İZMİR KÂTİP ÇELEBİ ÜNİVERSİTESİ
2010
FEN BİLİMLERİ ENSTİTÜSÜ

Spotify Veri Analizi
(2024)

Yazılım Mühendisliği Ana Bilim Dalı

Dönem Projesi

Engin Deniz SARIDEMİR

Y220234135

Proje Danışmanı: Prof. Dr. Öğr. Üyesi Aytuğ ONAN

Mayıs 2024

Spotify Şarkı ve Tür Veri Analizi ve Görselleştirme

Öz

Sayın Okuyucular,

Bu çalışma, Spotify'deki çeşitli müzik parçalarının yayın akış verilerini analiz etmeyi amaçlamaktadır. Veri seti, parçaların akış sayıları, listelerdeki varlık süreleri ve diğer özellikleri içermektedir. Projede, veriler önce temizlenmiş, ardından çeşitli istatistiksel analizler ve görselleştirme teknikleri kullanılarak incelenmiştir. Elde edilen bulgular, müzik parçalarının popülerlik düzeyleri ve pazar dinamikleri hakkında önemli bilgiler sağlamaktadır. Analizler sonucunda, en çok dinlenen şarkılar ve onların özellikleri detaylı bir şekilde irdelenmiştir. Bu çalışma, müzik endüstrisi profesyonellerine ve araştırmacılara, müzik trendlerini daha iyi anlama konusunda değerli içgörüler sunmaktadır. Saygılarımla,

Anahtar Sözcükler: Spotify, Müzik, Veri Analizi, Veri Görselleştirmesi

Analyzing and Visualizing Spotify Song and Genre Data

Abstract

Dear Readers,

This study aims to analyze the streaming data of various tracks on Spotify. The dataset includes information on the number of streams, presence on charts, and other characteristics of the tracks. The data was initially cleaned, followed by detailed statistical analyses and visualization techniques to examine the trends. The findings provide significant insights into the popularity levels of tracks and market dynamics. The analyses detailed the most listened tracks and their characteristics extensively. This research offers valuable insights to music industry professionals and researchers, enabling a better understanding of music trends.

Keywords: Spotify, Music, Data Visualization, Data Analysis

Teşekkür

Proje çalışmama katkılarından dolayı proje danışmanım Prof. Dr. Öğr. Üyesi Aytuğ ONAN'a teşekkür ederim.

İçindekiler

Şekiller Listesi

1.Giriş

Veri Biliminin Önemi:

Veri bilimi, dijital çağın en önemli disiplinlerinden biri olarak ön plana çıkmaktadır. Büyük verinin artan önemi ve karmaşıklığı, iş dünyasından akademiye, hükümet politikalarından günlük yaşam kararlarına kadar birçok alanda veri biliminin etkisini artırmaktadır. Bu disiplin, çeşitli veri kaynaklarından elde edilen büyük miktarda veriyi analiz ederek değerli bilgiler ve anlayışlar sağlar. Veri biliminin amacı, veriden anlam çıkarmak ve bu bilgileri stratejik karar alma, tahminleme ve optimizasyon süreçlerinde kullanmaktır.

Python'un Veri Bilimindeki Yeri:

Python, veri bilimi alanında en popüler programlama dillerinden biridir. Bu popülerliğin arkasındaki temel nedenler; Python'un okunabilirliği, esnekliği ve geniş kütüphane ekosistemi içerisinde veri analizi ve makine öğrenimi için güçlü araçlar sunmasıdır. Python, veri manipülasyonu (Pandas), sayısal hesaplamalar (NumPy), istatistiksel analiz (SciPy), veri görselleştirme (Matplotlib, Seaborn) ve makine öğrenimi (Scikit-Learn) gibi çeşitli alanlarda etkin çözümler sunar. Bu özellikler, Python'u hem amatörler hem de profesyoneller için tercih edilen bir dil haline getirmiştir.

Jupyter Notebook'un Rolü:

Jupyter Notebook, veri bilimciler arasında yaygın olarak kullanılan etkileşimli bir geliştirme ortamıdır. Kod yazma, not almak, görselleştirme yapmak ve sonuçları paylaşmak gibi işlemleri bir arada sunarak veri analizi sürecini kolaylaştırır ve daha etkileşimli hale getirir. Jupyter, analiz sürecini adım adım belgelemeye ve sonuçları anlaşılır bir şekilde sunmaya olanak tanır. Bu, özellikle akademik araştırmalar, veri bilimi eğitimi ve iş dünyasındaki raporlamalar için büyük bir avantajdır.

Bu Çalışmanın Amacı:

Bu makale, Python ve Jupyter Notebook kullanılarak yapılan veri analizinin temel adımlarını ve metodolojilerini, örnek bir Jupyter notebook üzerinden ayrıntılı bir şekilde ele almaktadır. Çalışma, veri yükleme, temizleme, analiz, görselleştirme ve sonuç çıkarma gibi temel süreçleri kapsamakta ve bu süreçlerin nasıl uygulanacağını göstermektedir. Ayrıca, kullanılan her bir kütüphane ve metodun veri bilimi pratiklerindeki önemi ve rolü detaylandırılmaktadır.

2. Metod

2.1.Verilerde Python Kütüphaneleri

2.1.1 Matplotlib ve Seaborn (Görselleştirme):

Matplotlib, Python'da statik, etkileşimli ve animasyonlu grafikler oluşturmak için kullanılan bir kütüphanedir. Bu kütüphane, veri görselleştirme için geniş bir fonksiyon seti sunar. Seaborn ise, Matplotlib tabanlı, daha yüksek seviyeli bir görselleştirme kütüphanesidir ve çekici, anlamlı istatistiksel grafikler oluşturmayı kolaylaştırır. Bu iki kütüphane, veri setlerinin görsel analizinde kullanılmakta ve veri hikayelerinin anlatılmasında etkili araçlar olarak ön plana çıkmaktadır.

2.1.2 NumPy (Sayısal Hesaplamalar):

NumPy, büyük, çok boyutlu diziler ve matrisler için destek sunan, Python programlama dilindeki temel paketlerden biridir. Bu kütüphane, veri bilimi uygulamalarında hızlı ve etkili sayısal hesaplamalar için yaygın olarak kullanılır. NumPy, veri analizinde temel algoritma ve fonksiyonlarıyla, özellikle büyük veri setleri üzerinde çalışırken hız ve verimlilik sağlar.

2.1.3 Pandas (Veri Manipülasyonu ve Analizi):

Pandas, veri manipülasyonu ve analizi için Python'da geliştirilmiş açık kaynak bir kütüphanedir. Pandas, DataFrame adı verilen etkili veri yapıları üzerinde çalışır. Bu kütüphane, veri okuma, temizleme, dönüştürme ve analiz işlemlerini kolaylaştırır. Pandas'ın sunduğu işlevsellik, veri bilimindeki temel görevler için vazgeçilmezdir ve genellikle veri setlerinin ilk işlenmesi ve keşfedici veri analizi aşamalarında kullanılır.

2.1.4 Jupyter Notebook'un Kullanımı:

Jupyter Notebook, veri bilimi projelerinde kod yazımı, not almak, sonuçları görselleştirmek ve paylaşmak için kullanılan etkileşimli bir web uygulamasıdır. Bu ortam, araştırmacılara ve veri bilimcilere kodları hücreler halinde düzenleme ve çalıştırma imkânı sunar, böylece analiz süreçlerinin adım adım izlenmesini ve

dokümante edilmesini sağlar. Jupyter, hem eğitim hem de profesyonel çalışmalarda tercih edilen bir araçtır, çünkü karmaşık veri analizlerini daha erişilebilir ve anlaşılır hale getirir.

