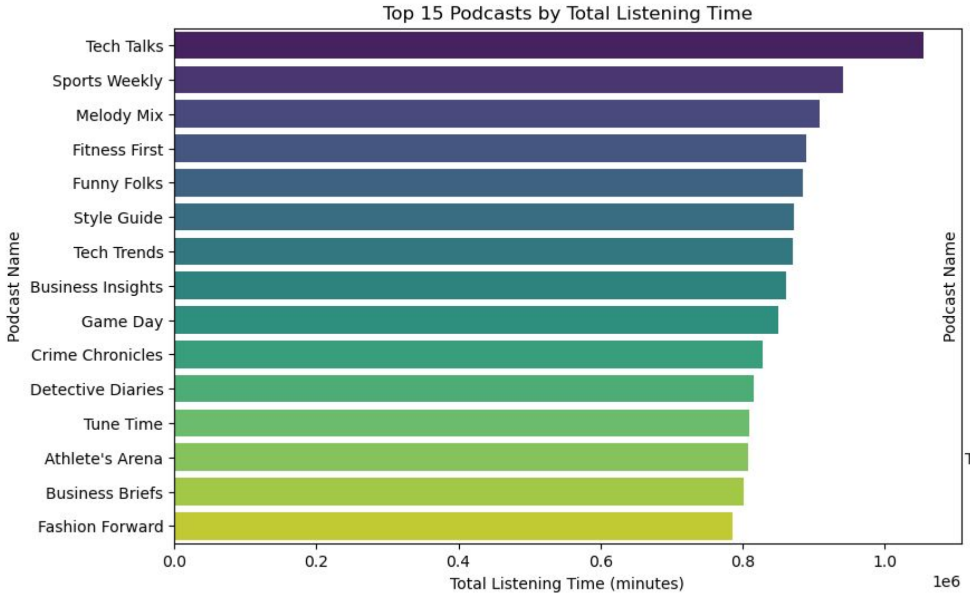
# Listening Time by Sentiment

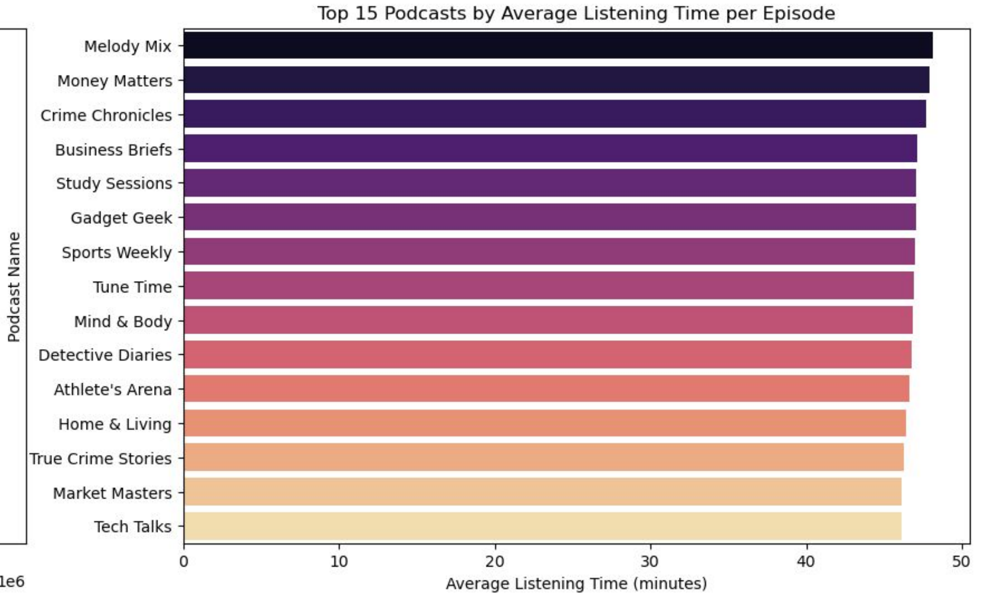
- ➔ Sentiment categories show similar average listening times.
- ➔ Episodes with **positive sentiment had a slightly higher average listening time.**
- ➔ Overall, sentiment had limited impact on prediction performance.



