Proceedings of the 4th International Conference on Signal Processing and Machine Learning
DOI: 10.54254/2755-2721/52/20241657

Investigating the working principle of the recommending system and designing a personal music recommending system

Arvin Ha

Beijing No.4 High School International Campus, China

Arvinha1220@gmail.com

Abstract. This paper, "Investigating the Working Principle of the Recommending System and Designing a Personal Music Recommending System," delves into the intricacies of recommendation systems, with a specific focus on music. It begins by exploring the fundamental principles that underpin recommendation systems.The latter part of the paper presents the design of a novel personal music recommendation system. This system leverages the results of K-means clustering to provide faster personalized music recommendations. It considers various auditory factors such as valence (the musical positiveness conveyed by a track), duration (the length of the track), and energy (a measure from 0.0 to 1.0 and represents a perceptual measure of intensity and activity). These factors are calculated by Spotify and made available in an open data set. This research contributes significantly to the field by providing a deeper understanding of recommendation systems and proposing an innovative solution for personalized music recommendations. The novelty of this system lies in its use of K-means clustering and the specific auditory factors it considers, which sets it apart from existing systems.

Keywords: Data Science, Computer Science, Statistics, Clustering, Recommending System

1. Introduction

According to a 2022 survey from the International Federation of the Phonographic Industry (IFPI), 69% of the world's population stated that listening to music is important for their mental health. The top five ways that listeners engage with music are through video streaming (82%), audio streaming (74%), radio (71%), short-form video (68%), and social media (49%). Most of these methods require software on mobile devices.

Furthermore, musical preferences vary greatly among individuals. Some may enjoy pop music, while others prefer rock music. Therefore, a music recommendation system can be incredibly helpful for music lovers looking to discover new songs that align with their tastes. A music recommendation system provides personalized song recommendations based on the user's listening history [1].

However, most popular online music platforms have recommendation systems that prioritize promoting the latest or most well-known songs, which often require a membership to listen. This approach may not always align with the user's preferences and can limit their discovery of new music[2].

Therefore, this paper proposes the development of a non-commercial music recommendation system that focuses solely on the user's preferences. This system aims to enhance the user's music discovery experience by prioritizing their tastes over commercial interests. The working principle of the

DOI: 10.54254/2755-2721/52/20241657

recommendation system and the design of a personalized music recommendation system based on clustering will be discussed in detail in the following sections [3].

2. Literature Review

2.1. Sub-section 1: Clustering

The application of the K-Means Clustering algorithm for predicting students' academic performance is explored in a paper by Oyeladee, O. J, Oladipupo, O. O, Obagbuwa, I. C (2010). This paper provides a practical example of using Python for K-means clustering, which is directly relevant to our research as we employ a similar method for our music recommendation system [4].

A survey by Amineh Amini, Teh Ying Wah & Hadi Saboohi (2014) provides a comprehensive overview of density-based clustering algorithms for data streams. This survey is particularly useful for our research as it discusses the unique features and limitations of these algorithms, providing a foundation for our understanding of data stream clustering [5].

The book "Mining of Massive Datasets" by Jure Leskovec, Anand Rajaraman, Jeff Ullman provides a basic understanding of distance calculation and clustering, particularly K-means clustering. It introduces the elbow method for determining the number of clusters, which we have adopted in our research[6].

2.2. Sub-section 2: Recommending System

A paper by Jie Lu, Dianshuang Wu, Mingsong Mao, Wei Wang, Guangquan Zhang (2015) reviews the application developments of recommendation systems. It categorizes the applications into eight main categories and summarizes the related recommendation techniques used in each category. This paper is crucial for our research as it provides a different perspective on recommendation systems, focusing on the application rather than the mathematical or coding approach [7].

Chapter 9 of the book "Mining of Massive Datasets" by Jure Leskovec, Anand Rajaraman, Jeff Ullman provides a clear explanation of the working principles and applications of recommendation systems. This chapter forms the knowledge basis for our essay and introduces the Euclidean distance method, which we use in our recommendation system [6].

A paper by Juanjuan Shi (2021) provides a real-life example of using data mining and machine learning algorithms to build a complex music recommendation system in multiple time dimensions. This paper is particularly relevant to our research as it offers a comparative goal and helps in the process of building our recommendation system [8].

