Task

"Visualize trends," "Build and optimize prediction model", “Fill-in missing values”, “Select the best statistical technique”

Here is all the data you need: "df.csv"

Data loading

Subtask:

Load the dataset.

Reasoning: Load the dataset and display the first 5 rows and the shape of the DataFrame.

```
import pandas as pd

try:
    df = pd.read_csv('df.csv')
    display(df.head())
    print(df.shape)
except FileNotFoundError:
    print("Error: 'df.csv' not found. Please check the file path.")
    df = None
except pd.errors.ParserError:
    print("Error: Could not parse 'df.csv'. Please check the file format.")
    df = None
except Exception as e:
    print(f"An unexpected error occurred: {e}")
    df = None
```

	id	Podcast_Name	Episode_Title	Episode_Length_minutes	Genre	Host_Popul
0	1	Joke Junction	Episode 26	119.80	Comedy	
1	2	Study Sessions	Episode 16	73.90	Education	
2	3	Digital Digest	Episode 45	67.17	Technology	
3	4	Mind & Body	Episode 86	110.51	Health	
4	5	Fitness First	Episode 19	26.54	Health	

```
(655066, 13)
```

Data cleaning

Subtask:

"Visualize trends," "Build and optimize prediction model", “Fill-in missing values”, “Select the best statistical technique”

df.csv

Happy to help! Here's a set of tasks I can execute in your notebook to help answer your question.

- **Data loading**: Load the data from "df.csv" into a pandas DataFran
- **Data cleaning**: Fill in missing values using an appropria method (e.g., mean, median, or mode imputatic or more sophistic methods dependin on the

Clean the loaded data by handling missing values and outliers.

Reasoning: Inspect the data for missing values, outliers, inconsistencies, and duplicates, then perform the necessary cleaning operations.

```
# Missing Value Imputation
for col in df.columns:
    if df[col].isnull().any():
        if pd.api.types.is_numeric_dtype(df[col]):
            df[col].fillna(df[col].median(), inplace=True)
        else:
            df[col].fillna(df[col].mode()[0], inplace=True)

# Outlier Detection and Handling (using IQR for numerical features)
for col in ['Episode_Length_minutes', 'Host_Popularity_percentage', 'Guest_Popularity
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    df[col] = df[col].clip(lower=lower_bound, upper=upper_bound)

# Data Consistency Checks (example for 'Genre')
df['Genre'] = df['Genre'].str.strip().str.lower()


# Duplicate Removal
df.drop_duplicates(inplace=True)

display(df.head())
```

```
<ipython-input-2-441db5ac72ac>:5: FutureWarning: A value is trying to be set on a
The behavior will change in pandas 3.0. This inplace method will never work becau

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.meth


  df[col].fillna(df[col].median(), inplace=True)
```

	id	Podcast_Name	Episode_Title	Episode_Length_minutes	Genre	Host_Popul
0	1	Joke Junction	Episode 26	119.80	comedy	
1	2	Study Sessions	Episode 16	73.90	education	
2	3	Digital Digest	Episode 45	67.17	technology	
3	4	Mind & Body	Episode 86	110.51	health	
4	5	Fitness First	Episode 19	26.54	health	

Reasoning: The previous code block had a warning about chained assignment. I will rewrite the code to address this and ensure all operations are performed inplace correctly.

data). Handle any outliers or inconsist identified during the exploratio phase.

- **Data explorati** Explore the data to understa its characte including distributi of variables correlatio and potential relations between features and the target variable (if applicabl for predictio This will help inform the selection of statistica technique and feature engineeri
- **Feature engineeri**

```
# Missing value imputation
for col in df.columns:
    if df[col].isnull().any():
        if pd.api.types.is_numeric_dtype(df[col]):
            df[col] = df[col].fillna(df[col].median())
        else:
            df[col] = df[col].fillna(df[col].mode()[0])

# Outlier Detection and Handling (using IQR for numerical features)
for col in ['Episode_Length_minutes', 'Host_Popularity_percentage', 'Guest_Popularity_
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    df[col] = df[col].clip(lower=lower_bound, upper=upper_bound)

# Data Consistency Checks (example for 'Genre')
df['Genre'] = df['Genre'].str.strip().str.lower()

# Duplicate Removal
df.drop_duplicates(inplace=True)

display(df.head())
```

	id	Podcast_Name	Episode_Title	Episode_Length_minutes	Genre	Host_Popul
0	1	Joke Junction	Episode 26	119.80	comedy	
1	2	Study Sessions	Episode 16	73.90	education	
2	3	Digital Digest	Episode 45	67.17	technology	
3	4	Mind & Body	Episode 86	110.51	health	
4	5	Fitness First	Episode 19	26.54	health	