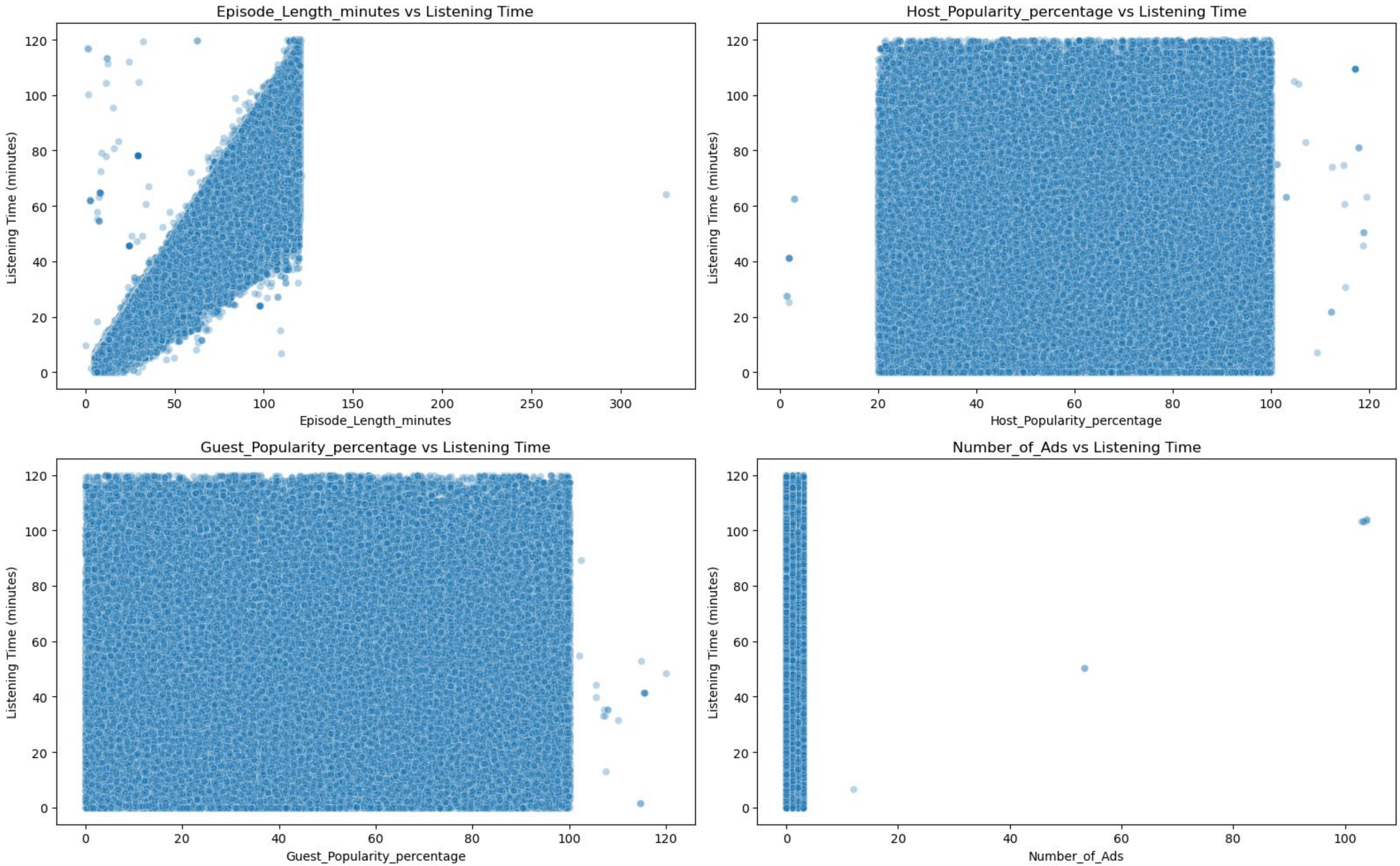
# Podcast Performance Insights

➔ "Tech Talks", "Sports Weekly", and "Melody Mix" led in total listening time.
➔ "Money Matters" and "Crime Chronicles" had the highest average listening time per episode, showing strong engagement.
➔ Due to many missing values, features like Episode_Length_minutes and Guest_Popularity_percentage were not used directly in modeling but helped create derived metrics during preprocessing.



