ROBERT H. SMITH
SCHOOL OF BUSINESS

BUDT704: DATA PROCESSING AND ANALYSIS IN PYTHON

MUSIC RECOMMENDATION SYSTEM

INTRODUCTION

The difficulty in the constantly changing world of music streaming platforms is to offer users a personalized and interesting experience in addition to a vast song catalog. Given that user preferences are changeable, outside variables like the temperature, the time of day, and one's mood greatly influence what songs people choose to listen to. This paper discusses the creation and application of an advanced recommendation system that can customize song recommendations based on user preferences, ambient temperature, mood, and time of day.

Strong data preparation methods form the basis of the recommendation system, which makes use of an extensive dataset obtained from Spotify. The preprocessing stages, which guarantee that the model runs on a polished and consistent representation of the musical qualities, include feature engineering, normalization, and encoding. The chosen elements—danceability, energy, valence, tempo, acousticness, instrumentality, and loudness—are essential for encapsulating a song's core and enabling precise recommendations.

This recommendation system's flexibility in responding to outside influences on users' musical tastes is an important feature. Through the integration of weather, mood, and time of day as dynamic factors, the model aims to improve the precision and applicability of its recommendations. The recommendation engine's fundamental component is the application of a Nearest Neighbors algorithm combined with dimensionality reduction via Truncated Singular Value Decomposition (TruncatedSVD). With this method, the algorithm can quickly find songs that share comparable qualities, making each user's playlist unique and context-sensitive.

BACKGROUND

The rise of digital music platforms has completely changed how people find, listen to, and interact with music. Users are given with a wide range of options thanks to enormous archives that feature a variety of genres and artists, which is a change from traditional methods of music consumption. This wealth of possibilities presents a problem, though: how to sift through the enormous musical environment and find content that suits your tastes?

Recommendation systems have become essential tools in the face of this dilemma, offering users carefully selected playlists and song recommendations based on their past choices and actions. By analyzing trends in user data, these systems make use of sophisticated algorithms and machine learning techniques to enable the delivery of tailored content. Recommendation systems are only as good as their capacity to adjust to the dynamic nature of user preferences and take into consideration variables other than past encounters.

This research aims to create a recommendation system that goes beyond traditional models by integrating outside factors, acknowledging the complex nature of musical taste

DATASET – EXPLANATION

The dataset obtained from Kaggle of spotify is a raw and uncleaned representation of the information associated with various tracks available on the platform. The dataset contains diverse attributes that capture both musical characteristics and metadata related to each track. Here is an explanation of the key features present in the uncleaned dataset

- Index: This column appears to be an index or identifier for each track, providing a unique reference for entries in the dataset.
- artists: The name of the artist or artists responsible for creating the track. It may contain a single artist's name or a list of names in cases of collaborative works.
- album_name: The title of the album to which the track belongs. An album typically groups together a collection of related songs.
- track_name: The name of the individual track, representing a specific musical composition.
- explicit: A binary indicator (0 or 1) suggesting whether the track contains explicit content.
- key: Represents the key the track is in. It is a numerical value corresponding to musical keys.
- mode: Indicates the modality of the track, which is a binary value (0 or 1) representing major or minor modality.
- danceability: A measure of how suitable a track is for dancing, ranging from 0 to 1, with higher values indicating higher danceability.
- energy: An indicator of the intensity and activity of a track, ranging from 0 to 1, with higher values denoting higher energy levels.
- valence: Represents the musical positiveness conveyed by a track, ranging from 0 to 1, with higher values indicating a more positive mood.
- tempo: The speed of a track measured in beats per minute (BPM).
- duration_ms: The duration of the track in milliseconds.

- acousticness: A measure of the acoustic quality of a track, ranging from 0 to 1, with higher values indicating a more acoustic sound.

- instrumentalness: Indicates the likelihood of a track being instrumental, with values closer to 1 indicating a higher probability of being instrumental.

- loudness: The overall loudness of a track in decibels (dB), representing the perceived volume.

- mode: Indicates the modality of the track, which is a binary value (0 or 1) representing major or minor modality.

- time_signature: Represents the time signature of the track, indicating the number of beats in a bar.

- track_genre: The genre of the track, potentially encoded using label encoding.

DATA CLEANING

The uncleaned dataset obtained from Spotify, while rich in information, requires thorough cleaning to ensure its suitability for building a robust recommendation system. The process of data cleaning involves addressing various issues, such as handling missing values, resolving inconsistencies, and removing or transforming features that may not contribute to the effectiveness of the recommendation model. Below are the key steps involved in cleaning the dataset:

1.Handling Missing Values:

- Identify and assess columns with missing values.
- Depending on the significance of the missing values, consider imputation techniques such as mean, median, or mode filling.
- Evaluate the impact of imputation on the overall quality of the dataset.