2.2. Verileri Anlama ve İnceleme

2.2.1.Kütüphane İçe Aktarımı ve Ortam Ayarları:

İlk adımda, veri görselleştirme için Matplotlib ve Seaborn kütüphaneleri içe aktarılır. set_style ve set_palette metodları, grafiklerin görsel stilini belirlemek için kullanılır. NumPy ve Pandas, veri manipülasyonu ve analizi için temel araçlardır(Şekil2.1). Bu adım, veri analizi için gerekli kütüphanelerin kurulumunu ve yapılandırılmasını kapsar.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import style
import seaborn as sns
import sklearn.cluster as skcluster
from scipy.stats import f_oneway
from sklearn.inspection import DecisionBoundaryDisplay
from sklearn.preprocessing import MinMaxScaler
```

Şekil2.1.: Kullanılan kütüphanelerin listesi

2.2.2.Veri Yükleme:

Bu kod satırı, **Pandas** kütüphanesinin **read_csv()** fonksiyonunu kullanarak bir CSV dosyasını okur ve verileri bir DataFrame'e yükler. **read_csv()** fonksiyonu, CSV dosyasını okumak için kullanılırken, dosyanın yolunu ve dosyanın karakter kodlamasını belirtmek için parametreler alır.

"C:\\Users\\engin\\OneDrive\\Masaüstü\\Bitirme projesi\\spotify-2023.csv" ifadesi, okunacak CSV dosyasının tam yolunu belirtir. **encoding="latin-1"** parametresi, dosyanın karakter kodlamasını belirtir. Burada "latin-1" karakter kodlaması kullanılmıştır, çünkü bazı karakterlerin Latin-1 karakter kümesinde yer alması ve bu karakterlerin doğru şekilde okunması gerekmektedir.

Sonuç olarak, **df** değişkeni, verileri içeren bir Pandas DataFrame'e atanır ve bu DataFrame daha sonra veri analizi ve işlemleri için kullanılabilir. (Şekil2.2).

```
df = pd.read_csv("C:\\Users\\engin\\OneDrive\\Masaüstü\\Bitirme projesi\\spotify-2023.csv", encoding="latin-1")
```

Şekil2.2.: Veri Dosyasının Okunması ve DataFrame Oluşturulması

2.2.3. Tekil Şarkı İsimlerinin Alınması:

DataFrame içindeki **"track_name"** sütunundaki benzersiz (unique) şarkı isimlerini döndürür. **unique()** fonksiyonu, belirtilen sütundaki benzersiz değerleri döndürür.

Örneğin, eğer veri setinde birçok satırda aynı şarkı ismi varsa, bu fonksiyon bu ismi bir kez alır ve tekrar eden isimleri atar. Bu, veri setinde hangi şarkıların bulunduğunu anlamak için faydalıdır ve özellikle veri setindeki benzersiz içeriği incelemek gerektiğinde kullanışlıdır.

Sonuç olarak, bu kod satırı, DataFrame içindeki **"track_name"** sütunundaki tekil şarkı isimlerini döndürür. (Şekil2.3.)

```
In [3]: df["track_name"].unique()

Out[3]: array(['Seven (feat. Latto) (Explicit Ver.)', 'LALA', 'vampire',
       'Cruel Summer', 'WHERE SHE GOES', 'Sprinter', 'Ella Baila Sola',
       'Columbia', 'fukumean', 'La Bebe - Remix', 'un x100to',
       'Super Shy', 'Flowers', 'Daylight', 'As It Was', 'Kill Bill',
       'Cupid - Twin Ver.',
       'What Was I Made For? [From The Motion Picture "Barbie"]',
       'Classy 101', 'Like Crazy', 'LADY GAGA',
       'I Can See You (Taylori¿½I¿½i¿½s Version) (From The ',
       'I Wanna Be Yours', 'Peso Pluma: Bzrp Music Sessions, Vol. 55',
       'Popular (with Playboi Carti & Madonna) - The Idol Vol. 1 (Music from the HBO Original Series)',
       'SABOR FRESA', 'Calm Down (with Selena Gomez)', 'MOJABI GHOST',
       'Last Night', 'Dance The Night (From Barbie The Album)', 'Rush',
       'TULUM', "Creepin'", 'Anti-Hero', 'TQG', 'Los del Espacio',
       'Fri¿½i¿½gil (feat. Grupo Front', 'Blank Space', 'Style', 'TQM',
       'El Azul', 'Sunflower - Spider-Man: Into the Spider-Verse',
       "I'm Good (Blue)", 'See You Again',
       'Barbie World (with Aqua) [From Barbie The Album]',
       'Angels Like You', "I Ain't Worried", 'Die For You', 'Starboy',
       'Die For You - Remix', 'El Cielo', "Baby Don't Hurt Me",
```

Şekil2.3.: Veri Seti Bilgilerinin İncelenmesi

2.2.4. DataFrame Bilgi Alımı:

DataFrame'in genel bilgilerini sağlar. info() fonksiyonu, DataFrame hakkında özet bilgiler sunar. Bu bilgiler arasında sütunların adları, her sütundaki toplam değer sayısı, her sütundaki eksik değerlerin sayısı, sütun veri tipleri ve bellek kullanımı bulunur.

Bu bilgiler, veri setinin boyutu, veri tipleri ve eksik değerlerin varlığı gibi önemli özellikler hakkında genel bir fikir verir. Bu, veri setinin genel yapısını anlamak için yararlıdır ve veri analizi sürecinde veri setinin ön işleme gereksinimlerini belirlemeye yardımcı olur.

Sonuç olarak, bu kod satırı, DataFrame hakkında genel bilgiler sağlar ve veri setinin temel özelliklerini özetler. (Şekil2.4.)

```
In [4]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 953 entries, 0 to 952
Data columns (total 24 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   track_name            953 non-null    object
 1   artist(s)_name        953 non-null    object
 2   artist_count          953 non-null    int64
 3   released_year         953 non-null    int64
 4   released_month        953 non-null    int64
 5   released_day          953 non-null    int64
 6   in_spotify_playlists  953 non-null    int64
 7   in_spotify_charts     953 non-null    int64
 8   streams               953 non-null    object
 9   in_apple_playlists    953 non-null    int64
 10  in_apple_charts       953 non-null    int64
 11  in_deezer_playlists   953 non-null    object
 12  in_deezer_charts      953 non-null    int64
 13  in_shazam_charts      903 non-null    object
 14  bpm                   953 non-null    int64
 15  key                   858 non-null    object
 16  mode                  953 non-null    object
 17  danceability_%        953 non-null    int64
 18  valence_%             953 non-null    int64
 19  energy_%              953 non-null    int64
 20  acousticness_%        953 non-null    int64
 21  instrumentalness_%    953 non-null    int64
 22  liveness_%            953 non-null    int64
 23  speechiness_%         953 non-null    int64
dtypes: int64(17), object(7)
memory usage: 178.8+ KB
```

Şekil2.4.: DataFrame Bilgi Alımı

2.2.5. Eksik Değerlerin Sayısının Hesaplanması:

DataFrame içindeki her bir sütundaki eksik değerlerin sayısını hesaplar. **isna()** fonksiyonu, her bir hücrenin eksik olup olmadığını kontrol eder ve eksik (NaN) değerler için **True**, eksik olmayan değerler için **False** döndürür. **sum()** fonksiyonu

ise her sütundaki **True** değerlerin sayısını toplar, yani her bir sütundaki eksik değerlerin toplam sayısını verir.

Bu bilgi, veri setindeki eksik veri miktarını anlamak için önemlidir. Eksik veriler, veri analizi ve modelleme işlemleri için bir sorun olabilir ve bu nedenle bu değerlerin belirlenmesi ve bunlarla nasıl başa çıkılacağının belirlenmesi önemlidir.

Sonuç olarak, bu kod satırı, DataFrame içindeki her bir sütundaki eksik değerlerin sayısını hesaplar ve bunları bir dizi olarak döndürür. Bu, eksik verilerin analiz edilmesi ve eksik değerlerle başa çıkılması için ilk adımdır. (Şekil2.5.)

```
In [5]: df.isna().sum()

Out[5]: track_name              0
        artist(s)_name          0
        artist_count            0
        released_year           0
        released_month          0
        released_day            0
        in_spotify_playlists    0
        in_spotify_charts       0
        streams                 0
        in_apple_playlists      0
        in_apple_charts         0
        in_deezer_playlists     0
        in_deezer_charts        0
        in_shazam_charts       50
        bpm                     0
        key                    95
        mode                    0
        danceability_%          0
        valence_%               0
        energy_%                0
        acousticness_%          0
        instrumentalness_%      0
        liveness_%              0
        speechiness_%           0
        dtype: int64
```

Şekil2.5.: Eksik Değerlerin Sayısının Hesaplanması

2.2.6.Eksik Değerlerin Düşürülmesi:

DataFrame içindeki eksik değerleri içeren satırların (gözlemlerin) düşürülmesini sağlar. **dropna()** fonksiyonu, eksik değer içeren satırları kaldırır.

inplace=True parametresi, değişikliklerin DataFrame'e doğrudan uygulanmasını sağlar. Böylece, DataFrame üzerinde işlem yapıldıktan sonra **df** değişkeninin güncellenmiş hali saklanır.

Eksik değerlerin düşürülmesi, eksik verilerle başa çıkmak için bir yöntemdir. Ancak, bu yaklaşım bazen veri kaybına neden olabilir ve veri setinin analizi için yeterli veri miktarını azaltabilir. Bu nedenle, eksik değerlerin neden kaynaklandığını ve veri setinin doğasını anlamak önemlidir.

Sonuç olarak, bu kod satırı, DataFrame içindeki eksik değerleri içeren satırları kaldırır ve DataFrame'in güncellenmiş bir versiyonunu döndürür. (Şekil2.6.)