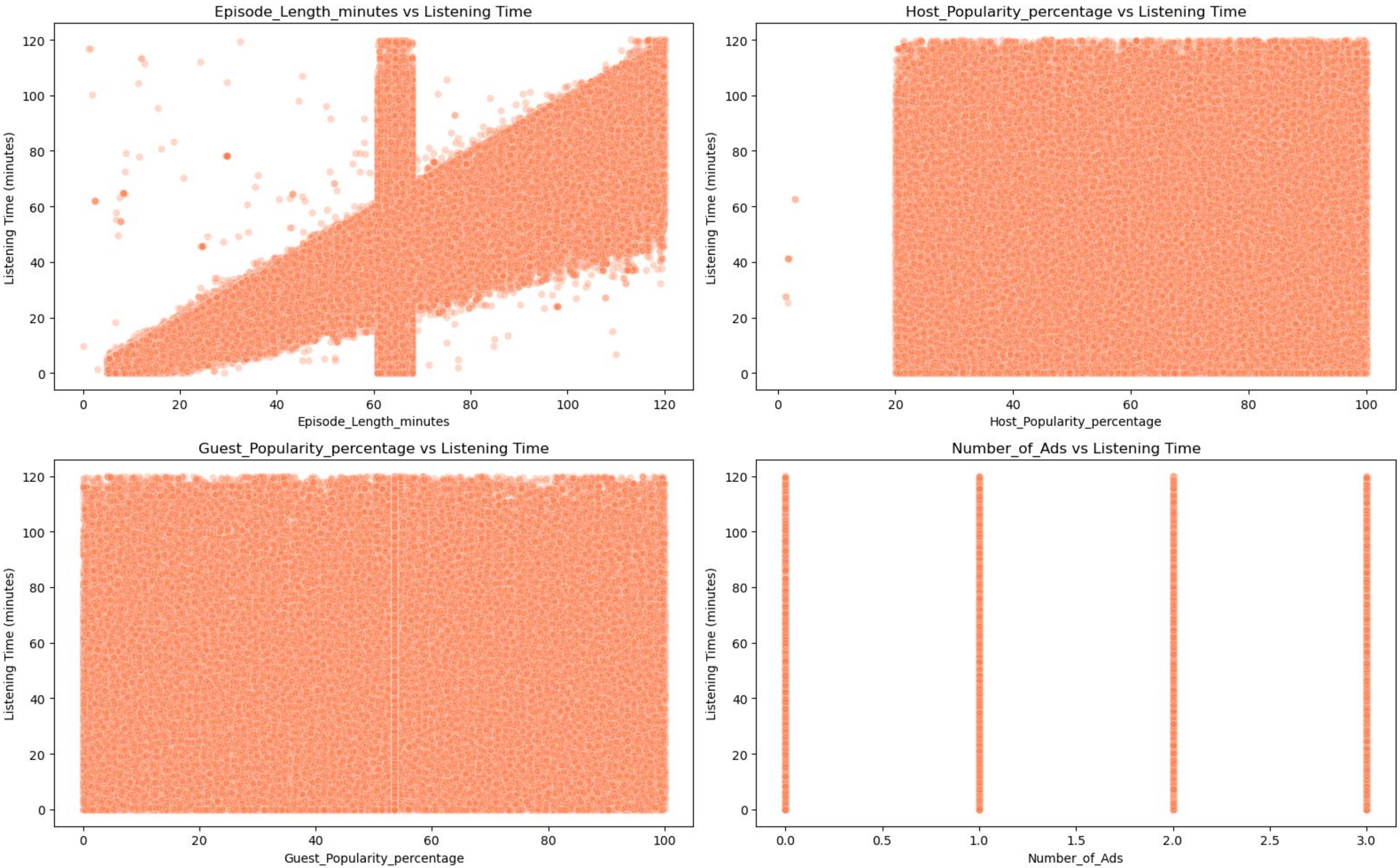



# OUTLIERS



Episode_Length_minutes vs Listening Time
Listening Time (minutes)
Episode_Length_minutes
Host_Popularity_percentage vs Listening Time
Listening Time (minutes)
Host_Popularity_percentage
Guest_Popularity_percentage vs Listening Time
Listening Time (minutes)
Guest_Popularity_percentage
Number_of_Ads vs Listening Time
Listening Time (minutes)
Number_of_Ads