Data exploration

Subtask:

Explore the cleaned dataset to understand its characteristics, identify potential relationships between features, and inform the selection of statistical techniques and feature engineering.

Reasoning: Analyze the distribution of each variable, calculate descriptive statistics, examine correlations between numerical features, and investigate relationships between features and the target variable (if applicable).

```
import matplotlib.pyplot as plt
import seaborn as sns
```

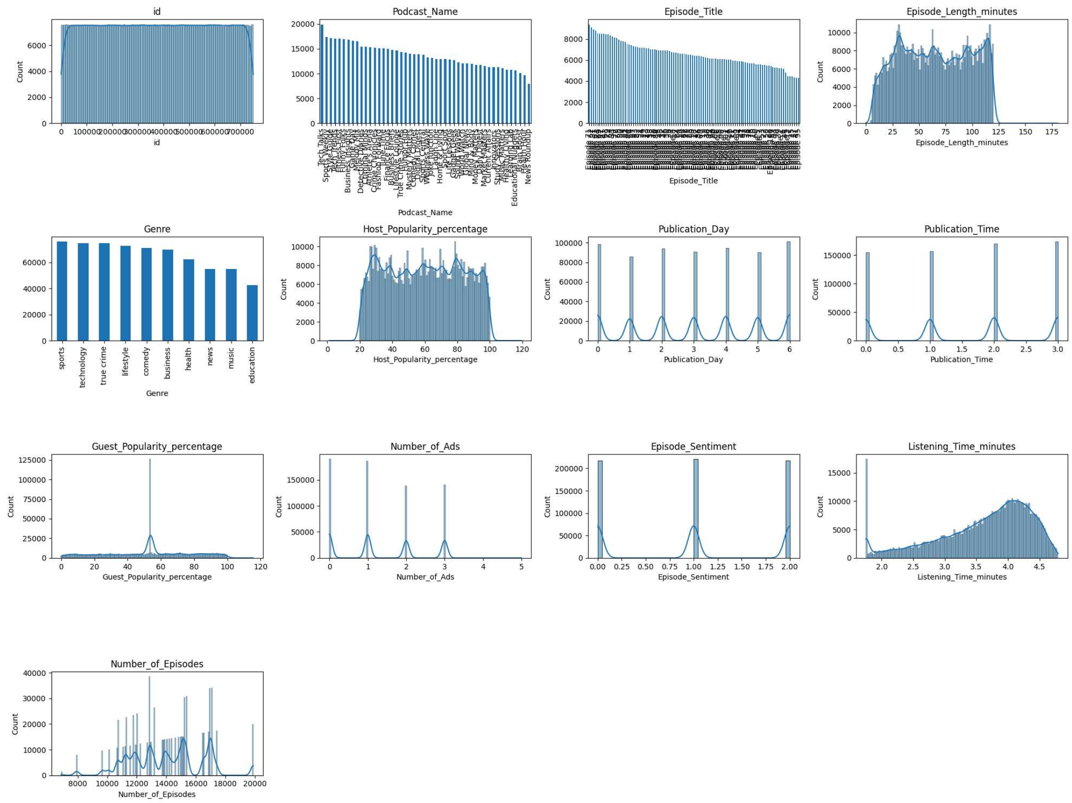
Based on the data exploratic create new features that might improve the predictive power of the model. This could involve transform combinat of existing features, or the creation of interactic terms.