```
In [6]: df.dropna(inplace=True)
```

Şekil2.6.: Eksik Değerlerin Düşürülmesi

2.2.7. DataFrame Sütunlarının Dolaşılması ve Benzersiz Değerlerin Alınması:

Pandas DataFrame içindeki her sütunu dolaşır ve her bir sütunun benzersiz değerlerini yazdırır. **for** döngüsü, DataFrame içindeki her sütunu tek tek işlemek için kullanılır. Döngü her bir sütunun adını **col** değişkenine atar ve ardından bu sütunun benzersiz değerlerini alır.

İlk olarak, **print(df[col])** ifadesi, DataFrame içindeki mevcut sütunu yazdırır. Bu, sütunun tüm değerlerini görmek için kullanılır ve veri setinin yapısını anlamak için faydalıdır.

Ardından, **print(df[col].unique())** ifadesi, ilgili sütundaki benzersiz değerleri yazdırır. Bu, sütundaki farklı kategorik veya kümelenmiş veri türlerini anlamak için kullanışlıdır. Her bir benzersiz değer, sütundaki farklı kategorileri veya özellikleri temsil eder.

Bu kod parçası, veri setinin içeriğini anlamak için önemlidir. Her bir sütunun benzersiz değerlerinin belirlenmesi, veri setinin içeriği hakkında bilgi edinmek ve veri setini analiz etmek için gereklidir. Bu, veri setindeki özelliklerin türünü, dağılımını ve benzersiz değerlerini anlamak için bir adım sağlar. (Şekil2.7)

```
In [8]: for col in df:
            print (df[col])
            print(df[col].unique())
```

```
0      Seven (feat. Latto) (Explicit Ver.)
1                                     LALA
2                                  vampire
3                             Cruel Summer
4                            WHERE SHE GOES
                      ...
948                          My Mind & Me
949              Bigger Than The Whole Sky
950                   A Veces (feat. Feid)
951                          En La De Ella
952                                  Alone
Name: track_name, Length: 817, dtype: object
['Seven (feat. Latto) (Explicit Ver.)' 'LALA' 'vampire' 'Cruel Summer'
 'WHERE SHE GOES' 'Sprinter' 'Ella Baila Sola' 'Columbia' 'fukumean'
 'La Bebe - Remix' 'un x100to' 'Super Shy' 'Daylight' 'Kill Bill'
 'Cupid - Twin Ver.' 'Classy 101' 'Like Crazy' 'LADY GAGA'
 'I Can See You (Taylori¿¾i¿¾i¿¾s Version) (From The '
 'Peso Pluma: Bzrp Music Sessions, Vol. 55'
 'Popular (with Playboi Carti & Madonna) - The Idol Vol. 1 (Music from the HBO Original Series)'
```

Şekil2.7.: Sütunlarının Dolaşılması ve Benzersiz Değerlerin Alınması

2.2.8 Benzersiz Akış Değerlerinin Alınması:

DataFrame içindeki "streams" adlı sütundaki benzersiz (unique) akış değerlerini döndürür. "streams" sütunu genellikle şarkıların veya diğer medya öğelerinin ne kadar dinlendiğini veya izlendiğini temsil eder.

unique() fonksiyonu, belirtilen sütundaki benzersiz değerleri döndürür. Bu, sütundaki farklı akış miktarlarını anlamak için kullanışlıdır. Her bir benzersiz değer, veri setinde farklı düzeylerde akışı temsil eder.

Bu bilgi, veri setindeki akış verilerinin dağılımını ve çeşitliliğini anlamak için önemlidir. Hangi akış miktarlarının daha yaygın olduğunu veya nadir olduğunu belirlemek, veri setinin analiz edilmesi ve anlaşılması için önemlidir. Ayrıca, bu benzersiz değerler, veri setinin ölçeklendirilmesi ve modelleme işlemleri için de kullanılabilir. (Şekil2.8.)

```
In [9]: df["streams"].unique()

Out[9]: array(['141381703', '133716286', '140003974', '800840817', '303236322',
       '183706234', '725980112', '58149378', '95217315', '553634067',
       '505671438', '58255150', '387570742', '1163093654', '496795686',
       '335222234', '363369738', '86444842', '52135248', '200647221',
       '115364561', '78300654', '899183384', '61245289', '429829812',
       '127408954', '22581161', '52294266', '843957510', '999748277',
       '618990393', '188933502', '1355959075', '786181836', '176553476',
       '354495408', '2808096550', '1109433169', '1047101291', '570515054',
       '1647990401', '2565529693', '518745108', '107753850', '177740666',
       '153372011', '57876440', '256483385', '1214083358', '111947664',
       '156338624', '720434240', '357925728', '674072710', '1755214421',
       '404562836', '373199958', '14780425', '39578178', '54266102',
       '751134527', '1356565093', '1592909789', '635412045', '1230675890',
       '585695368', '43857627', '2009094673', '600976848', '39709092',
       '39228929', '2665343922', '223633238', '1440757818', '165484133',
       '58054811', '157058870', '95131998', '250305248', '685032533',
       '38411956', '144565150', '127567540', '399686758', '983637508',
       '118482347', '882831184', '286400165', '172825906', '1241559043',
       '29562220', '77309611', '1605224506', '1116995633', '838079900',
```

Şekil2.8.: Benzersiz Akış Değerlerinin Alınması

2.2.9. Satırın Düşürülmesi:

Bu kod satırı, DataFrame'deki belirli bir satırın (gözlem) kaldırılmasını sağlar. drop() fonksiyonu, belirtilen satırın indeksini kullanarak bu satırı DataFrame'den kaldırır.

prod_df, güncellenmiş DataFrame'i temsil eden yeni bir değişkene atanır. drop() fonksiyonu, DataFrame üzerinde doğrudan değişiklik yapmaz, bu nedenle değiştirilmiş DataFrame bir değişkene atanmalıdır.

Bu tür bir işlem, bazı durumlarda analiz veya modelleme için geçersiz veya gereksiz bir gözlem olduğunda veya veri setinden çıkarılması gerektiğinde kullanılır. Bu, veri setinin temizlenmesi veya analiz edilmesi için bir ön işlem adımı olabilir. Özellikle, belirli bir satırın veri setinden çıkarılması, o satırdaki verilerin analizine veya modele dahil edilmesine karar verildiğinde kullanışlıdır.

```
In [10]: prod_df= df.drop(574)
```

Şekil2.9.: prod_df= df.drop kodu ile satırın düşürülmesi

2.2.10. Benzersiz Akış Değerlerinin Alınması (Güncellenmiş DataFrame İçin):

Bu kod satırı, güncellenmiş DataFrame olan **prod_df** içindeki "streams" adlı sütundaki benzersiz (unique) akış değerlerini döndürür. Önceki adımda belirli bir satırın düşürülmesiyle DataFrame güncellenmiş ve artık bu güncellenmiş DataFrame üzerinde işlem yapılıyor.

unique() fonksiyonu, belirtilen sütundaki benzersiz değerleri döndürür. Bu, güncellenmiş DataFrame içindeki farklı akış miktarlarını anlamak için kullanışlıdır. Her bir benzersiz değer, veri setinde farklı düzeylerde akışı temsil eder.