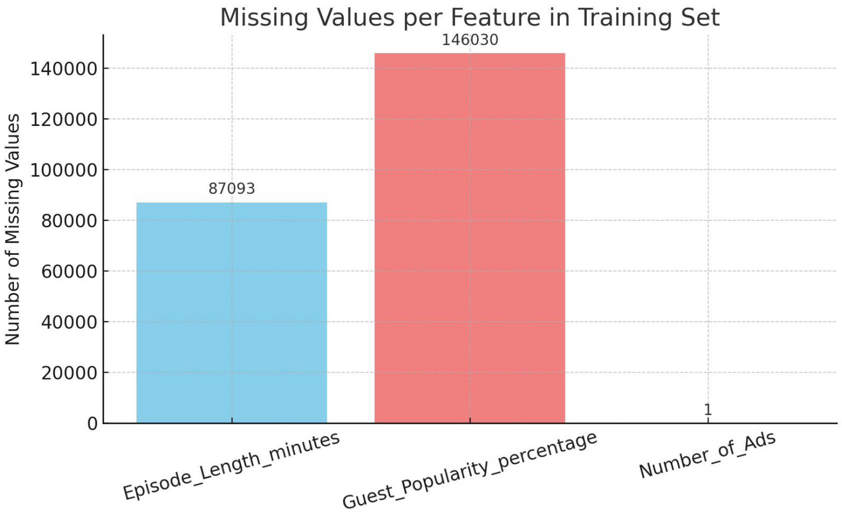
AFTER HANDLING



# MISSING VALUES

```
Missing values in test:
 Episode_Length_minutes          28736
Guest_Popularity_percentage     48832

Missing values in train:
 Episode_Length_minutes          87093
Guest_Popularity_percentage    146030
Number_of_Ads                       1
```




**Episode_Length_minutes** was our most important feature, so we imputed missing values carefully using the Podcast Name and Genre to maintain consistency.

**Guest_Popularity_percentage had ~19.47%** missing values. These likely indicated no guest rather than missing data, so we created a "No Guest" category and imputed the remaining nulls with the mean.

**Number_of_Ads** had only one missing value and was filled with the mode.

*Challenge of communication on deciding how to handle this*

# Feature Engineering Highlights

## Ad_Density

- ➔ Calculated as Number of Ads / Episode Length
- ➔ Reflects ad concentration and its potential effect on listener engagement
- ➔ Helps detect patterns related to ad saturation and drop-off behavior

## Guest_Popularity_missing

- ➔ Created a binary feature to indicate episodes without a guest
- ➔ Addressed the ambiguity in missing Guest_Popularity_percentage values
- ➔ Provided a clearer signal during modeling and preserved meaningful variation

# Regression Models

Training the Data

## Modeling Approach

Since our target variable (Listening_Time_minutes) is continuous, we focused exclusively on regression models.

The dataset's structure and the prediction goal aligned best with supervised regression techniques.

## Models Trained

- Linear Regression – as a baseline model
- Random Forest Regressor – for non-linear relationships and feature importance
- Gradient Boosting Regressor – for improved performance through boosting
- XGBoost Regressor – a powerful gradient boosting implementation

# Testing

***Final Model Results***

***Best Result we Obtained***

| Model | RMSE | $R^2$ |
|---|---|---|
| Linear Regression (Initial) | 13.3071 | 0.7593 |
| Linear Regression (Final Preprocessing) | 13.2846 | 0.7602 |
| Gradient Boosting (First Run) | 13.1363 | 0.7655 |
| Gradient Boosting (Tuned) | 13.0846 | 0.7673 |
| Gradient Boosting (Feature Importance) | 13.11 | 0.7665 |
| Gradient Boosting | 13.1363 | 0.7654 |
| Gradient Boosting (Tuned - Final) | 13.0636 | 0.7681 |
| Gradient Boosting (Feature Based) | 13.09 | 0.7672 |
| Gradient Boosting (Final) | 13.134 | 0.7656 |

| Model | RMSE | $R^2$ |
|---|---|---|
| Random Forest (Initial) | 12.7906 | 0.7776 |
| Random Forest (First Run) | 12.7821 | 0.7779 |
| Random Forest (Important Features) | 13.1374 | 0.7654 |
| Random Forest (Hyperparameters) | 13.6069 | 0.7484 |
| Random Forest (Encoding Strategy) | 12.7916 | 0.7776 |
| Random Forest (Final Preprocessed) | 12.7693 | 0.7784 |
| XGBoost (Initial) | 13.14 | 0.7677 |
| XGBoost (Tuned) | 13.16 | 0.7645 |
| XGBoost (Final) | 13.04 | 0.7689 |

# Gradient Boost

## First attempt

Trained the initial model with default parameters — performance was promising.

## Improving

Applied RandomizedSearchCV to find the best parameters — results improved slightly.

Explored feature importance to simplify the model — similar accuracy with fewer inputs.

