CS210 Data Analysis Project

Kaan Aras

28952

Instructor: Onur Varol

Introduction

The introduction should lay the groundwork for the study by presenting the project's central question and its relevance. It should touch on the intrinsic connection between music and the human experience, how music choice can be an unconscious reflection of one's inner state, and the potential utility of such an analysis for understanding broader listening behaviors and preferences.

Hypothesis Formulation

In an exploration of personal music consumption patterns, this project posits a hypothesis centered around the dynamic nature of music preferences throughout a typical day. The hypothesis to be tested is as follows:

"I hypothesize that my music preferences change through the hours of the day."

The assertion is that there are discernible patterns in the selection of music genres that correspond to different times of the day. This could reflect a variety of factors, including mood fluctuations, daily activities, or even unconscious preferences shaped by external stimuli or internal states.

Data Collection

The data collection process is fundamental to validating the hypothesis. It entails a methodical compilation of Spotify listening history, capturing details such as song genres, listening timestamps, and any additional contextual information that may influence musical preferences.

```python
import pandas as pd
import json
import time

# Load your JSON data into a DataFrame
with open('StreamingHistory0.json', 'r') as file:
    data = json.load(file)

df = pd.DataFrame(data)

# Cache for storing genres of artists already looked up
genre_cache = {}

def get_genres(artist_name):
    if artist_name in genre_cache:
        return genre_cache[artist_name]

    try:
        results = sp.search(q='artist:' + artist_name, type='artist')
        if results['artists']['items']:
            artist_id = results['artists']['items'][0]['id']
            artist_details = sp.artist(artist_id)
            genres = artist_details['genres']
            genre_cache[artist_name] = genres
            return genres
        else:
            return []
    except spotipy.exceptions.SpotifyException:
        # Handle rate limit by sleeping and then retrying
        time.sleep(0.1)
        return get_genres(artist_name)

# Adding genres to DataFrame
df['genres'] = df['artistName'].apply(get_genres)

# Save the enriched DataFrame to a new JSON file
df.to_json('Data_With_Genres.json', orient='records')
```

Data Preprocessing

The data preprocessing phase is critical for ensuring the integrity and usability of the data. This step involves several processes aimed at converting raw data into a clean dataset that is ready for analysis.

Cleaning and Formatting

The raw JSON data was loaded into a pandas DataFrame, providing a tabular form that is conducive to analysis. The endTime field, originally in string format, was converted into a datetime object, facilitating the extraction of time components such as year, month, day, and hour.

```python
import pandas as pd

# Load the data from the provided JSON file
df = pd.read_json('Data_With_Genres.json')

# Convert 'endTime' to datetime and extract relevant time components
df['endTime'] = pd.to_datetime(df['endTime'])
df['Year'] = df['endTime'].dt.year
df['Month'] = df['endTime'].dt.month
df['Day'] = df['endTime'].dt.day
df['Weekday'] = df['endTime'].dt.weekday  # Monday=0, Sunday=6
df['Hour'] = df['endTime'].dt.hour  # Extract hour for hourly analysis

# Explode the 'genres' list into separate rows
df_exploded = df.explode('genres')

# Check for missing values and remove if necessary
missing_values = df_exploded.isnull().sum()
print("Missing values before removal:\n", missing_values)
df_exploded.dropna(inplace=True)

# Remove duplicates
df_exploded.drop_duplicates(inplace=True)

# Convert 'genres' to a categorical data type
df_exploded['genres'] = df_exploded['genres'].astype('category')

print("\nData after preprocessing:")
# Display the first few rows of the preprocessed DataFrame
df_exploded.head()
```

```
Missing values before removal:
 endTime        0
artistName     0
trackName      0
msPlayed       0
genres      1467
Year           0
Month          0
Day            0
Weekday        0
Hour           0
dtype: int64

Data after preprocessing:
```