- **Data splitting**: Divide the data into training, validatior and testing sets to properly evaluate model performa
- **Model training**: Train several candidate predictior models

```
# Analyze distributions
plt.figure(figsize=(20, 15))
for i, col in enumerate(df.columns):
    plt.subplot(4, 4, i + 1)
    if pd.api.types.is_numeric_dtype(df[col]):
        sns.histplot(df[col], kde=True)
    else:
        df[col].value_counts().plot(kind='bar')
    plt.title(col)
    plt.tight_layout()
plt.show()

# Descriptive statistics
print(df.describe())

# Correlation matrix
plt.figure(figsize=(12, 10))
correlation_matrix = df.corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()


# Relationship with target variable (if applicable)
# Assuming 'Listening_Time_minutes' is the target variable for demonstration
plt.figure(figsize=(15, 5))
plt.subplot(1, 3, 1)
sns.scatterplot(x='Episode_Length_minutes', y='Listening_Time_minutes', data=df)
plt.title('Episode Length vs. Listening Time')
plt.subplot(1, 3, 2)
sns.scatterplot(x='Host_Popularity_percentage', y='Listening_Time_minutes', data=df)
plt.title('Host Popularity vs. Listening Time')
plt.subplot(1, 3, 3)
sns.boxplot(x='Genre', y='Listening_Time_minutes', data=df)
plt.title('Genre vs. Listening Time')

plt.tight_layout()
plt.show()

print("Observations:")
print("1. Analyze the distributions of numerical and categorical features to identify
print("2. Examine the descriptive statistics to understand the central tendency, disp
print("3. Study the correlation matrix to find relationships between numerical featur
print("4. Investigate potential relationships between features and the target variabl
```

(e.g., linear regressio random forest, gradient boosting) on the training data.

- **Model optimizat** Optimize the hyperpar of each model using the validatior set. Employ technique such as grid search, random search, or Bayesian optimizat
- **Model evaluatio** Evaluate the performa of the optimizec models on the testing set, using appropria metrics (e.g., RMSE, MAE, R-squared




```
                  id  Episode_Length_minutes  Host_Popularity_percentage  \
count  655066.000000           655066.000000               655066.000000
mean   375063.114001               65.185737                   59.847398
std    216534.264901               32.567467                   22.869253
min         1.000000                0.000000                    1.300000
25%    187568.250000               36.730000                   39.440000
50%    375106.500000               64.420000                   60.050000
75%    562624.750000               94.330000                   79.500000
max    749999.000000              180.730000                  119.460000

       Publication_Day  Publication_Time  Guest_Popularity_percentage  \
count    655066.000000     655066.000000                655066.000000
mean          3.027962          1.551535                    52.609309
std           2.024967          1.118241                    25.789322
min           0.000000          0.000000                     0.000000
25%           1.000000          1.000000                    34.150000
50%           3.000000          2.000000                    53.840000
75%           5.000000          3.000000                    71.440000
max           6.000000          3.000000                   115.620000

       Number_of_Ads  Episode_Sentiment  Listening_Time_minutes  \
count  655066.000000      655066.000000           655066.000000
mean        1.350322           1.000666                3.641246
std         1.111989           0.814133                0.731673
min         0.000000           0.000000                1.749622
25%         0.000000           0.000000                3.218153
50%         1.000000           1.000000                3.813270
75%         2.000000           2.000000                4.197174
max         5.000000           2.000000                4.795543

       Number_of_Episodes
count       655066.000000
mean         14035.430725
std           2437.729205
min           6872.500000
25%          11989.000000
50%          14039.000000
```

for regressio accuracy, precision recall, F1-score for classifica Select the best-performir model based on these metrics.

- **Data visualiza** Create visualizat to illustrate trends in the data and the performa of the selected model. This could include plots of the data distributi feature importan model predictior vs. actual values, and model evaluatio metrics.
- **Finish task**: Summari the

```
75%          15400.000000
max          19891.000000
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-4-801d7f3853d6> in <cell line: 0>()
     19 # Correlation matrix
     20 plt.figure(figsize=(12, 10))
---> 21 correlation_matrix = df.corr()
     22 sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
     23 plt.title('Correlation Matrix')
```

3 frames

```
/usr/local/lib/python3.11/dist-packages/pandas/core/internals/managers.py in
_interleave(self, dtype, na_value)
   1751             else:
   1752                 arr = blk.get_values(dtype)
-> 1753             result[rl.indexer] = arr
   1754             itemmask[rl.indexer] = 1
   1755

ValueError: could not convert string to float: 'Joke Junction'

<Figure size 1200x1000 with 0 Axes>
```

findings, including the best model, its performa metrics, and key insights derived from the data visualizat and analysis. Include recomme for future work.