```
In [11]: prod_df["streams"].unique()

Out[11]: array(['141381703', '133716286', '140003974', '800840817', '303236322',
                '183706234', '725980112', '58149378', '95217315', '553634067',
                '505671438', '58255150', '387570742', '1163093654', '496795686',
                '335222234', '363369738', '86444842', '52135248', '200647221',
                '115364561', '78300654', '899183384', '61245289', '429829812',
                '127408954', '22581161', '52294266', '843957510', '999748277',
                '618990393', '188933502', '1355959075', '786181836', '176553476',
                '354495408', '2808096550', '1109433169', '1047101291', '570515054',
                '1647990401', '2565529693', '518745108', '107753850', '177740666',
                '153372011', '57876440', '256483385', '1214083358', '111947664',
                '156338624', '720434240', '357925728', '674072710', '1755214421',
                '404562836', '373199958', '14780425', '39578178', '54266102',
                '751134527', '1356565093', '1592909789', '635412045', '1230675890',
                '585695368', '43857627', '2009094673', '600976848', '39709092',
                '39228929', '2665343922', '223633238', '1440757818', '165484133',
                '58054811', '157058870', '95131998', '250305248', '685032533',
                '38411956', '144565150', '127567540', '399686758', '983637508',
                '118482347', '882831184', '286400165', '172825906', '1241559043',
                '29562220', '77309611', '1605224506', '1116995633', '838079900',
```

Şekil2.10.: Benzersiz Akış Değerlerinin Alınması (Güncellenmiş DataFrame İçin)

2.2.11 Veri Tipi Dönüşümleri ve Sütun İşlemleri:

Bu kod bloğu, belirli sütunlarda veri tipi dönüşümleri yapar ve ardından belirli bir sütunun veri tipini değiştirir.

İlk for döngüsü, convert_to_string sözlüğünde belirtilen sütunlar için bir döngü başlatır. Her bir sütun adı için, ilgili sütunu dize (string) veri türüne dönüştürür. Bu, sütundaki verilerin metin veya kategorik olarak ele alınması gerektiğinde kullanışlıdır.

İkinci for döngüsü, convert_to_numeric sözlüğünde belirtilen sütunlar için bir döngü başlatır. Her bir sütun adı için, virgül işaretlerini kaldırır ve ardından sütunu tamsayıya (integer) dönüştürür. Bu, sütundaki verilerin sayısal olarak ele alınması gerektiğinde kullanışlıdır.

Son olarak, "streams" sütunu için ayrı bir işlem yapılır. Bu sütun, akış miktarlarını temsil eder ve tamsayı değerler içermelidir. Bu nedenle, astype() fonksiyonu kullanılarak bu sütun tamsayıya dönüştürülür.

Bu tür veri tipi dönüşümleri, veri setindeki sütunların doğru şekilde işlenmesini sağlar ve analiz edilmesini kolaylaştırır. Özellikle, sayısal sütunlar için uygun veri tiplerine dönüştürme, veri setinin doğruluğunu artırır ve analiz sonuçlarını daha anlamlı hale getirir.

```
convert_to_string = {'track_name', 'artist(s)_name', 'key', 'mode'}
convert_to_numeric = {"in_deezer_playlists", "in_shazam_charts"}
```

```
for col_name in convert_to_string:
    df[col_name] = df[col_name].astype("string")
```

```
for col_name in convert_to_numeric:
    df[col_name] = df[col_name].str.replace(',', '').astype(np.int64)
df["streams"] = df["streams"].astype(np.int64)
```

Şekil2.11.: Veri Tipi Dönüşümleri ve Sütun İşlemleri

2.2.12. Sütunun Düşürülmesi:

Bu kod satırı, DataFrame'deki belirli bir sütunun (özellikle "track_name" sütunu) kaldırılmasını sağlar. drop() fonksiyonu, belirtilen sütunu DataFrame'den kaldırır.

"track_name" parametresi, kaldırılacak sütunun adını belirtir. axis=1 parametresi, sütunu sütunlar boyunca (yatay olarak) kaldırmak için kullanılır. inplace=True parametresi, değişikliklerin DataFrame'e doğrudan uygulanmasını sağlar. Böylece, DataFrame üzerinde yapılan değişikliklerin kalıcı olmasını sağlar.

Belirli bir sütunun düşürülmesi, genellikle analizde veya modellemede kullanılmayan veya gereksiz olan bir özelliğin kaldırılması gerektiğinde kullanılır. Bu, veri setinin boyutunu azaltabilir ve analiz veya modelleme sürecini basitleştirebilir.

```
: df.drop("track_name", axis=1, inplace=True)
```

Şekil2.12.: Sütunun Düşürülmesi

2.2.13. Akış Sayılarına Göre Sıralama:

Bu kod satırı, DataFrame'deki verileri "streams" sütunu temel alınarak azalan sırada sıralar. sort_values() fonksiyonu, belirtilen sütuna göre verileri sıralamak için kullanılır. ascending=False parametresi, sıralamanın azalan (yüksekten düşüğe) şekilde yapılmasını sağlar.

Bu işlem, şarkıların veya içeriklerin popülerliklerine göre sıralanmasını sağlar. Akış sayıları yüksek olan şarkılar veya içerikler, DataFrame'in en üstüne yerleştirilir. Bu, en popüler şarkıları veya içerikleri hızlı bir şekilde görmek ve analiz etmek için kullanışlı bir yöntemdir.

```
: df.sort_values("streams", ascending = False)
```

:

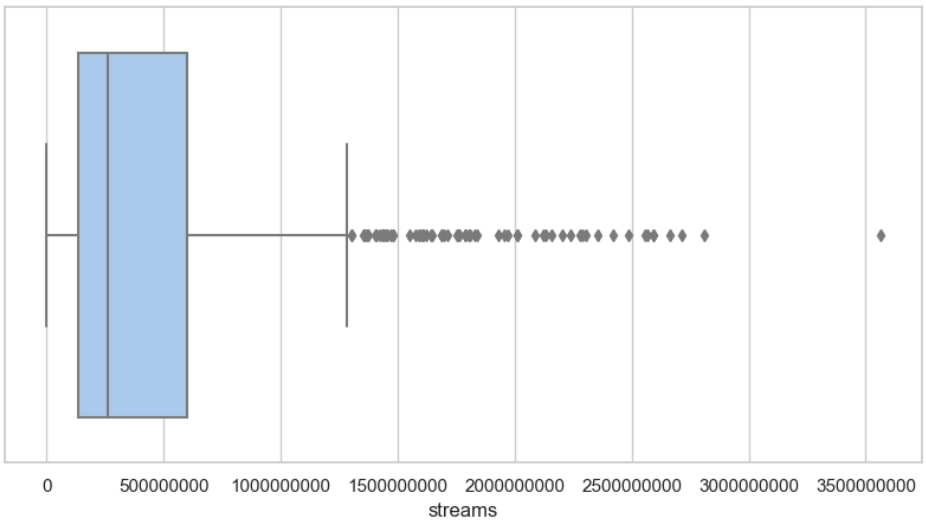
	artist(s)_name	artist_count	released_year	released_month	released_day	in_spotify_playlists	in_spotify_charts	streams	in_apple_playlists	in_apple_cl
179	Ed Sheeran	1	2017	1	6	32181	10	3562543890	33	
41	Post Malone, Swae Lee	2	2018	10	9	24094	78	2808096550	372	
162	Drake, WizKid, Kyla	3	2016	4	4	43257	24	2713922350	433	
84	Justin Bieber, The Kid Laroi	2	2021	7	9	17050	36	2665343922	492	
140	Imagine Dragons	1	2017	1	31	18986	23	2594040133	250	
...	...	...	...	...	...	...	...	...	...	
30	Troye Sivan	1	2023	7	13	864	78	22581161	71	
68	Post Malone	1	2023	7	14	410	36	14780425	36	

Şekil2.13.: Akış Sayılarına Göre Sıralama

2.2.14. Akış Sayılarının Kutu Grafiği Görselleştirilmesi:

Bu kod bloğu, Seaborn ve Matplotlib kütüphaneleri kullanılarak bir veri setindeki "streams" sütununun dağılımını görselleştirmek için kullanılmaktadır. İlk olarak, sns.set fonksiyonu ile grafik için görsel stil ayarlamaları yapılır; burada "whitegrid" stili seçilerek arka plan beyaz ve ızgaralı olarak belirlenir, "pastel" paleti ile de renkler yumuşak tonlarda ayarlanır. plt.figure ile grafik boyutu 10 inç genişlik ve 5 inç yükseklik olarak tanımlanır. sns.boxplot fonksiyonu kullanılarak "streams" sütunundaki değerler için bir kutu grafiği çizilir ve bu grafik yatay olarak konumlandırılır (orient="h"). plt.ticklabel_format ile x eksenindeki etiketlerin formatı düz metin olarak ayarlanır, bu da büyük sayıların daha kolay okunmasını sağlar. Son adımda plt.show() ile grafik ekrana getirilir. Kutu grafiği, veri dağılımının merkezi eğilimini, çeyrekliklerini ve aykırı değerlerini göstererek, veri setindeki "streams" değerlerinin genel özelliklerine dair fikir verir. Bu görselleştirme, veri analizinde önemli bir rol oynar çünkü verinin özetlenmiş bir görünümünü sunar ve olası dışsal değerlerin tespit edilmesine yardımcı olur.