## Observations

Gradient Boosting trained faster than Random Forest but did not outperform it.

We had to split the dataset due to memory limitations, which restricted further tuning..

# XGBoost

## First attempt

Ran the model with default parameters — performance was solid and comparable to Gradient Boosting.

## Improving

Applied hyperparameter tuning — results got slightly worst.

Also tested optimized preprocessing, but improvements were minimal.

## Observations

XGBoost was efficient and stable, but did not outperform Random Forest or tuned Gradient Boosting.

Improvements were limited despite tuning, and results with full data got minimal better..

# Random Forest

## First attempt

Ran the base model with default settings — served as a strong baseline.

## Improving

RandomizedSearch for tuning, results got slight bad performance.

Tested feature importance to reduce input size — not relevant results.

Tried improving encoding, but it didn't enhance results.

## Observations

Random Forest delivered solid results, the best of out all models, but had long training times, limiting our ability to experiment further and optimize the model.

# Final Results & Key Insights

*After testing multiple models and tuning strategies, Random Forest with improved preprocessing achieved the best results, outperforming others in both RMSE and $R^2$.*

| Model | RMSE | $R^2$ |
|---|---|---|
| Linear Regression (Final) | 13.2846 | 0.7602 |
| Gradient Boosting (Final) | 13.1340 | 0.7656 |
| Gradient Boosting (Tuned Final) | 13.0636 | 0.7681 |
| Gradient Boosting (Feature Based) | 13.0900 | 0.7672 |
| Random Forest (Final Preprocessed) | **12.7693** | **0.7784** |
| XGBoost (Final) | 13.0400 | 0.7689 |

# Kaggle Competition

Out of deadline Result

Last submitted Result

| Submission and Description | Private Score | Public Score |
|---|---|---|
| **submissionRF.csv**<br>Complete (after deadline) · 18s ago | 12.80040 | 12.86694 |

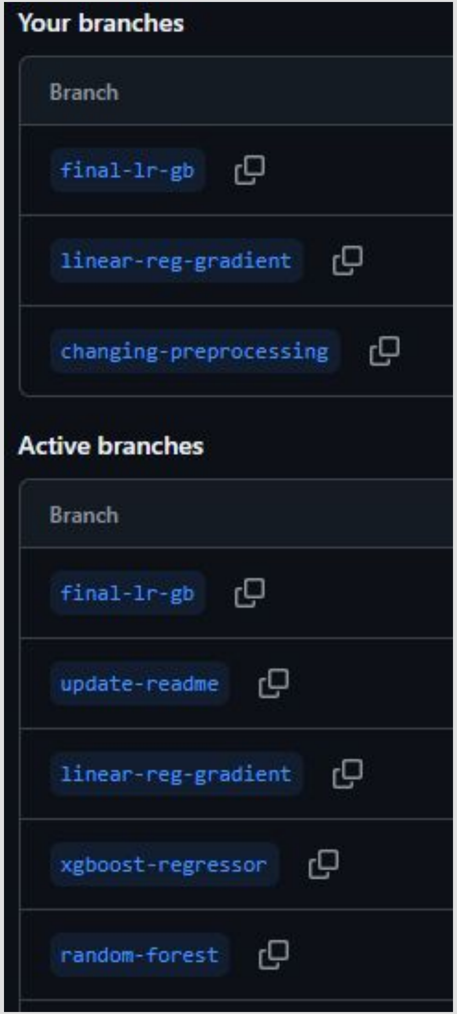
YOUR RECENT SUBMISSION

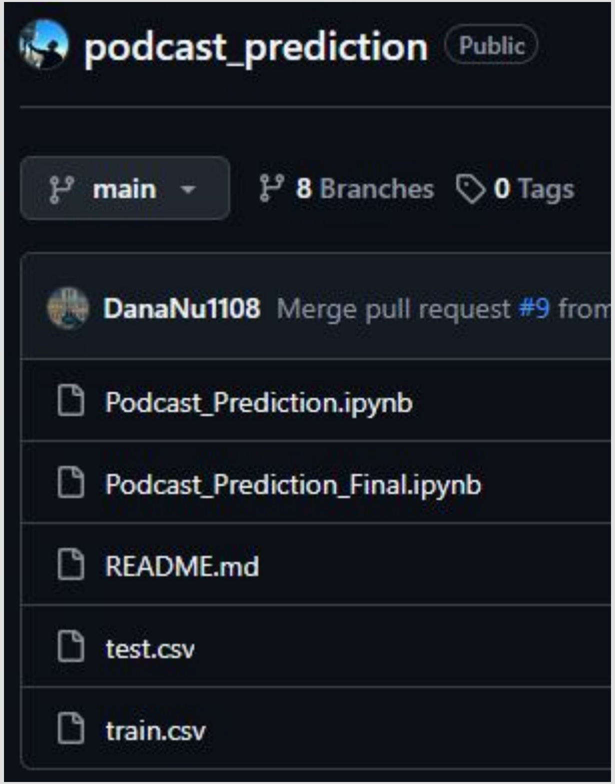
**submission.csv**
Submitted by SO · Submitted 28 seconds ago