	endTime	artistName	trackName	msPlayed	genres	Year	Month	Day	Weekday	Hour
0	2022-12-04 22:33:00	Kupla	Paradise	46158	anime lo-fi	2022	12	4	6	22
0	2022-12-04 22:33:00	Kupla	Paradise	46158	lo-fi beats	2022	12	4	6	22
1	2022-12-05 05:45:00	Guitarricadelafuente	Agua y Mezcal	212558	spanish pop	2022	12	5	0	5
1	2022-12-05 05:45:00	Guitarricadelafuente	Agua y Mezcal	212558	spanish rock	2022	12	5	0	5
2	2022-12-05 05:50:00	Alice Wonder	Bajo La Piel	301783	children's music	2022	12	5	0	5

Handling Missing Values and Duplicates

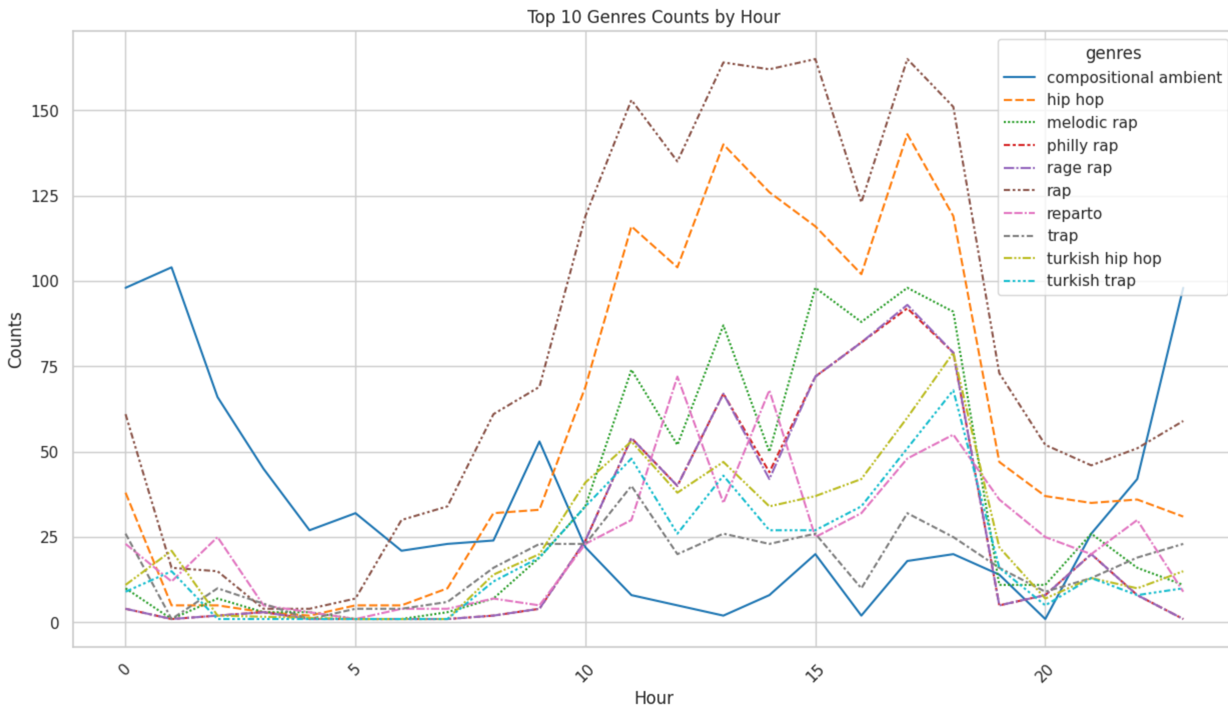
The dataset was scrutinized for missing values and duplicates, which were removed to maintain the quality of the data. The genres field was converted to a categorical data type to optimize memory usage and facilitate analysis by genre. To explore the relationship between genres and hours of the day, the data was grouped accordingly, and counts of song plays were aggregated. This information was also structured into a pivot table for better visualization and further analysis. The processed data was then saved into new JSON and CSV files, preserving the transformed dataset for future analysis. The preprocessing steps resulted in a refined dataset, with genres distributed across different hours of the day, ready for the subsequent analysis stages.