```
sns.set(style="whitegrid", palette="pastel")
plt.figure(figsize=(10, 5))
sns.boxplot(x="streams", data=df, orient="h")
plt.ticklabel_format(style='plain', axis='x')
plt.show()
```



Şekil2.14.: Akış Sayılarının Kutu Grafiği Görselleştirilmesi

2.2.15. Veri Filtreleme ve Sıralama:

Bu kod bloğu, veri setindeki "streams" sütunundaki değerlerin alt yüzde 1'lik dilimini belirlemek ve bu değerin üzerindeki verilerle bir alt veri seti oluşturmak için kullanılmaktadır. İlk olarak, df["streams"].quantile(0.01) ifadesi ile "streams" sütunundaki değerlerin alt yüzde 1'lik kesim noktası hesaplanır ve bu değer lower_bound değişkenine atanır. Bu işlem, veri setindeki potansiyel aykırı düşük

değerleri tanımlamak için kullanılır. Daha sonra, df[df["streams"] > lower_bound] ifadesi ile, hesaplanan lower_bound değerinden büyük "streams" değerlerine sahip satırlar seçilir ve bu yeni alt kümeye df_filtered adı verilir. Son olarak, df_filtered.sort_values("streams", ascending = False) ile bu filtrelenmiş DataFrame, "streams" sütunu baz alınarak azalan sırada sıralanır. Bu işlem, veri setinden belirli bir eşik değerin altında kalan verileri çıkarmak ve kalan verileri popülerliklerine göre sıralamak için yararlıdır, böylece en popüler veriler kolaylıkla incelenebilir. Bu teknik, veri analizinde ve veri ön işlemede sıklıkla kullanılır, özellikle büyük veri setlerinde aykırı değerlerin veya istenmeyen verilerin filtrelenmesi gerektiğinde önem kazanır.

```
lower_bound = df["streams"].quantile(0.01)

df_filtered = df[df["streams"] > lower_bound]
df_filtered.sort_values("streams", ascending = False)
```

	artist(s)_name	artist_count	released_year	released_month	released_day	in_spotify_playlists	in_spotify_charts	streams	in_apple_playlists	in_apple_cl
179	Ed Sheeran	1	2017	1	6	32181	10	3562543890	33	
41	Post Malone, Swae Lee	2	2018	10	9	24094	78	2808096550	372	
162	Drake, WizKid, Kyla	3	2016	4	4	43257	24	2713922350	433	
84	Justin Bieber, The Kid Laroi	2	2021	7	9	17050	36	2665343922	492	
140	Imagine Dragons	1	2017	1	31	18986	23	2594040133	250	
...	...	...	...	...	...	...	...	...	...	
113	Taylor Swift	1	2023	7	7	99	15	36912123	21	
289	RM, Colde	2	2023	5	4	105	0	34502215	5	

Şekil2.15.: Veri Filtreleme ve Sıralama

2.2.16. Spotify Listelerindeki Sıralamalarına Göre Verileri Sıralama:

Bu kod satırı, veri setindeki girdileri "in_spotify_charts" sütununa göre azalan sırada sıralar. df.sort_values("in_spotify_charts", ascending=False) fonksiyonu kullanılarak, veri seti bu sütundaki değerlere göre büyükten küçüğe doğru sıralanır. Bu işlem, Spotify listelerinde en çok yer alan şarkıları veya içerikleri üstte göstermek için kullanılır, böylece en popüler olanlar kolaylıkla görülebilir. Bu tür bir sıralama, veri setinin analizinde, özellikle müzik endüstrisi veya popüler medya içeriği üzerine çalışmalarda, hangi şarkıların veya sanatçıların daha geniş bir kitleye ulaştığını anlamak için önemli bir adımdır. Bu işlem aynı zamanda, veri görselleştirmeleri ve istatistiksel analizler yapılırken veri setinin uygun bir şekilde hazırlanmasına da yardımcı olur.

```
df.sort_values("in_spotify_charts", ascending = False)
```

	artist(s)_name	artist_count	released_year	released_month	released_day	in_spotify_playlists	in_spotify_charts	streams	in_apple_playlists	in_apple_cl
0	Latto, Jung Kook	2	2023	7	14	553	147	141381703	43	
2	Olivia Rodrigo	1	2023	6	30	1397	113	140003974	94	
29	Dua Lipa	1	2023	5	25	2988	101	127408954	0	
3	Taylor Swift	1	2019	8	23	7858	100	800840817	116	
13	David Kushner	1	2023	4	14	3528	98	387570742	80	
...	...	...	...	...	...	...	...	...	...	
677	Em Beihold	1	2022	1	28	2026	0	258714692	47	
675	Chase Atlantic	1	2017	10	4	2742	0	498960285	5	

Şekil2.16.: Spotify Listelerindeki Sıralamalarına Göre Verileri Sıralama

2.2.17. Spotify Listelerindeki Giriş Sayılarının Frekans Dağılımı:

Bu kod satırı, "in_spotify_charts" sütunundaki değerlerin frekansını hesaplar ve bu değerlerin kaç kez tekrarlandığını gösterir. df['in_spotify_charts'].value_counts() fonksiyonu kullanılarak, her bir değerin DataFrame içinde kaç defa geçtiği sayılır. Bu, Spotify listelerinde bir şarkının veya sanatçının ne kadar sık yer aldığını anlamak için kullanılır. Elde edilen frekans dağılımı, bu sütundaki değerlerin yaygınlığını ve çeşitliliğini gösterir, bu da analizlerde önemli bir bilgi kaynağıdır. Örneğin, bazı şarkıların çok sık listelere girdiğini veya bazılarının nadiren yer aldığını bu yöntemle gözlemleyebiliriz. Bu bilgi, pazarlama stratejileri, popülerlik analizleri veya müzik endüstrisi trendlerini anlamak için değerli olabilir.

```
df['in_spotify_charts'].value_counts()
```

```
in_spotify_charts
0      347
4       41
2       36
6       31
8       16
      ...
79       1
66       1
41       1
62       1
37       1
Name: count, Length: 76, dtype: int64
```

Şekil2.17.: Giriş Sayılarının Frekans Dağılımı

2.2.18. Veri Seti İstatistiksel Özeti:

df.describe() fonksiyonu, DataFrame'deki sayısal sütunlar için istatistiksel bir özet sağlar. Bu işlem, veri setindeki sayısal değerlerin merkezi eğilimini (ortalama, medyan), yayılımını (standart sapma, minimum, maksimum değerler), ve çeyreklik değerlerini (ilk çeyrek ve üçüncü çeyrek) içerir. İstatistiksel özet tablosu, her bir sayısal sütun için bu değerleri listeleyerek veri setinin genel dağılımını ve temel özelliklerini gösterir.

```
df.describe()
```

	artist_count	released_year	released_month	released_day	in_spotify_playlists	in_spotify_charts	streams	in_apple_playlists	in_apple_charts	in_de
count	816.000000	816.000000	816.000000	816.000000	816.000000	816.000000	8.160000e+02	816.000000	816.000000	
mean	1.568627	2018.517157	6.024510	13.712010	4852.316176	11.736520	4.689858e+08	60.215686	49.534314	
std	0.876522	10.701973	3.570415	9.294719	7745.565488	18.624555	5.231267e+08	74.953565	49.570536	
min	1.000000	1930.000000	1.000000	1.000000	31.000000	0.000000	2.762000e+03	0.000000	0.000000	
25%	1.000000	2021.000000	3.000000	5.000000	829.000000	0.000000	1.342848e+08	12.000000	6.000000	
50%	1.000000	2022.000000	5.000000	13.000000	2037.500000	3.000000	2.638368e+08	32.000000	34.500000	
75%	2.000000	2022.000000	9.000000	22.000000	4890.750000	16.000000	6.011986e+08	78.250000	84.000000	
max	8.000000	2023.000000	12.000000	31.000000	52898.000000	147.000000	3.562544e+09	532.000000	275.000000	

Şekil2.18.: Veri Seti İstatistiksel Özeti

2.2.19. Korelasyon Matrisi Hesaplaması:

Bu kod bloğu, belirli sütunlar arasındaki istatistiksel ilişkileri incelemek için bir korelasyon matrisi oluşturur. df[...] içinde belirtilen sütunlar, veri setindeki çeşitli özellikleri (Spotify, Apple, Deezer listeleri ve çizelgeleri; şarkı özellikleri gibi bpm, anahtar, mod; ve şarkıların dans edilebilirlik, neşe, enerji, akustiklik gibi yüzdesel özellikler) temsil eder. .corr() fonksiyonu, Pearson korelasyon katsayılarını hesaplamak için kullanılır, bu da iki değişken arasındaki doğrusal ilişkinin gücünü ve yönünü ölçer.

method='pearson' parametresi, Pearson korelasyon metodunun kullanılmasını sağlar, bu metod iki sürekli değişken arasındaki doğrusal ilişkiyi değerlendirir. numeric_only=True parametresi, sadece sayısal sütunlar arasında korelasyon hesaplanmasını sağlar, böylece metinsel veya kategorik veriler bu hesaplamaya dahil edilmez.

Elde edilen corr_matrix değişkeni, her bir değişken çifti için korelasyon katsayılarını içeren bir kare matristir. Bu matris, değişkenlerin birbirleriyle olan ilişkilerini anlamada önemli bir araçtır. Örneğin, yüksek pozitif bir korelasyon, bir değişkenin değeri arttıkça diğer değişkenin de değerinin arttığını gösterirken; yüksek negatif bir korelasyon, bir değişkenin değeri arttıkça diğer değişkenin değerinin azaldığını gösterir. Bu bilgiler, veri setindeki özellikler arasındaki ilişkileri anlamak ve bu özelliklerin nasıl birlikte hareket ettiğini görmek için değerlidir.