The paper "Enhanced Books Recommendation Using Clustering Techniques and Knowledge Graphs" discusses the use of clustering techniques in recommendation systems. This paper is particularly relevant as it demonstrates the application of clustering techniques, including the use of Euclidean distance, in a recommendation system [9].

The paper "Functions of units, scales and quantitative data: Fundamental differences in numerical traceability between sciences" discusses the importance of standardizing numerical values in data analysis. This paper is particularly relevant as it underscores the necessity of standardizing numerical values in our dataset to ensure all parameters contribute equally to the clustering process [10].

In their 2023 paper, Li et al. address the challenge of microtext in chatbots, a prevalent form of communication on social media. They incorporate microtext normalization into a chatbot, improving its understanding and performance. This work is relevant to our research as it highlights the importance of adapting to evolving language use in digital platforms, crucial for developing effective music recommendation systems [11].

Xu and Tian(2022) offer a comprehensive survey of clustering algorithms. They analyze traditional and modern algorithms, emphasizing the importance of feature extraction, selection, and result evaluation. This work supports our choice of the K-means clustering algorithm for our music recommendation system [12].

DOI: 10.54254/2755-2721/52/20241657

3. Methodology

The dataset used for this research is an open dataset published by Spotify, a digital music, podcast, and video service. This dataset contains songs from 1921 to 2020, along with their musical parameters. These parameters, which include valence(a measure of musical positiveness), acousticness(a measure of the presence of acoustic sound.), danceability(describes how suitable a track is for dancing based on a combination of musical elements including tempo, rhythm stability, beat strength, and overall regularity), duration(the length of the song or composition), energy(a measure of intensity and activity), instrumentalness(Predicts whether a track contains vocals. instrumentalness is measured on a scale of 0 to 100), liveness(detects the presence of an audience in the recording), loudness(the attribute of a sound that determines the magnitude of the auditory sensation produced and that primarily depends on the amplitude of the sound wave involved), speechiness(measures the presence of spoken words in a song), tempo(the speed or pace of a given composition), and popularity(a measure of the song's overall popularity with listeners), provide a comprehensive characterization of each song. These parameters were chosen because they capture various aspects of a song that can influence a listener's preference.

After downloading the dataset as a CSV file, the data was analyzed using Stata and the Python library, Pandas. Stata was used to calculate summary statistics such as the mean, max, and min values of each parameter. This helped to understand the distribution of values for each parameter. Pandas is a powerful Python library that provides data structures and functions needed for data manipulation and analysis. In this project, it is used to DataFrame object in a two-dimensional size-mutable, potentially heterogeneous tabular data structure with labeled axes, achieve data visualization easily and read the CSV file. (Fig.1)

The dataset contains 170,653 data points and 19 columns, with 9 columns in float form, 6 in integer form, and 4 in string form.(Fig.2) Given the significant differences in the range of values for the numerical parameters, standardization was necessary to ensure that all parameters contribute equally to the clustering process. For this, the StandardScaler function from the sklearn library in Python was used. This function standardizes features by removing the mean and scaling to unit variance [10].

This methodology ensures a thorough analysis of the dataset and prepares it for the subsequent clustering process. The use of K-means clustering, as informed by the literature, will be discussed in the following sections.

```
. import delimited "C:\Users\18410\Desktop\Data science\Spotify\data.csv"
(19 vars, 170,653 obs)

. su
```

Variable	Obs	Mean	Std. Dev.	Min	Max
valence	170,653	.5285872	.2631715	0	1
year	170,653	1976.787	25.91785	1921	2020
acousticness	170,653	.5021148	.3760317	0	.996
artists	0				
danceability	170,653	.5373955	.1761377	0	.988
duration_ms	170,653	230948.3	126118.4	5108	5403500
energy	170,653	.4823888	.2676457	0	1
explicit	170,653	.0845751	.2782492	0	1
id	0				
instrument~s	170,653	.1670096	.3134747	0	1
key	170,653	5.199844	3.515094	0	11
liveness	170,653	.2058387	.1748047	0	1
loudness	170,653	-11.46799	5.697943	-60	3.855
mode	170,653	.7069023	.4551842	0	1
name	0				
popularity	170,653	31.43179	21.82662	0	100
release_date	0				
speechiness	170,653	.0983933	.1627401	0	.97
tempo	170,653	116.8616	30.70853	0	243.507

```
.
```