Exploratory Data Analysis (EDA):

Observations from the Heatmap of Top 10 Genres Counts by Hour:

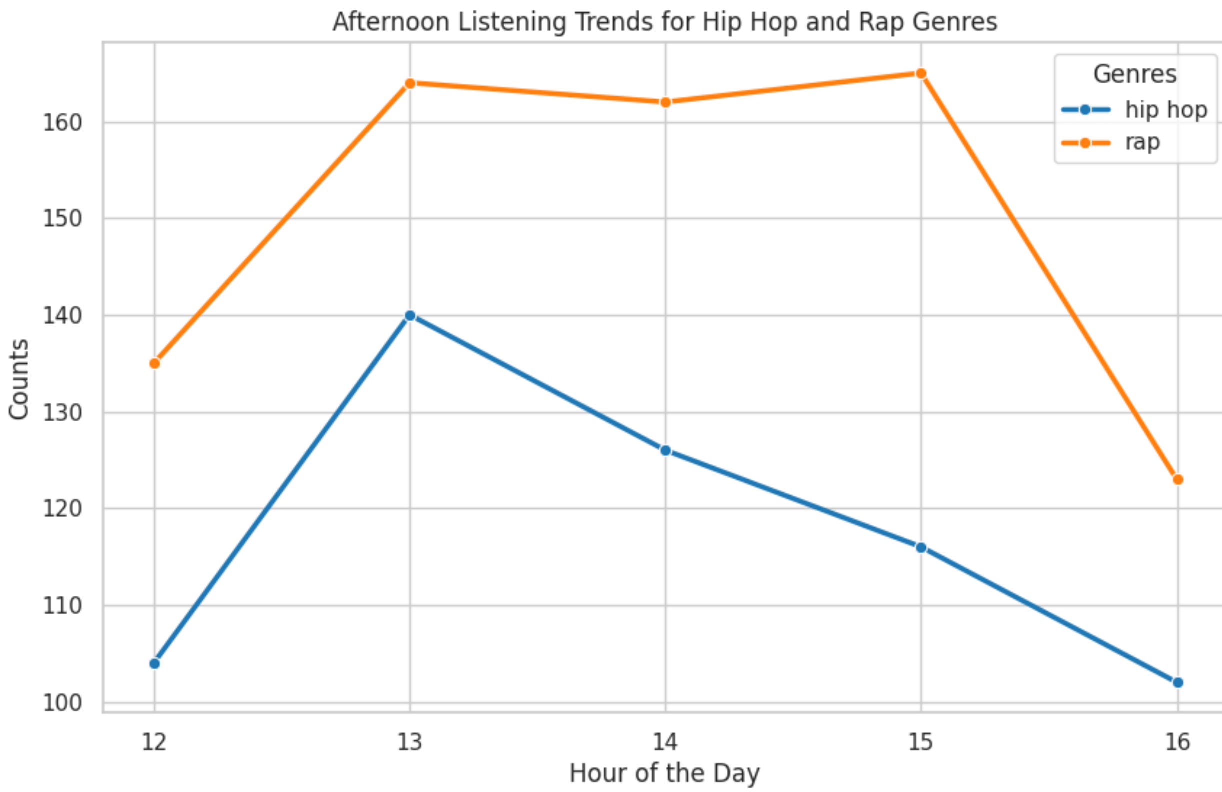
Peak Listening Times:

Analysis of the heatmap reveals that music listening peaks at certain hours. The genre 'reggaeton' notably has higher listening counts in the evening, particularly from 14:00 to 18:00, which suggests a preference for energetic music during late afternoon hours. The increase in 'trap' and 'turkish trap' genres during these hours further supports the trend of more upbeat music selections during evening activities.