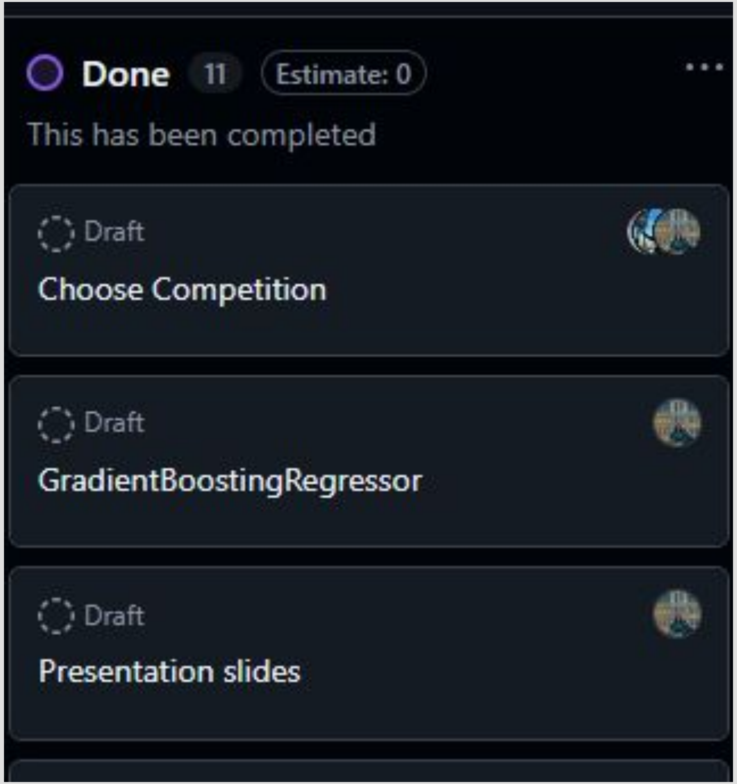
Score: 12.90504
Public score: 12.99744

| | | |
|---|---|---|
| **submission2.csv**<br>Complete (after deadline) · 18s ago | 13.42432 | 13.53060 |
| **submissionxgboost.csv**<br>Complete (after deadline) · 4h ago | 13.03634 | 13.16022 |

# GITHUB VIEW


Your branches
Branch
final-lr-gb
linear-reg-gradient
changing-preprocessing
Active branches
Branch
final-lr-gb
update-readme
linear-reg-gradient
xgboost-regressor
random-forest


podcast_prediction
Public
main
8 Branches
0 Tags
DanaNu1108
Merge pull request #9 from
Podcast_Prediction.ipynb
Podcast_Prediction_Final.ipynb
README.md
test.csv
train.csv


Done
11
Estimate: 0
This has been completed
Draft
Choose Competition
Draft
GradientBoostingRegressor
Draft
Presentation slides

# Conclusion

*Random Forest was the best-performing model, delivering the highest $R^2$ and lowest RMSE after improved preprocessing.*

*Throughout the project, we analyzed the dataset from multiple angles to better understand what drives podcast listening time. We discovered that Episode_Length_minutes was the most influential feature, while genre, sentiment, and publication timing had minimal predictive power.*

*To enhance model performance, we engineered features like Ad_Density and a No_Guest, and applied various preprocessing strategies to handle missing data and reduce noise.*

*This project showed that beyond model tuning, deep data understanding and iterative processing are key to building effective predictive models.*

# Final Thoughts

*This project was a valuable experience in applying machine learning to real-world data. By combining data exploration, thoughtful preprocessing, and model experimentation, we gained both technical insights and a deeper understanding of podcast engagement patterns.*

*Our work highlighted the importance of staying curious, testing different approaches, and letting the data guide decision-making.*

***Thank you for your time!***