Figure 1. Result of using Stata to analyze the data

DOI: 10.54254/2755-2721/52/20241657

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 170653 entries, 0 to 170652
Data columns (total 19 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   valence           170653 non-null  float64
 1   year              170653 non-null  int64
 2   acousticness      170653 non-null  float64
 3   artists           170653 non-null  object
 4   danceability      170653 non-null  float64
 5   duration_ms       170653 non-null  int64
 6   energy            170653 non-null  float64
 7   explicit          170653 non-null  int64
 8   id                170653 non-null  object
 9   instrumentalness  170653 non-null  float64
 10  key               170653 non-null  int64
 11  liveness          170653 non-null  float64
 12  loudness          170653 non-null  float64
 13  mode              170653 non-null  int64
 14  name              170653 non-null  object
 15  popularity        170653 non-null  int64
 16  release_date      170653 non-null  object
 17  speechiness       170653 non-null  float64
 18  tempo             170653 non-null  float64
dtypes: float64(9), int64(6), object(4)
memory usage: 24.7+ MB
None
```

Figure 2. The result of using Panda to analyze the data

4. Result

The Elbow Method is a widely used technique in determining the optimal number of clusters in a data set. The method involves plotting the explained variation (or the sum of squared errors, SSE) as a function of the number of clusters. The “elbow” of the curve, where the rate of decrease sharply shifts, indicates the optimal number of clusters. This is based on the idea that increasing the number of clusters will naturally decrease the SSE, but after a certain point, the reduction in SSE becomes minimal, indicating that additional clusters are not providing substantial better fit to the data. In this research, the Elbow Method was applied to determine that five clusters provided the most effective categorization of the songs.The same method can be used to choose the number of parameters in other data-driven models, such as the number of principal components to describe a data set. Therefore, in order to examine how many clusters could categorize the songs most effectively, the elbow method is being used.

```
import pandas as pd
import seaborn as sns
from sklearn.datasets import load iris
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

data = pd.read_csv("data.csv")
iris = load_iris()
X = pd.DataFrame(iris.data,columns=iris['feature_names'])

data = X[['sepal length (cm)', 'sepal width (cm)','petal length (cm)']]

sse={}
for k in range(1,30):
    kmeans = KMeans(n_clusters=k, max_iter=1000).fit(data)
```

DOI: 10.54254/2755-2721/52/20241657