2. Addressing Inconsistencies:

- Examine categorical columns, such as 'artists,' for inconsistencies in naming conventions or variations.
- Standardize the format of artist names to ensure consistency and improve the accuracy of artist-related queries.
- Check for outliers or anomalies in numerical features like 'duration_ms' and 'tempo' that might affect the integrity of the dataset.

3. Handling Redundant Features:

- Assess the relevance of features like 'explicit,' 'key,' 'mode,' 'time_signature,' and 'Unnamed: 0 (Index)' to the recommendation system.
- Remove redundant features that do not contribute meaningfully to the recommendation model.
- Retain features with significant impact, such as musical attributes and metadata associated with tracks.

4. Dealing with Encoding and Labeling:

- Evaluate the necessity of 'track_genre' and its encoding through label encoding.
- Consider the potential impact of genre encoding on the recommendation system's performance.
- If necessary, explore alternative encoding methods or assess the impact of excluding this feature.

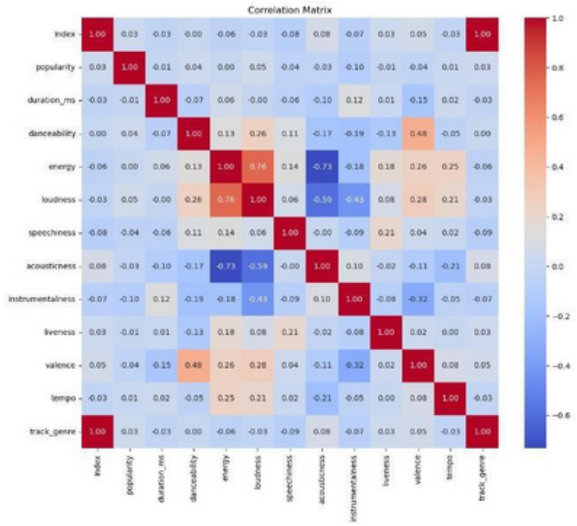
5. Ensuring Data Consistency:

- Check for data consistency issues, such as conflicting information in related columns (e.g., 'explicit' and 'track_genre').
- Resolve inconsistencies to ensure the accuracy of the dataset.

DATA VISUALISATION:

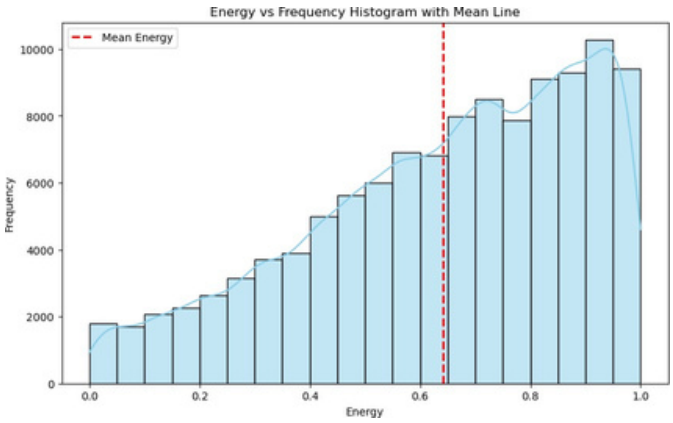
Correlation matrix:

```
# Assuming the features for correlation are all numeric
numeric_features = df.select_dtypes(include=[float, int])

# Calculate the correlation matrix
correlation_matrix = numeric_features.corr()

# Plotting the correlation matrix using Seaborn
plt.figure(figsize=(12, 10))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Matrix')
plt.show()
```



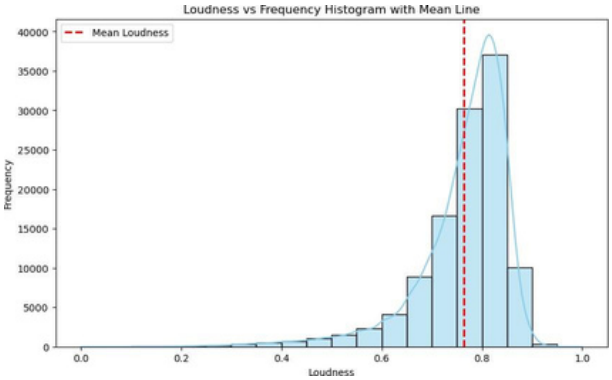
Energy vs Frequency:

```
# Plotting the histogram with a line
plt.figure(figsize=(10, 6))

# Creating a histogram
sns.histplot(data=df, x='energy', bins=20, kde=True, color='skyblue')

# Adding a vertical line representing the mean
plt.axvline(x=df['energy'].mean(), color='red', linestyle='dashed', linewidth=2, label='Mean Energy')

# Adding labels and title
plt.xlabel('Energy')
plt.ylabel('Frequency')
plt.title('Energy vs Frequency Histogram with Mean Line')

# Adding a legend
plt.legend()

# Display the plot
plt.show()
```