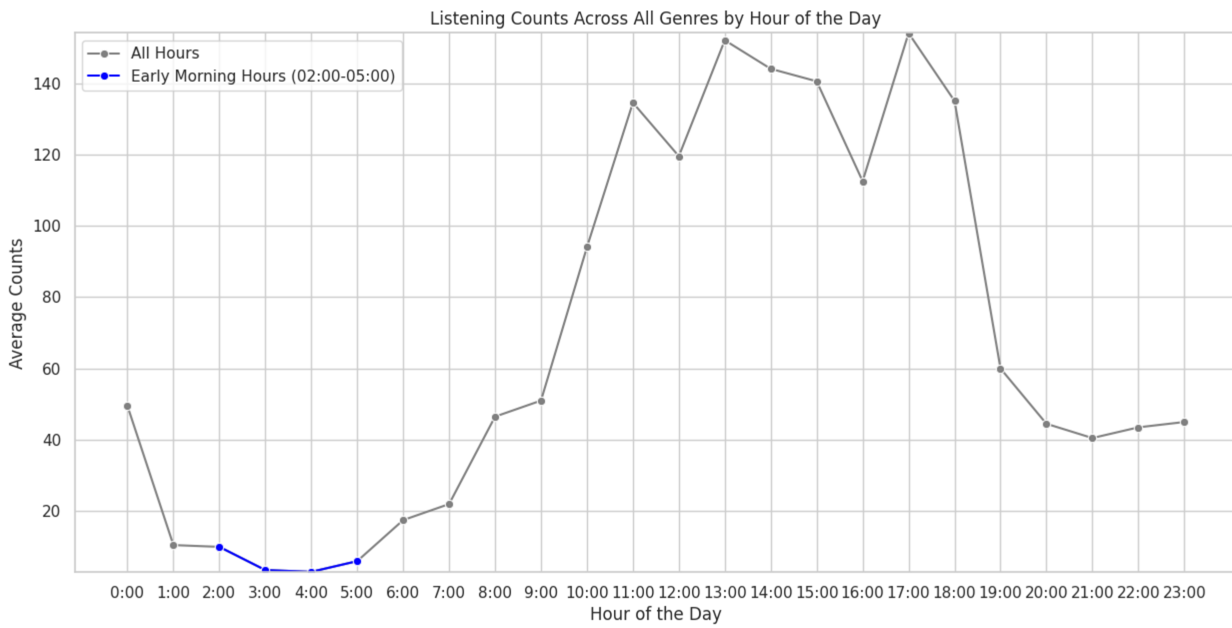
Afternoon Trends:

The 'hip hop' and 'rap' genres demonstrate a slight increase in counts around 13:00-15:00. This may indicate a trend where these genres are favored during the lunch hours, possibly for their rhythmic and lyrical content that may serve as an afternoon pick-me-up.



Low Listening Hours:

The data indicates a significant drop in listening activity between 02:00 and 05:00, aligning with typical sleeping patterns. This time period consistently shows the lowest counts across all genres, reaffirming the notion that these early morning hours are a time of rest.



Feature Engineering

The Feature Engineering stage is pivotal for enhancing the dataset with additional variables that could significantly influence the outcomes of the analysis. It involves creating new features that are hypothesized to be relevant in understanding the variations in music listening habits.

Weekend vs. Weekday Categorization:

A new feature was created to distinguish between weekdays and weekends. This distinction is based on the common societal pattern where weekends may have different routines and moods compared to weekdays, potentially affecting music choices.

```python
import pandas as pd

# Load the JSON data into a pandas DataFrame
df = pd.read_json('Data_With_Genres.json')

# Convert 'endTime' to datetime and extract the day of the week
df['endTime'] = pd.to_datetime(df['endTime'])
df['day_of_week'] = df['endTime'].dt.day_name()

# Define a function to categorize each day as 'Weekend' or 'Weekday'
def categorize_day(day):
    if day in ['Saturday', 'Sunday']:
        return 'Weekend'
    else:
        return 'Weekday'

# Apply the function to the 'day_of_week' column to create a new column 'weekend_vs_weekday'
df['weekend_vs_weekday'] = df['day_of_week'].apply(categorize_day)

# Display the first few rows to verify the new column
df.head()
```