```
corr_matrix = df[["in_spotify_playlists",
                  "in_spotify_charts",
                  "streams",
                  "in_apple_playlists",
                  "in_apple_charts",
                  "in_deezer_playlists",
                  "in_deezer_charts",
                  "in_shazam_charts",
                  "bpm",
                  "key",
                  "mode",
                  "danceability_%",
                  "valence_%",
                  "energy_%",
                  "acousticness_%",
                  "instrumentalness_%",
                  "liveness_%",
                  "speechiness_%"]].corr(method='pearson', numeric_only=True)
corr_matrix
```

	in_spotify_playlists	in_spotify_charts	streams	in_apple_playlists	in_apple_charts	in_deezer_playlists	in_deezer_charts	in_shazam_charts
in_spotify_playlists	1.000000	0.141186	0.780404	0.688306	0.215504	0.829629	0.101158	0.082094
in_spotify_charts	0.141186	1.000000	0.214034	0.207841	0.556509	0.077715	0.566020	0.573956
streams	0.780404	0.214034	1.000000	0.735321	0.269137	0.610812	0.184329	0.015715
in_apple_playlists	0.688306	0.207841	0.735321	1.000000	0.364339	0.468576	0.326598	0.138757
in_apple_charts	0.215504	0.556509	0.269137	0.364339	1.000000	0.136927	0.366295	0.419483
in_deezer_playlists	0.829629	0.077715	0.610812	0.468576	0.136927	1.000000	0.039101	0.084672
in_deezer_charts	0.101158	0.566020	0.184329	0.326598	0.366295	0.039101	1.000000	0.389482
in_shazam_charts	0.082094	0.573956	0.015715	0.138757	0.419483	0.084672	0.389482	1.000000
bpm	-0.034628	0.028496	-0.025694	0.005237	0.026531	-0.038320	0.021487	0.057052
danceability_%	-0.097348	0.050623	-0.093268	-0.013306	-0.027371	-0.070113	0.080926	-0.006668
valence_%	-0.029527	0.050862	-0.051014	0.042017	0.044229	-0.008299	0.073786	-0.017054
energy_%	0.035968	0.105217	-0.036499	0.040211	0.129793	0.081936	0.104529	0.078289
acousticness_%	-0.064896	-0.073482	-0.005751	-0.071554	-0.097803	-0.075443	-0.035575	-0.021482
instrumentalness_%	-0.024631	-0.005963	-0.033039	-0.054524	-0.000615	-0.012126	0.003396	0.054561
liveness_%	-0.052002	-0.026655	-0.056664	-0.065062	-0.007271	-0.024304	0.003602	-0.045118
speechiness_%	-0.077870	-0.094726	-0.099968	-0.098233	-0.147513	-0.057164	-0.081971	-0.071566

Şekil2.19.: Korelasyon Matrisi Hesaplaması

3.Data Visualization

3.1. Korelasyon Matrisinin Görselleştirilmesi:

Bu kod bloğu, daha önce hesaplanan korelasyon matrisini bir ısı haritası (heatmap) olarak görselleştirir, böylece değişkenler arasındaki ilişkileri görsel olarak incelemek mümkün olur. İlk olarak, plt.subplots(figsize=(13, 10)) ile 13x10 boyutlarında bir grafik alanı oluşturulur. Ardından, sns.heatmap() fonksiyonu kullanılarak korelasyon matrisi çizilir.

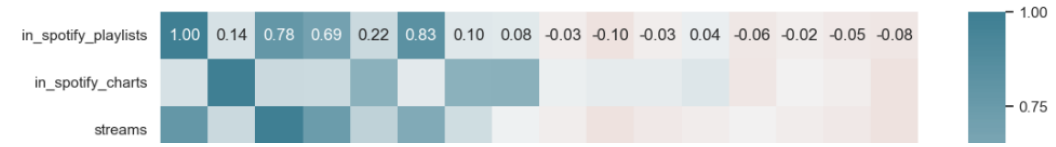
vmin=-1, vmax=1, center=0 parametreleri, ısı haritasının renk skalasının -1 ile +1 arasında olmasını ve merkezinin 0'da olmasını sağlar, bu değerler Pearson korelasyon katsayılarının olası aralığını temsil eder. cmap=sns.diverging_palette(20, 220, n=200) ile, renk paleti olarak iki ucu kapsayan bir gradyan seçilir; burada mavi düşük (negatif) değerler için, kırmızı ise yüksek (pozitif) değerler için kullanılır.

square=True parametresi, her bir hücrenin kare şeklinde olmasını sağlar. fmt='0.2f', annot=True ile her hücreye korelasyon değerleri iki ondalık basamak hassasiyetinde yazdırılır ve hücrelere etiket olarak eklenir.

Son olarak, ax.set_xticklabels() fonksiyonu ile x eksenindeki etiketlerin 45 derece döndürülerek sağa doğru hizalanmasını sağlar; bu, etiketlerin okunmasını kolaylaştırır ve grafikteki tıkanıklığı önler.

plt.show() ile grafik gösterilir. Bu görselleştirme, veri setindeki değişkenler arasındaki ilişkileri hızlıca gözden geçirmek ve potansiyel olarak önemli ilişkileri tanımak için etkili bir yoldur. Özellikle büyük veri setlerinde, hangi özelliklerin birbiriyle güçlü bir ilişki içinde olduğunu anlamak için kullanılır.

```
fig, ax = plt.subplots(figsize=(13, 10))


ax = sns.heatmap(
    corr_matrix,
    vmin=-1, vmax=1, center=0,
    cmap=sns.diverging_palette(20, 220, n=200),
    square=True,
    fmt = '0.2f',
    annot=True
)


ax.set_xticklabels(
    ax.get_xticklabels(),
    rotation=45,
    horizontalalignment='right'
)


plt.show()
```



Şekil3.1.: Korelasyon Matrisinin Görselleştirilmesi

3.2. Veri Setindeki Değişkenler İçin Histogram Görselleştirmesi:

Bu kod bloğu, veri setindeki belirli değişkenler için histogramlar çizerek, bu değişkenlerin dağılımlarını görselleştirir. İlk olarak, import math ve import matplotlib.pyplot as plt ile gerekli kütüphaneler içe aktarılır. Sonrasında, incelenecek sütunlar ve bu sütunlar için kullanılacak renkler birer liste halinde tanımlanır.

Histogramlar için gerekli satır ve sütun sayısı, len(sütunlar_grafiği) ile sütun sayısını alarak ve math.ceil() fonksiyonu kullanarak hesaplanır. Bu, toplam grafik sayısının yarısına eşit ya da bir fazla satır olacak şekilde ayarlanmasını sağlar.

plt.subplots() ile belirlenen satır ve sütun sayısına göre bir grafik ızgarası oluşturulur ve her bir grafik için boyut (16, 16) olarak ayarlanır. Döngü (for) içerisinde, her bir değişken için histogramlar çizilir. Her histogram, ilgili değişkenin sıklık dağılımını gösterir ve bu sıklık, belirlenen renklerle ifade edilir.

Her bir grafik için x ekseni etiketleri, y ekseni etiketleri ve grafik başlıkları ayarlanarak, hangi değişkenin dağılımını temsil ettiği açıkça belirtilir. Histogramların daha anlaşılır olması için fontsize ve weight parametreleri ile etiket ve başlık özellikleri belirlenir.

Son olarak, eğer tüm alt grafikler kullanılmıyorsa, kullanılmayan grafik alanları fig.delaxes() ile kaldırılır. plt.tight_layout() ile grafikler arası doğru mesafe ayarlanarak görsel çakışmalar önlenir ve plt.show() ile tüm grafikler gösterilir.

Bu tür bir görselleştirme, veri setindeki değişkenlerin dağılımını ve sıklık bilgilerini detaylı bir şekilde incelemek için önemlidir. Her bir histogram, veri setindeki ilgili değişkenin ne kadar yaygın olduğunu veya belirli bir değere sahip örneklerin sayısını görsel olarak sergiler.