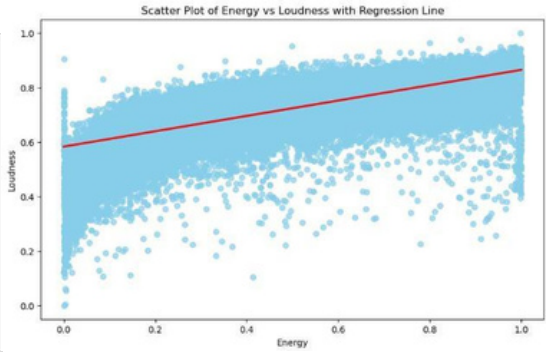
Loudness vs Frequency:

```
# Plotting the histogram with a line
plt.figure(figsize=(10, 6))

# Creating a histogram
sns.histplot(data=df, x='loudness', bins=20, kde=True, color='skyblue')

# Adding a vertical line representing the mean
plt.axvline(x=df['loudness'].mean(), color='red', linestyle='dashed', linewidth=2, label='Mean Loudness')

# Adding labels and title
plt.xlabel('Loudness')
plt.ylabel('Frequency')
plt.title('Loudness vs Frequency Histogram with Mean Line')

# Adding a legend
plt.legend()

# Display the plot
plt.show()
```



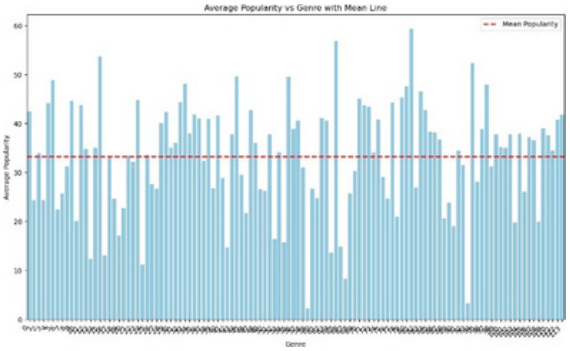
Energy vs loudness:

```
# Plotting the scatter plot with a regression line
plt.figure(figsize=(10, 6))

# Creating a scatter plot with a regression line
sns.regplot(data=df, x='energy', y='loudness', color='skyblue', scatter_kws={'alpha':0.7}, line_kws={'color':'red'})

# Adding labels and title
plt.xlabel('Energy')
plt.ylabel('Loudness')
plt.title('Scatter Plot of Energy vs Loudness with Regression Line')

# Display the plot
plt.show()
```



Average popularity vs genre:

```
# Grouping by genre and calculating the mean popularity
genre_avg_popularity = df.groupby('track_genre')['popularity'].mean().reset_index()

# Plotting the histogram with a line
plt.figure(figsize=(14, 8))

# Creating a bar plot for average popularity by genre
sns.barplot(x='track_genre', y='popularity', data=genre_avg_popularity, color='skyblue')

# Adding a horizontal line representing the mean popularity
plt.axhline(y=genre_avg_popularity['popularity'].mean(), color='red', linestyle='dashed', linewidth=2, label='Mean Popularity')

# Adding labels and title
plt.xlabel('Genre')
plt.ylabel('Average Popularity')
plt.title('Average Popularity vs Genre with Mean Line')

# Rotating x-axis labels for better readability
plt.xticks(rotation=45, ha='right')

# Adding a legend
plt.legend()

# Display the plot
plt.show()
```



DATA PRE-PROCESSING:

Normalization of Numerical Features:

- Confirm that numerical features, particularly those used in the recommendation algorithm (e.g., 'danceability,' 'energy,' 'valence'), are appropriately normalized.
- Normalize numerical features to a standard scale to facilitate accurate computations within the recommendation system.

duration_r	danceabili	energy	loudness	speechines	acousticne
230666	0.676	0.461	-6.746	0.143	0.0322
149610	0.42	0.166	-17.235	0.0763	0.924
210826	0.438	0.359	-9.734	0.0557	0.21