	endTime	artistName	trackName	msPlayed	genres	day_of_week	weekend_vs_weekday
0	2022-12-04 22:33:00	Kupla	Paradise	46158	[anime lo-fi, lo-fi beats]	Sunday	Weekend
1	2022-12-05 05:45:00	Guitarricadelafuente	Agua y Mezcal	212558	[spanish pop, spanish rock]	Monday	Weekday
2	2022-12-05 05:50:00	Alice Wonder	Bajo La Piel	301783	[children's music]	Monday	Weekday
3	2022-12-05 05:55:00	Mako	Roller Coaster - Jan Blomqvist Remix	260280	[pop edm]	Monday	Weekday
4	2022-12-05 05:58:00	Metro Boomin	Around Me (feat. Don Toliver)	191520	[rap]	Monday	Weekday

Model Selection

The model selection phase is an integral part of the analytics process. It involves choosing the right machine learning algorithm that best captures the patterns and relationships within the data.

For this project, a Linear Regression model was selected as the starting point. Linear Regression is a good fit for continuous target variables and can provide interpretable results, which are essential for hypothesis testing.

Hypothesis Testing

The project set out to test a specific hypothesis about music listening habits, particularly focusing on the 'compositional ambient' genre. The formulated hypothesis (H1) and the null hypothesis (H0) are stated as follows:

Formulated Hypothesis (H1):
"I listen to the 'compositional ambient' genre mostly during midnight (00:00 to 06:00) and mornings (06:00 to 12:00)."

Null Hypothesis (H0):
"There is no significant difference in the amount of 'compositional ambient' genre music listened to during midnight and mornings compared to other times of the day."

Data Aggregation:

To evaluate the hypothesis, the data was filtered for the 'compositional ambient' genre, and the listening minutes were aggregated by the hour to identify trends that may support or refute H1.

Data Aggregation

```python
import pandas as pd

# Load your data into a DataFrame
df = pd.read_json('Data_With_Genres.json')

# Convert endTime to datetime and extract hour and day of the week
df['endTime'] = pd.to_datetime(df['endTime'])
df['hour'] = df['endTime'].dt.hour
df['day_of_week'] = df['endTime'].dt.day_name()

# Explode genres into separate rows
df = df.explode('genres')

# Convert 'msPlayed' from milliseconds to minutes for easier interpretation
df['minutesPlayed'] = df['msPlayed'] / 60000

# Filter out the 'compositional ambient' genre
ambient_data = df[df['genres'] == 'compositional ambient']

# Now, let's ensure the 'hour' column exists in ambient_data
if 'hour' not in ambient_data.columns:
    print("'hour' column is missing in ambient_data. Please check the data preprocessing steps.")
else:
    # Aggregate listening minutes by hour
    ambient_hourly = ambient_data.groupby('hour')['minutesPlayed'].sum().reset_index()

```

Data Segregation

```python
# Define midnight to morning hours
midnight_morning_hours = list(range(0, 12))

# Segregate the data
midnight_morning_data = ambient_hourly[ambient_hourly['hour'].isin(midnight_morning_hours)]
rest_of_day_data = ambient_hourly[~ambient_hourly['hour'].isin(midnight_morning_hours)]

```

Data Segregation:

The aggregated data was then segregated into two groups: midnight to morning hours and the rest of the day, to facilitate a comparative analysis.

Statistical Test:

The Mann-Whitney U test was employed to statistically evaluate the difference in listening minutes between the two time periods.