```
    data["clusters"]= kmeans.labels_

    sse[k] = kmeans.inertia_
plt.figure()
plt.plot(list(sse.keys()),list(sse.values()))
plt.xlabel("Number of cluster")
plt.ylabel("SSE")
plt.show()
```

In the coding part, the first step is to import the libraries needed for the elbow method, including Panda, Seaborn, Iris, k-mean and matplot. The choice of Python libraries in this research was guided by their specific functionalities that facilitated the data analysis and clustering process. Pandas, a popular data manipulation library, was used for its efficient DataFrame structure that allows for easy data cleaning and analysis. The StandardScaler function from the sklearn library was used to standardize the numerical values in the data set, ensuring all parameters contribute equally to the clustering process. The sklearn library was also used for its implementation of the K-means algorithm, which is known for its efficiency on large datasets. For visualizing the clustering results, Matplotlib was used for its flexibility and control over every aspect of the figure, and PCA (Principal Component Analysis) from sklearn was used to reduce the dimensionality of the data for visualization. These libraries collectively provided the necessary tools to carry out the clustering analysis and build the recommendation system. The second step is to import the data set and read it through Panda. Then, write a "for" loop and let the computer calculate every sum of squared errors(SSE) value from 1 cluster to 30 clusters. Finally let the computer print a plot which has its x-axis representing the number of clusters and y-axis representing the SSE value. The calculation method of SSE is shown below, which is calculating the sum of the square of the difference of the distance between each data point and their cluster center.

$$\text{SSE} = \sum_{i=1}^{k} \sum_{p \in Ci} |p - mi|^2$$

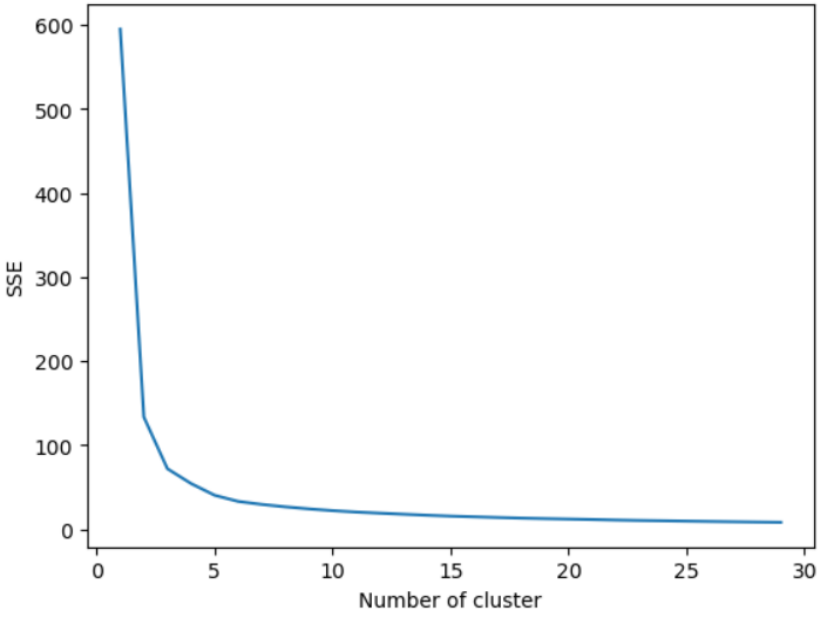

Figure 3. The graph of elbow method result

As shown in Fig.3, the turning point of this graph is when the cluster number equals 5, because after this point, the slope of this curve does not change rapidly. Therefore, the best quantity of clusters is 5.

After the number of clusters is determined, the clustering code can be finished:

DOI: 10.54254/2755-2721/52/20241657

```
import os
import numpy as np
import pandas as pd
import seaborn as sns
import plotly.express as px
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.manifold import TSNE
from sklearn.decomposition import PCA
from sklearn.metrics import euclidean_distances
from scipy.spatial.distance import cdist
import warnings
warnings.filterwarnings("ignore")

data = pd.read_csv("data.csv")
song_cluster_pipeline = Pipeline([("scaler", StandardScaler()),
                                  ("kmeans", KMeans(n_clusters=5, verbose=False))], verbose=False)
X = data.select_dtypes(np.number)
number_cols = list(X.columns)
song_cluster_pipeline.fit(X)
song_cluster_labels = song_cluster_pipeline.predict(X)
data['cluster_label'] = song_cluster_labels

pca_pipeline = Pipeline([('scaler', StandardScaler()), ('PCA', PCA(n_components=2))])
song_embedding = pca_pipeline.fit_transform(X)
projection = pd.DataFrame(columns=["x", "y"], data=song_embedding)
projection["title"] = data["name"]
projection["cluster"] = data['cluster_label']