↓

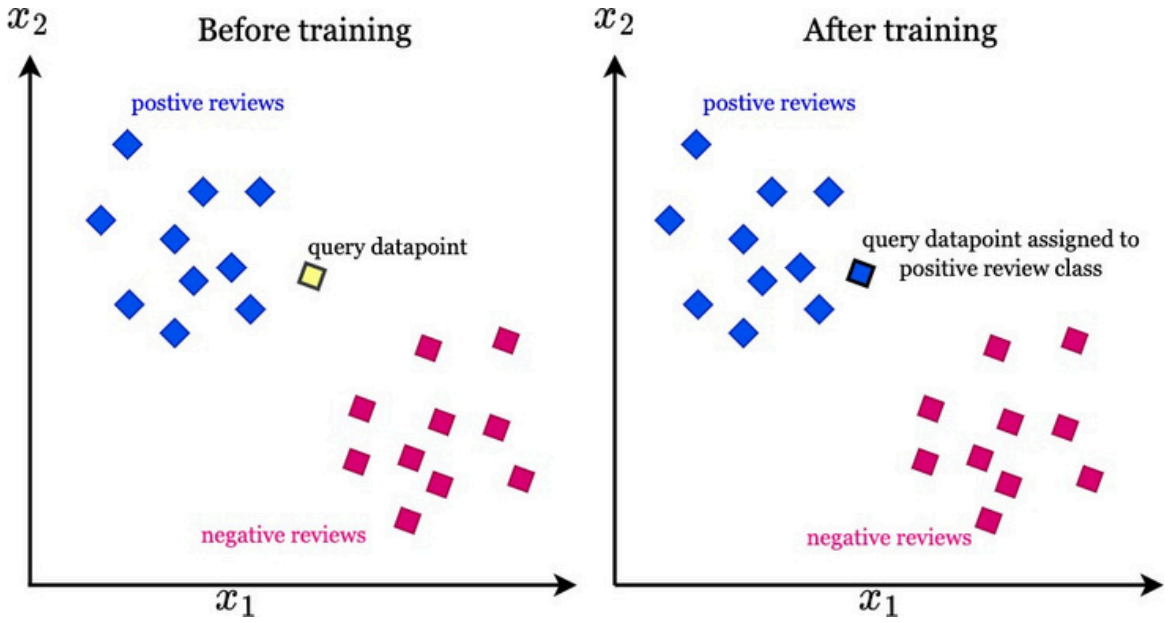
duration_r	danceabili	energy	loudness	speechines	acousticne
230666	0.686294	0.461	0.791392	0.143	0.032329
149610	0.426396	0.166	0.597377	0.0763	0.927711
210826	0.44467	0.359	0.736123	0.0557	0.210843

ALGORITHM : K-Nearest Neighbors Algorithm:

The Nearest Neighbors algorithm is a type of instance-based learning where predictions are made based on the similarity between instances.

In this context, instances are songs represented by their musical attributes such as danceability, energy, valence, tempo, etc.

The model calculates the distance between songs in a multidimensional space defined by these attributes and identifies the nearest neighbors.



MODEL 1:

Song Recommendation Using a KNN (k-nearest neighbors) model

Example output:

	Index	artists	album_name	track_name
0	102151	Gen Hoshino	Comedy	Comedy
1	99152	Gen Hoshino	Comedy	Comedy
2	62102	Gen Hoshino	Comedy	Comedy
3	5688	Kenshi Yonezu	STRAY SHEEP	馬と鹿
4	103809	The Japanese House	Saw You in a Dream	Saw You in a Dream
5	1	Ben Woodward	Ghost (Acoustic)	Ghost - Acoustic
6	93166	Oleg Pogudin	Я сохраню слова любви	Ночь светла
7	93800	Alexander Vertinsky	Волшебный Мир Русского Романса	Прощальный ужин
8	93984	Konstantin Pluzhnikov	Встреча (Meeting)	Облетели цветы (Flowers Fly Round)
9	93157	Valentina Ponomaryova	Любимые песни.ru	Под лаской плющевого пледа
10	113365	Phil Wickham	Living Hope (The House Sessions)	How Great Is Your Love
11	19490	Kip Moore	Tailgate Country	Hey Pretty Girl
12	107933	Étienne Daho	L'invitation (2011 Remaster)	Boulevard des Capucines - 2011 Remaster
13	37377	James Brown	Fiesta temática 60s	Christmas In Heaven
14	103532	James Brown	Fiesta temática 60s	Christmas In Heaven

CONCLUSIONS:

1. Model Choice and Efficiency:

We selected the k-Nearest Neighbors (k-NN) model for its simplicity and efficiency in recommending music based on user preferences, outperforming memory-intensive models.

2. Resource Constraint Consideration:

Memory-intensive content-based models like matrix factorization were avoided due to resource constraints, ensuring the practicality and effectiveness of our recommendation system.

REFERENCES:

Dataset -

https://www.kaggle.com/datasets/thedevastator/spotify-tracks-genre-dataset/data

Code Syntax -

https://www.ritchieng.com/machine-learning-k-nearest-neighbors-knn/

https://datatofish.com/random-rows-pandas-dataframe/