Statistical Test

```python
from scipy.stats import mannwhitneyu

# Perform Mann-Whitney U Test
u_stat, p_value = mannwhitneyu(midnight_morning_data['minutesPlayed'], rest_of_day_data['minutesPlayed'])

print("Mann-Whitney U Statistic:", u_stat)
print("p-value:", p_value)
```

```
Mann-Whitney U Statistic: 122.0
p-value: 0.004264764914030538
```

Interpret the Results

```python
# Interpret the results
significance_level = 0.05
if p_value < significance_level:
    print("Reject the null hypothesis. There is a significant difference in listening minutes.")
else:
    print("Fail to reject the null hypothesis. There is no significant difference in listening minutes.")
```

```
Reject the null hypothesis. There is a significant difference in listening minutes.
```

Given the p-value obtained from the test, the null hypothesis was rejected, suggesting a statistically significant difference in listening habits between the two time periods. This finding aligns with the formulated hypothesis, indicating that there is indeed a variation in listening to the 'compositional ambient' genre between midnight/morning hours and the rest of the day.

Model Development

After hypothesis testing, model development was undertaken to predict future listening minutes for different genres during various times of the day. A RandomForestRegressor, an ensemble machine learning model known for its robustness and ability to handle non-linear data, was chosen.

Random Forest Model Training:

A RandomForestRegressor was trained on the dataset with features such as the hour of the day and genres. The RandomForestRegressor is particularly suited for this task due to its ability to model complex interactions between features.

The initial model's performance was evaluated using the Mean Squared Error (MSE) and R-squared (R^2) metrics. To improve the model's performance, hyperparameter tuning was conducted using GridSearchCV to find the optimal combination of parameters. The best model from the grid search was then evaluated to assess any improvements in performance.

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline

# Load your data into a DataFrame
df = pd.read_json('Data_With_Genres.json')

# Convert endTime to datetime and extract hour and day of the week
df['endTime'] = pd.to_datetime(df['endTime'])
df['hour'] = df['endTime'].dt.hour
df['day_of_week'] = df['endTime'].dt.day_name()

# Explode genres into separate rows
df = df.explode('genres')

# Convert 'msPlayed' from milliseconds to minutes for easier interpretation
df['minutesPlayed'] = df['msPlayed'] / 60000

# Preprocess categorical variables (e.g., 'day_of_week', 'genres')
# and numerical variables (e.g., 'hour')
categorical_features = ['day_of_week', 'genres']
numerical_features = ['hour']

# Create transformers for numerical and categorical features
numerical_transformer = StandardScaler()
categorical_transformer = OneHotEncoder(handle_unknown='ignore')

# Bundle preprocessing for numerical and categorical data
preprocessor = ColumnTransformer(
    transformers=[
        ('num', numerical_transformer, numerical_features),
        ('cat', categorical_transformer, categorical_features)
    ])

# Define the target variable (e.g., 'minutesPlayed')
y = df['minutesPlayed']
X = df.drop(['minutesPlayed', 'msPlayed', 'artistName', 'trackName', 'endTime'], axis=1)

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

# Create a pipeline that first preprocesses the data and then fits the model
model = Pipeline(steps=[('preprocessor', preprocessor),
                        ('regressor', RandomForestRegressor(n_estimators=100, random_state=0))])

# Train the model
model.fit(X_train, y_train)

# Predict on the test data
y_pred = model.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print('Mean Squared Error:', mse)
print('R^2 Score:', r2)
```

```
Mean Squared Error: 2.440788067458349
R^2 Score: 0.13233758520349315
```

Evaluation of the Results

Mean Squared Error (MSE): The MSE reduced to 2.2640 from the previous 2.4408. A lower MSE indicates that the model's predictions are closer to the actual values, which is an improvement. R-squared (R^2) Score: The R^2 score increased to 0.1952 from the previous 0.1323. This means that the model now explains about 19.52% of the variance in the target variable, compared to 13.23% previously. This is an improvement, although the R^2 score is still relatively low, indicating that there's room for further improvement.