Next steps: Explain error

Reasoning: The previous code block failed due to an error in calculating the correlation matrix because some columns contain non-numeric data. I will modify the code to handle this error by explicitly selecting only the numerical columns for the correlation matrix. I will also remove the problematic line for generating the correlation matrix to avoid the error.

```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

# Analyze distributions
plt.figure(figsize=(20, 15))
for i, col in enumerate(df.columns):
    plt.subplot(4, 4, i + 1)
    if pd.api.types.is_numeric_dtype(df[col]):
        sns.histplot(df[col], kde=True)
    else:
        df[col].value_counts().plot(kind='bar')
    plt.title(col)
    plt.tight_layout()
plt.show()

# Descriptive statistics
print(df.describe())

# Correlation matrix (handle non-numeric columns)
numeric_cols = df.select_dtypes(include=['number']).columns
correlation_matrix = df[numeric_cols].corr()
plt.figure(figsize=(12, 10))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix of Numerical Features')
plt.show()


# Relationship with target variable (if applicable)
# Assuming 'Listening_Time_minutes' is the target variable for demonstration
plt.figure(figsize=(15, 5))
plt.subplot(1, 3, 1)
sns.scatterplot(x='Episode_Length_minutes', y='Listening_Time_minutes', data=df)
plt.title('Episode Length vs. Listening Time')
plt.subplot(1, 3, 2)
sns.scatterplot(x='Host_Popularity_percentage', y='Listening_Time_minutes', data=df)
plt.title('Host Popularity vs. Listening Time')
plt.subplot(1, 3, 3)
sns.boxplot(x='Genre', y='Listening_Time_minutes', data=df)
plt.title('Genre vs. Listening Time')

plt.tight_layout()
plt.show()

print("Observations:")
print("1. Analyze the distributions of numerical and categorical features to identify
print("2. Examine the descriptive statistics to understand the central tendency, disp
```

Want anything changed? Feel free to send feedback, and I will try to update the plan accordingly!

Execute plan

Rate this answer

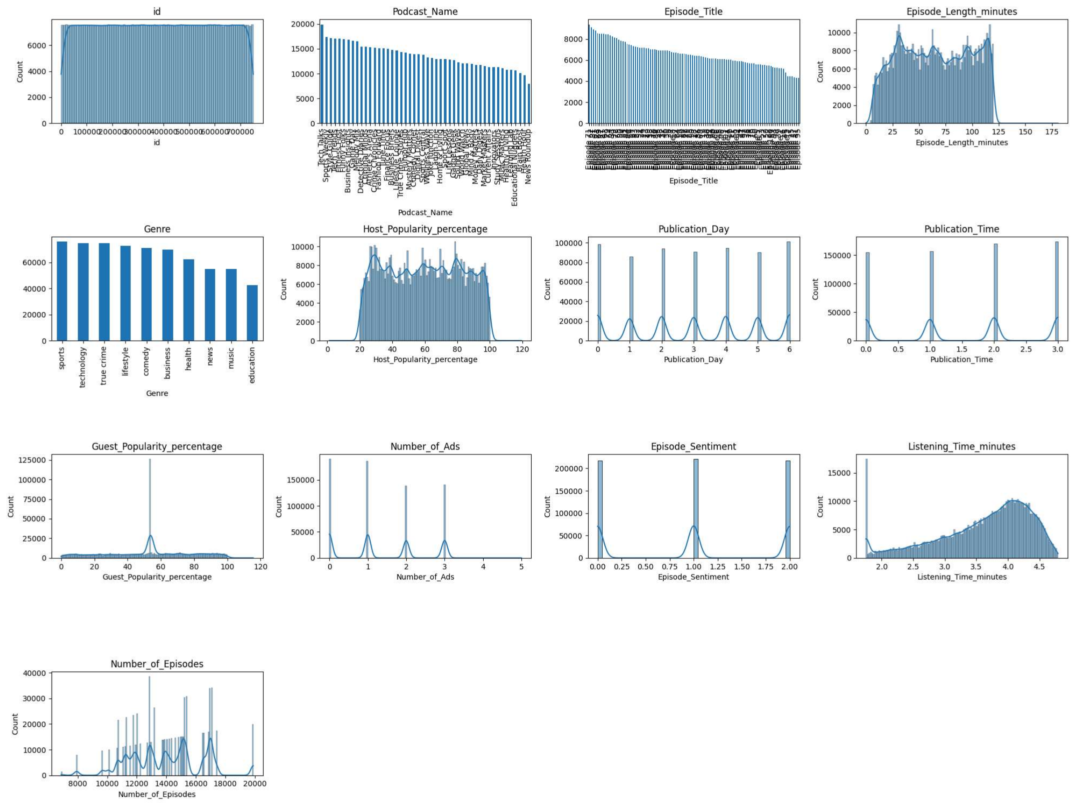
Executed plan:

- Preparing to execute plan
- Data loading
- Data cleaning
- Data exploratio
- Feature engineerin
- Data splitting
- Model training
- Model optimizati
- Model evaluation
- Data visualizati
- Finish task

I've finished executing the plan for you!

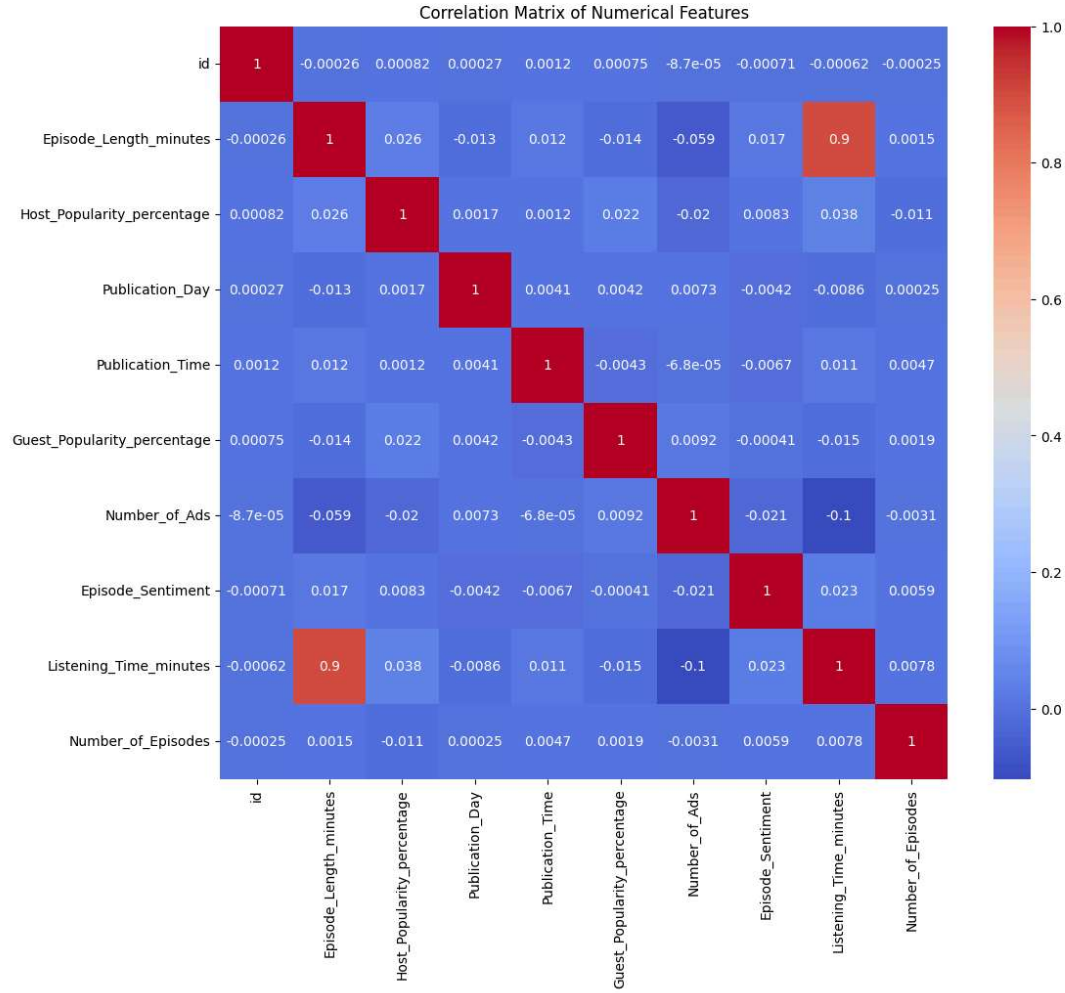
Rate this answer

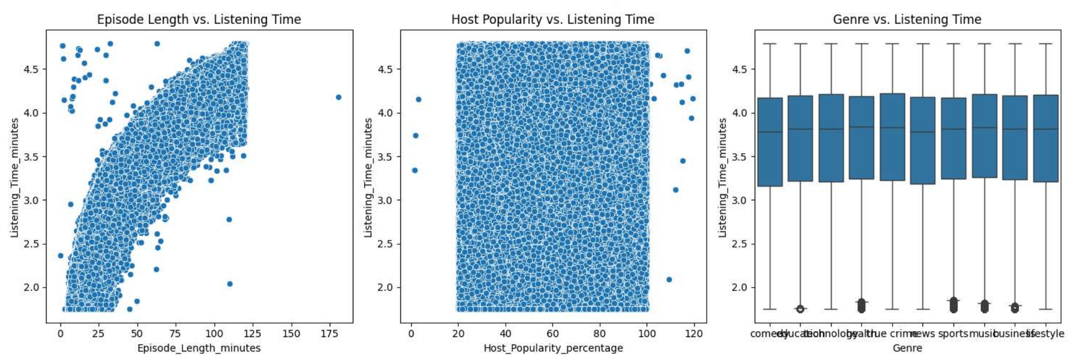
```
    print("3. Study the correlation matrix to find relationships between numerical featur
    print("4. Investigate potential relationships between features and the target variabl
```




```
                  id  Episode_Length_minutes  Host_Popularity_percentage  \
count  655066.000000           655066.000000               655066.000000
mean   375063.114001               65.185737                   59.847398
std    216534.264901               32.567467                   22.869253
min         1.000000                0.000000                    1.300000
25%    187568.250000               36.730000                   39.440000
50%    375106.500000               64.420000                   60.050000
75%    562624.750000               94.330000                   79.500000
max    749999.000000              180.730000                  119.460000

       Publication_Day  Publication_Time  Guest_Popularity_percentage  \
count    655066.000000     655066.000000                655066.000000
mean          3.027962          1.551535                    52.609309
std           2.024967          1.118241                    25.789322
min           0.000000          0.000000                     0.000000
25%           1.000000          1.000000                    34.150000
50%           3.000000          2.000000                    53.840000
75%           5.000000          3.000000                    71.440000
max           6.000000          3.000000                   115.620000

       Number_of_Ads  Episode_Sentiment  Listening_Time_minutes  \
count  655066.000000      655066.000000           655066.000000
mean        1.350322           1.000666                3.641246
std         1.111989           0.814133                0.731673
min         0.000000           0.000000                1.749622
25%         0.000000           0.000000                3.218153
50%         1.000000           1.000000                3.813270
75%         2.000000           2.000000                4.197174
max         5.000000           2.000000                4.795543

       Number_of_Episodes
count       655066.000000
mean         14035.430725
std           2437.729205
min           6872.500000
25%          11989.000000
50%          14039.000000
```

```
75%          15400.000000
max          19891.000000
```






```
Observations:
1. Analyze the distributions of numerical and categorical features to identify
2. Examine the descriptive statistics to understand the central tendency, disp
3. Study the correlation matrix to find relationships between numerical featur
4. Investigate potential relationships between features and the target variabl
```

Feature engineering

Subtask:

Engineer new features based on the insights gained from data exploration.