fig = px.scatter(projection, x="x", y="y", color='cluster', hover_data=['x', 'y', "title"])
fig.show()
```

The first step of clustering is also importing the Python libraries. The libraries being used are mainly divided into 3 groups. The first group is for analyzing the data, such as panda and standard scaler. The second group is the most important components, and they serve for the clustering pipeline, such as pipeline and k-mean. Last but not least, one group is used to create an image of the clustering result, such as matplot and PCA. After importing all of the libraries required, panda could read the imported data set, and the standard scalar could standardize the numerical values. This step is the key to success, because, before this steps, all of the parameters have different scales, so some of them will naturally have a higher weight than the others. However, this is not what the project initially wanted to achieve. Hence, adding a code to automatically standardize the data is necessary. The next step is to build the pipeline, and start clustering every single data and visualize the clusters with PCA. The image below is the result of clustering(Fig.4)

DOI: 10.54254/2755-2721/52/20241657

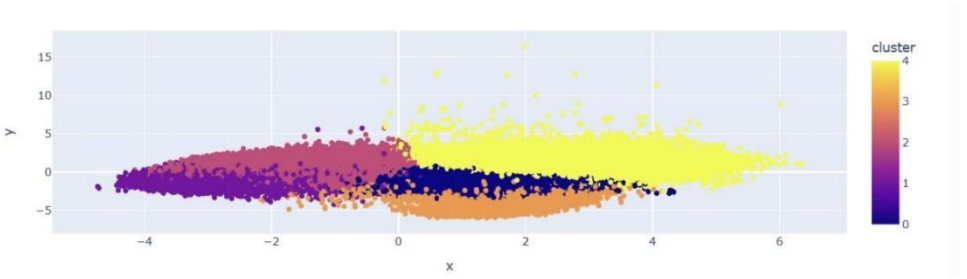

Figure 4. The result of clustering

In this picture, it clearly classify 170653 data points into clusters 0 to 4. Although there are a couple of outliers in this graph, most of the data points are arranged close to each other. Therefore, building a recommending system based on this type of clustering is feasible.

The last step of this project is to build a recommending system. The first step is letting the user input a song from the data set so the parameters of the song are accessible. Therefore, the program has to identify if the song is in the data set. If so, it could enter the second step. Then the program will locate the position of the imputed song and find the parameters of the it and which cluster this song belongs to. After that, the program will calculate the Euclidean distance between the input song and every other song in its cluster. Finally, print 10 songs that have the closest Euclidean distance to the input song. The calculation of Euclidean distance is shown below, which refers to the distance between 2 points in a n-dimensional Euclidean space . The code of recommending system is shown below [9].

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(xi - yi)^2} = \sqrt{(x1 - y1)^2 + (x2 - y2)^2 + \ldots + (xn - yn)^2}$$

```
from scipy.spatial.distance import cdist

def recommend_songs(song_title, data, projection):
    if song_title not in data['name'].values:
        return "The song title you entered is not in the dataset. Please enter a valid song title."

    song_cluster = data.loc[data['name'] == song_title, 'cluster_label'].values[0]

    cluster_data = data[data['cluster_label'] == song_cluster]

    song_vector = projection.loc[projection['title'] == song_title, ['x', 'y']].values[0]
    cluster_data['distance'] = cluster_data.apply(lambda row: cdist([song_vector],
                                                  [projection.loc[projection['title'] == row['name'],
                                                      ['x', 'y']].values[0]],
                                                  'euclidean'), axis=1)

    closest_songs = cluster_data.sort_values(by='distance', ascending=True)['name'][1:11]