```
import math
import matplotlib.pyplot as plt

sütunlar_grafiği = ['streams', 'in_spotify_playlists', 'in_apple_playlists', 'in_deezer_playlists', 'in_spotify_charts', 'in_app
renkler = ["#1FD662", "#FA354D", "#6F0375", "#0088FF", "#1FD662", "#FA354D", "#6F0375", "#0088FF"]

# Alt grafikler için gerekli satır ve sütun sayısını hesaplayın
sütun_sayısı = len(sütunlar_grafiği)
satır_sayısı = math.ceil(sütun_sayısı / 2)

# Hesaplanan satır ve sütun sayısına göre alt grafiklerin ızgarasını oluşturun
fig, eksenler = plt.subplots(satır_sayısı, 2, figsize=(16, 16))

for i, sütun in enumerate(sütunlar_grafiği):
    # Döngü indeksine göre geçerli alt grafiği seçin
    eksen = eksenler[i // 2, i % 2]

    # Geçerli sütun için bir histogram çizin
    eksen.hist(df[sütun], color=renkler[i], bins=20)  # Gerektiğinde bin sayısını ayarlayın

    # x ve y etiketlerini ekleyin
    eksen.set_xlabel(f"{sütun} Sayısı", fontsize=14)
    eksen.set_ylabel("Frekans")

    # Başlık ekleyin
    eksen.set_title(f"Şarkı Sayısına Göre {sütun} Sayısı", fontsize=14, weight="bold")

# Izgaradaki kullanılmayan alt grafikleri kaldırın
for i in range(sütun_sayısı, satır_sayısı * 2):
    fig.delaxes(eksenler.flatten()[i])

plt.tight_layout()
plt.show()
```

Şekil3.2.: Veri Setindeki Değişkenler İçin Histogram Görselleştirmesi

3.3. Müziksel Özelliklerin Histogramları ile Görselleştirme:

Bu kod bloğu, şarkıların çeşitli müziksel özelliklerine dair dağılımları görselleştirmek için histogramlar çizer. Özellikler arasında "bpm" (beats per minute), "danceability_%", "valence_%", "energy_%", "acousticness_%", "instrumentalness_%", "liveness_%", ve "speechiness_%" bulunur. Her bir özellik için ayrı bir histogram oluşturulur.

İlk olarak, import math ve import matplotlib.pyplot as plt ile gerekli kütüphaneler yüklenir. Sonrasında, incelenecek sütunlar ve bu sütunlar için kullanılacak renkler birer liste halinde belirtilir.

Grafiklerin düzenlenmesi için len(sütunlar_grafiği) ile sütun sayısı alınır ve math.ceil(sütun_sayısı / 2) ile kaç satır olması gerektiği hesaplanır. plt.subplots() fonksiyonu ile belirtilen satır ve sütun sayısına uygun alt grafikler oluşturulur ve bu grafiklerin boyutları (16, 16) olarak ayarlanır.

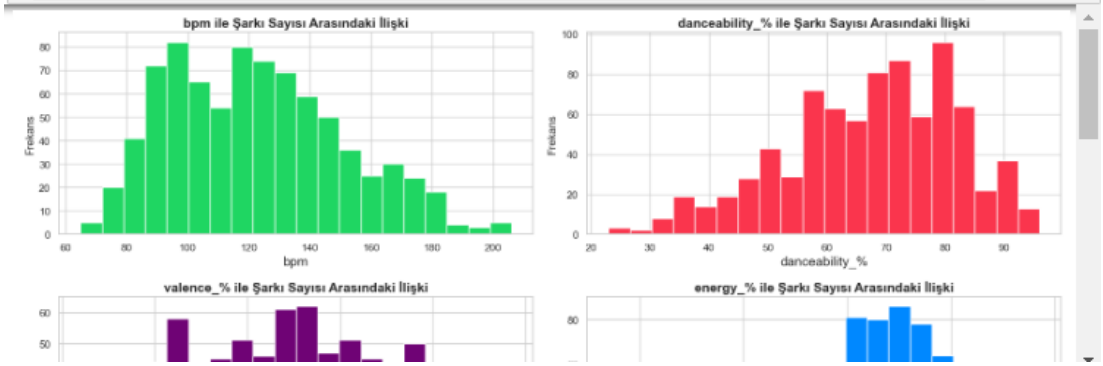
Döngü (for) içerisinde her bir değişken için histogramlar çizilir. eksen.hist() metodu ile belirtilen sütunun değerlerine göre histogram oluşturulur, burada "color" parametresiyle her bir histograma özel renk atanır ve "bins=20" ile histogramın bölümlendirilmesi sağlanır.

Her grafik için x ve y eksenlerine etiketler eklenir ve etiketlerin font büyüklüğü ve ağırlığı ayarlanır. Ayrıca, her bir grafik için anlaşılır başlıklar eklenir.

Eğer oluşturulan grafik matrisinde kullanılmayan grafik alanları varsa, fig.delaxes() ile bu alanlar kaldırılır. Son olarak, plt.tight_layout() ile grafikler arasındaki boşluklar düzenlenir ve plt.show() ile grafikler gösterilir.

Bu görselleştirme, şarkıların müziksel özelliklerinin dağılımlarını detaylı bir şekilde incelemek ve bu özelliklerin veri seti genelinde nasıl varyasyonlar gösterdiğini görmek için kullanılır. Her bir histogram, ilgili özelliğin frekansını ve yayılımını görsel olarak sergileyerek veri setindeki müziksel eğilimleri anlamaya yardımcı olur.

```python
import math
import matplotlib.pyplot as plt

sütunlar_grafiği = ['bpm', 'danceability_%', 'valence_%', 'energy_%', 'acousticness_%', 'instrumentalness_%', 'liveness_%', 'spe
renkler = ["#1FD662", "#FA354D", "#6F0375", "#0088FF", "#1FD662", "#FA354D", "#6F0375", "#0088FF"]

# Alt grafikler için gerekli satır ve sütun sayısını hesaplayın
sütun_sayısı = len(sütunlar_grafiği)
satır_sayısı = math.ceil(sütun_sayısı / 2)

# Hesaplanan satır ve sütun sayısına göre alt grafiklerin ızgarasını oluşturun
fig, eksenler = plt.subplots(satır_sayısı, 2, figsize=(16, 16))

for i, sütun in enumerate(sütunlar_grafiği):
    # Döngü indeksine göre geçerli alt grafiği seçin
    eksen = eksenler[i // 2, i % 2]

    # Geçerli sütun için bir histogram çizin
    eksen.hist(df[sütun], color=renkler[i], bins=20)  # Gerektiğinde bin sayısını ayarlayın