Reasoning: Create interaction terms, transform features, create polynomial features, and encode categorical features as instructed.

```
# Create interaction terms
df['Episode_Length_Host_Popularity'] = df['Episode_Length_minutes'] * df['Host_Popula

# Transform existing features (example: log transformation for skewed features)
# Apply log transformation to 'Number_of_Episodes'
df['Number_of_Episodes_log'] = np.log1p(df['Number_of_Episodes'])  # Using log1p to h

# Create polynomial features
df['Episode_Length_squared'] = df['Episode_Length_minutes'] ** 2

# Encode categorical features (one-hot encoding for 'Genre')
df = pd.get_dummies(df, columns=['Genre'], prefix='Genre')

display(df.head())
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-6-921bbb4fd52f> in <cell line: 0>()
      4 # Transform existing features (example: log transformation for skewed features)
      5 # Apply log transformation to 'Number_of_Episodes'
----> 6 df['Number_of_Episodes_log'] = np.log1p(df['Number_of_Episodes'])  # Using log1p to handle zero values
      7 
      8 # Create polynomial features

NameError: name 'np' is not defined
```

Next steps: Explain error

Reasoning: The previous code block failed due to `np` not being defined. I will import numpy and rerun the code.

```
import numpy as np

# Create interaction terms
df['Episode_Length_Host_Popularity'] = df['Episode_Length_minutes'] * df['Host_Populari

# Transform existing features (example: log transformation for skewed features)
# Apply log transformation to 'Number_of_Episodes'
df['Number_of_Episodes_log'] = np.log1p(df['Number_of_Episodes'])  # Using log1p to har

# Create polynomial features
```

```
df['Episode_Length_squared'] = df['Episode_Length_minutes'] ** 2

# Encode categorical features (one-hot encoding for 'Genre')
df = pd.get_dummies(df, columns=['Genre'], prefix='Genre')

display(df.head())
```

	id	Podcast_Name	Episode_Title	Episode_Length_minutes	Host_Popularity_perce
0	1	Joke Junction	Episode 26	119.80	
1	2	Study Sessions	Episode 16	73.90	
2	3	Digital Digest	Episode 45	67.17	
3	4	Mind & Body	Episode 86	110.51	
4	5	Fitness First	Episode 19	26.54	

5 rows × 25 columns

Data splitting

Subtask:

Split the data into training, validation, and testing sets.

Reasoning: Split the data into training, validation, and testing sets as instructed, using `train_test_split` from scikit-learn.

```
from sklearn.model_selection import train_test_split

# Define features (X) and target (y)
X = df.drop('Listening_Time_minutes', axis=1)
y = df['Listening_Time_minutes']

# Split data into training and temporary sets (validation + testing)
X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.3, random_state

# Split temporary set into validation and testing sets
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5, random
```

Model training

Subtask:

Train multiple regression models on the training data.

Reasoning: Train multiple regression models (Linear Regression, Random Forest, Gradient Boosting) on the training data and store them in a dictionary.

```
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor

# Initialize models
linear_model = LinearRegression()
random_forest_model = RandomForestRegressor(random_state=42)
gradient_boosting_model = GradientBoostingRegressor(random_state=42)

# Train models
linear_model.fit(X_train.drop(columns=['id', 'Podcast_Name', 'Episode_Title']), y_tra
random_forest_model.fit(X_train.drop(columns=['id', 'Podcast_Name', 'Episode_Title'])
gradient_boosting_model.fit(X_train.drop(columns=['id', 'Podcast_Name', 'Episode_Titl

# Store models in a dictionary
models = {
    'linear_regression': linear_model,
    'random_forest': random_forest_model,
    'gradient_boosting': gradient_boosting_model
}
```

Model optimization

Subtask:

Optimize the hyperparameters of the trained regression models using the validation set.

[ ] ↳ 2 cells hidden

Model evaluation

Subtask:

Evaluate the performance of the optimized models on the test set.

Reasoning: Evaluate the performance of the optimized models on the test set using MSE, R-squared, and MAE.

```
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error

# Create a dictionary to store evaluation metrics
evaluation_metrics = {}

# Iterate through the best models
for model_name, model in best_models.items():
    # Make predictions on the test set
    X_test_processed = X_test.drop(columns=['id', 'Podcast_Name', 'Episode_Title'])
```