    return closest_songs

print(recommend_songs('Melancholy', data, projection))
```

5. Discussion

The foremost outcome of this project is the song clustering result and the music recommending system. From the result, it is not hard to interpret that building a recommending system with an open song data

DOI: 10.54254/2755-2721/52/20241657

set utilizing Python is feasible. Clustering not only offers a clearer and more visualized perspective for the data set, but it also helps in increasing the working efficiency of the recommending system because the computer does not have to calculate the distance between the input song and every other song in the data set. Instead, it only needs to calculate and compare within the clusters. This also makes the recommending system more friendly to people who do not have a high quality device.

The generalizing ability of the results is limited because the data set only includes the songs from 1920 to 2020 which Spotify has on its platform. This means a user is not able to input or get any recommendations outside this data set. Therefore, if a user does not use Spotify to listen to music or like the latest songs, they will not get a proper result. Further studies should take the other songs into account in order to let more users experience the music recommending system and have a broader list of songs to either input or recommend. The methodology could still remain the same but combine more data sets with the same parameter and redo the elbow method and clustering.

Furthermore, the approach used in this project, which involves using clustering algorithms to categorize data and then making recommendations based on these categories, could be applied to a wide range of other recommendation systems. For example, in an e-commerce setting, products could be clustered based on various features such as price, brand, and customer reviews. Then, when a customer views a product, the system could recommend other products from the same cluster. Similarly, in a movie recommendation system, movies could be clustered based on genres, actors, directors, and viewer ratings. When a user watches a movie, the system could recommend other movies from the same cluster.

Nevertheless, other clustering methods could also achieve the same goal. As for the use of other clustering algorithms, there are several options that could potentially improve the performance of the recommendation system. For instance, hierarchical clustering could be used to create a hierarchy of clusters, which might provide more nuanced recommendations. Density-based clustering algorithms like DBSCAN could be used to handle datasets with noise and outliers, as they do not require specifying the number of clusters beforehand and can discover clusters of arbitrary shape. However, the choice of clustering algorithm would largely depend on the specific characteristics of the data set and the computational resources available. It would be interesting to compare the performance of different clustering algorithms in future research.

6. Conclusion

This project is aimed to solve a common problem for most of the current recommending systems. The issue is that these recommending systems are built on for-profit organizations such as music apps. These recommendation systems would not only consider the users' preferences but also manage to introduce the "membership only" songs to the user. In this way, the users are more likely to purchase a membership to listen to the recommended songs and further make the company gain more benefits. The approach used in this project, which involves clustering songs based on various auditory parameters and then recommending songs from the same cluster, could significantly improve the user experience in music recommendation systems. By focusing on the user's preferences rather than commercial interests, this approach ensures that the recommendations are tailored to the user's tastes, potentially leading to higher user satisfaction and engagement. The final outcome of this project is the clustering of the song data set and a non-profit open recommending system accessible to the public. For the music field, it allows more users to discover the songs that they may potentially enjoy by recommending them based on the auditory parameters.

Moreover, the use of clustering reduces the computational load of the recommendation process, as the system only needs to consider songs within the same cluster rather than the entire dataset. This could result in faster recommendation times, improving the responsiveness of the system and further enhancing the user experience.

In terms of the development of recommendation systems which belongs to the field of data science, this project demonstrates the feasibility and effectiveness of using clustering algorithms for recommendation tasks. It provides a practical example of how open datasets and Python libraries can be used to build a recommendation system, contributing to the body of knowledge in this field. Furthermore,

DOI: 10.54254/2755-2721/52/20241657

the project highlights the importance of data preprocessing, such as standardization, in ensuring that all parameters contribute equally to the clustering process. These insights could inform the design and implementation of future recommendation systems.

References

[1] Velankar, M., & Kulkarni, P. (2022). Music Recommendation Systems: Overview and Challenges. In Advances in Speech and Music Technology, 1–16.

[2] Roy, D., & Dutta, M. (2023). A systematic review and research perspective on recommender systems. ICAT 2022: Communications in Computer and Information Science 89–102.

[3] Lu, J., Wu, D., Mao, M., Wang, W., & Zhang, G. (2015). Recommender system application developments: A survey. Decision Support Systems, 74, 12–32.

[4] Oyelade, O. J., Oladipupo, O. O., & Obagbuwa, I. C. (2010). Application of K-Means Clustering algorithm for prediction of Students' Academic Performance. International Journal of Computer Science and Information Security, 7(1), 292–295.

[5] Amini, A., Wah, T. Y., & Saboohi, H. (2014). On Density-Based Data Streams Clustering Algorithms: A Survey. Journal of Computer Science and Technology, 29(1), 116–141.

[6] Leskovec, J., Rajaraman, A., & Ullman, J. D. (2014). Mining of Massive Datasets. Cambridge University Press.

[7] Schedl, M., Zamani, H., Chen, C., Deldjoo, Y., & Elahi, M. (2018). Current challenges and visions in music recommender systems research. International Journal of Multimedia Information Retrieval, 7, 95–1163.

[8] Shi, J. (2021). Music Recommendation Algorithm Based on Multidimensional Time-Series Model Analysis. Journal of Physics: Conference Series, 1760(1).

[9] Valdiviezo-Diaz, P., & Chicaiza, J. (2023). Enhanced Books Recommendation Using Clustering Techniques and Knowledge Graphs. In ICAT 2022: Communications in Computer and Information Science , 89–102.

[10] Uher, J. (2022). Functions of units, scales and quantitative data: Fundamental differences in numerical traceability between sciences. Quality & Quantity, 56, 2519–25481.

[11] Li, Y., Liu, K., Satapathy, R., Wang, S., & Cambria, E. (2023). Recent Developments in Recommender Systems: A Survey. International Journal of Multimedia Information Retrieval, 7, 95–1163.

[12] Xu, D., & Tian, Y. (2022). A Comprehensive Survey of Clustering Algorithms. Advances in Speech and Music Technology, 1–16.