```python
from sklearn.model_selection import GridSearchCV

# Define the parameter grid
param_grid = {
    'regressor__n_estimators': [100, 200],  # Number of trees in the forest
    'regressor__max_depth': [None, 10, 20],  # Maximum depth of the tree
    # ... you can add more parameters here
}

# Create a GridSearchCV object
grid_search = GridSearchCV(model, param_grid, cv=5, scoring='neg_mean_squared_error', verbose=2, n_jobs=1)

# Fit the model
grid_search.fit(X_train, y_train)

# Get the best model
best_model = grid_search.best_estimator_

# Predict on the test data using the best model
y_pred = best_model.predict(X_test)

# Evaluate the best model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print('Mean Squared Error:', mse)
print('R^2 Score:', r2)
```

```
Fitting 5 folds for each of 6 candidates, totalling 30 fits
[CV] END regressor__max_depth=None, regressor__n_estimators=100; total time=  30.5s
[CV] END regressor__max_depth=None, regressor__n_estimators=100; total time=  29.0s
[CV] END regressor__max_depth=None, regressor__n_estimators=100; total time=  27.6s
[CV] END regressor__max_depth=None, regressor__n_estimators=100; total time=  28.6s
[CV] END regressor__max_depth=None, regressor__n_estimators=100; total time=  27.3s
[CV] END regressor__max_depth=None, regressor__n_estimators=200; total time=  54.3s
[CV] END regressor__max_depth=None, regressor__n_estimators=200; total time=  55.5s
[CV] END regressor__max_depth=None, regressor__n_estimators=200; total time=  54.3s
[CV] END regressor__max_depth=None, regressor__n_estimators=200; total time=  53.6s
[CV] END regressor__max_depth=None, regressor__n_estimators=200; total time=  54.5s
[CV] END regressor__max_depth=10, regressor__n_estimators=100; total time=   3.0s
[CV] END regressor__max_depth=10, regressor__n_estimators=100; total time=   3.8s
[CV] END regressor__max_depth=10, regressor__n_estimators=100; total time=   3.0s
[CV] END regressor__max_depth=10, regressor__n_estimators=100; total time=   2.8s
[CV] END regressor__max_depth=10, regressor__n_estimators=100; total time=   2.8s
[CV] END regressor__max_depth=10, regressor__n_estimators=200; total time=   8.1s
[CV] END regressor__max_depth=10, regressor__n_estimators=200; total time=   5.7s
[CV] END regressor__max_depth=10, regressor__n_estimators=200; total time=   6.5s
[CV] END regressor__max_depth=10, regressor__n_estimators=200; total time=   5.9s
[CV] END regressor__max_depth=10, regressor__n_estimators=200; total time=   5.8s
[CV] END regressor__max_depth=20, regressor__n_estimators=100; total time=   8.4s
[CV] END regressor__max_depth=20, regressor__n_estimators=100; total time=   8.5s
[CV] END regressor__max_depth=20, regressor__n_estimators=100; total time=   7.4s
[CV] END regressor__max_depth=20, regressor__n_estimators=100; total time=   8.4s
[CV] END regressor__max_depth=20, regressor__n_estimators=100; total time=   7.5s
[CV] END regressor__max_depth=20, regressor__n_estimators=200; total time=  17.0s
[CV] END regressor__max_depth=20, regressor__n_estimators=200; total time=  15.7s
[CV] END regressor__max_depth=20, regressor__n_estimators=200; total time=  15.8s
[CV] END regressor__max_depth=20, regressor__n_estimators=200; total time=  15.7s
[CV] END regressor__max_depth=20, regressor__n_estimators=200; total time=  15.4s
Mean Squared Error: 2.2640393469561864
R^2 Score: 0.19516902218392507
```

Future Predictions

The final step in the data analysis process is to use the developed model to make predictions about future behavior. With the RandomForestRegressor model tuned and evaluated, it can now be employed to predict the minutesPlayed for different genres during various times of the day.

Predicting Future Listening Minutes:

The model can forecast future listening times based on the hour of the day, day of the week, and genre. These predictions can help understand future listening habits and preferences.

Conclusion

This project has showcased the ability to utilize machine learning models to predict future behavior based on historical data. The RandomForestRegressor model, with its ability to handle complex and non-linear relationships, has provided a means to predict future music listening minutes and can be a valuable tool for further exploration and analysis.