    # x ve y etiketlerini ekleyin
    eksen.set_xlabel(f"{sütun}", fontsize=14)
    eksen.set_ylabel("Frekans")

    # Başlık ekleyin
    eksen.set_title(f"{sütun} ile Şarkı Sayısı Arasındaki İlişki", fontsize=14, weight="bold")

# Izgaradaki kullanılmayan alt grafikleri kaldırın
for i in range(sütun_sayısı, satır_sayısı * 2):
    fig.delaxes(eksenler.flatten()[i])

plt.tight_layout()
plt.show()
```



Şekil3.3.: Müziksel Özelliklerin Histogramları ile Görselleştirme

3.4. BPM ve Dans Edilebilirlik Arasındaki İlişkinin Görselleştirilmesi:

Bu kod bloğu, şarkıların "bpm" (beats per minute, dakikadaki vuruş sayısı) ile "danceability_%" (dans edilebilirlik yüzdesi) özellikleri arasındaki ilişkiyi bir saçılma grafiği (scatter plot) ve bir polinom uyum çizgisi ile görselleştirir. İlk olarak, plt.scatter() fonksiyonu ile "bpm" değerleri x ekseninde ve "danceability_%" değerleri y ekseninde olacak şekilde bir saçılma grafiği çizilir. Bu grafikteki noktalar, "#FFA07A" (açık mercan rengi) ile renklendirilir, bu da her bir şarkının bu iki özellik arasındaki ilişkiyi görsel olarak ifade eder.

Daha sonra, np.polyfit() fonksiyonu ile "bpm" ve "danceability_%" arasındaki ilişkiyi en iyi ifade eden ikinci dereceden (derece 2) bir polinom uyum çizgisi

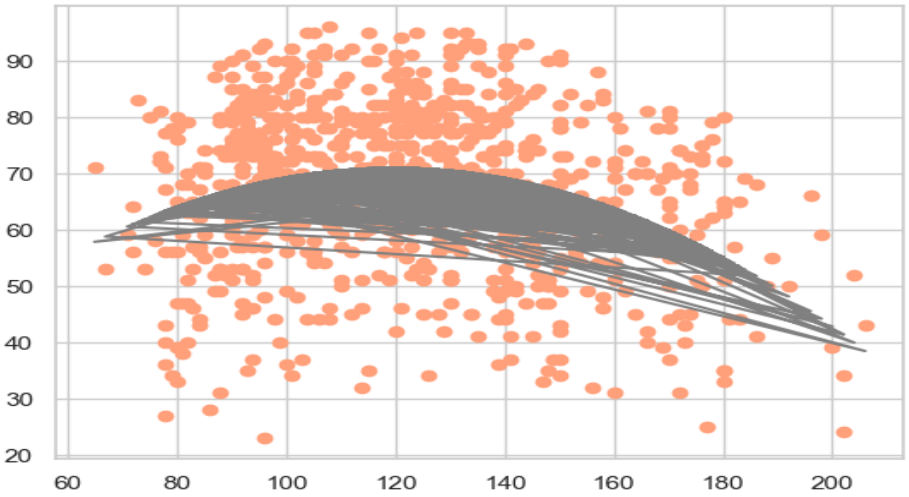
hesaplanır. Bu fonksiyon, verilen x ve y değerleri arasında en küçük kareler yöntemiyle bir uyum sağlar ve bu uyumun katsayılarını döndürür. np.poly1d() fonksiyonu bu katsayıları alarak bir polinom fonksiyonu oluşturur.

plt.plot() fonksiyonu kullanılarak, hesaplanan polinom fonksiyonundan elde edilen değerlerle, yine "bpm" değerlerine karşılık gelen dans edilebilirlik değerleri grafiğe çizilir. Bu uyum çizgisi "gray" (gri) renkle belirtilir.

Bu görselleştirme, "bpm" ve "danceability_%" arasındaki ilişkinin ne kadar güçlü olduğunu ve bu iki değişkenin nasıl bir eğilim gösterdiğini anlamak için yararlıdır. Saçılma grafiği ile verilerin dağılımı ve polinom çizgisi ile bu verilerin genel eğilimi arasındaki ilişki kolaylıkla gözlemlenebilir.

```
plt.scatter(df["bpm"], df["danceability_%"], color = "#FFA07A")
z = np.polyfit(df["bpm"], df["danceability_%"], 2)
p = np.poly1d(z)
plt.plot(df["bpm"], p(df["bpm"]), "gray")
```

```
[<matplotlib.lines.Line2D at 0x168b7cdf910>]
```



Şekil3.4.: BPM ve Dans Edilebilirlik Arasındaki İlişkinin Görselleştirilmesi

3.5. Yayın Yılına Göre Zaman Aralıklarının Oluşturulması:

Bu kod bloğu, veri setindeki şarkıların yayın yılı bilgisini kullanarak yeni zaman aralıkları sütunları oluşturur. İlk olarak, df['YearRange'] sütunu, df['released_year'] sütununun aynı değerleri ile oluşturulur. Bu, yayın yılı bilgisinin başka bir adla saklanması için yapılır ve genellikle veri manipülasyonu veya analiz süreçlerinde başvuru kolaylığı sağlamak amacıyla kullanılır.

Daha sonra, df['5YearRange'] ve df['10YearRange'] sütunları oluşturulur. Bu sütunlar, yayın yılını 5 ve 10 yıllık zaman aralıklarına gruplamak için kullanılır. Bu gruplama, // operatörü ile tamsayı bölmesi yaparak ve sonucu 5 veya 10 ile çarparak gerçekleştirilir. Örneğin, bir şarkının yayın yılı 1997 ise, 1997 // 5 * 5 işlemi sonucunda 1995 değeri 5YearRange sütununa, 1997 // 10 * 10 işlemi sonucunda ise 1990 değeri 10YearRange sütununa atanır.

Bu tür zaman aralığı sütunları, veri setinde zamanla ilgili trendleri incelemek için kullanışlıdır. Özellikle, müzik endüstrisindeki değişimler veya belirli dönemlerde popüler olan müzik türlerinin analizi gibi konularda faydalı olabilir. Zaman aralıkları, verileri daha geniş zaman dilimlerine göre gruplandırarak, uzun vadeli trendleri ve değişiklikleri daha net gözlemleme imkanı tanır. Bu, tarihî veriler üzerinden geniş çapta analizler yapılmasına olanak sağlar ve stratejik planlamalar için veri tabanlı kararlar alınmasına yardımcı olur.

```
df['YearRange'] = df['released_year']
df['5YearRange'] = df['released_year'] // 5 * 5
df['10YearRange'] = df['released_year'] // 10 * 10
```

Şekil3.5.: Yayın Yılına Göre Zaman Aralıklarının Oluşturulması:

3.6. Yıl Aralıklarına Göre Toplam Akış Sayılarının Görselleştirilmesi:

Bu kod bloğu, veri setindeki şarkıların her bir yıl aralığı için toplam akış sayılarını görselleştirir. İlk olarak, df.groupby('YearRange')['streams'].sum().reset_index() ifadesiyle, şarkıların yayın yıllarına göre gruplandırılıp, her bir grup için streams sütunundaki değerlerin toplamı hesaplanır ve bu veri yearly_stream_counts DataFrame'ine atanır.

plt.figure(figsize=(30, 6)) ile grafik boyutu belirlenir, genişliği 30 inç ve yüksekliği 6 inç olarak ayarlanır. plt.bar() fonksiyonu kullanılarak, yıl aralıklarına göre toplam akış sayıları bir çubuk grafikle gösterilir. Çubukların rengi "lightcoral" olarak belirlenir ve çubuk genişliği 1 birim olarak ayarlanır, bu da çubukların birbirine bitişik olmasını sağlar.

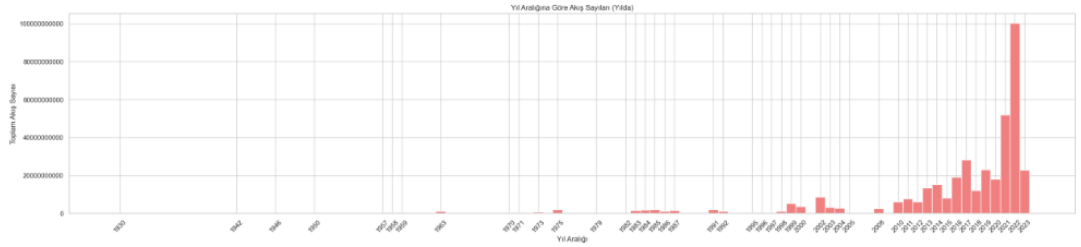
Grafik başlığı, x ve y eksenlerine etiketler eklenir. Başlık "Yıl Aralığına Göre Akış Sayıları (Yılda)" olarak belirlenirken, x ekseni etiketi "Yıl Aralığı", y ekseni etiketi

ise "Toplam Akış Sayısı" olarak belirlenir. X eksenindeki etiketler, yıl aralıklarını temsil eder ve 45 derece döndürülerek yerleştirilir, bu da okunabilirliği artırır.

plt.ticklabel_format(style='plain', axis='y') ile y eksenindeki etiketlerin bilimsel gösterim yerine düz metin olarak gösterilmesi sağlanır. plt.grid(True) ile arka plana ızgara çizgileri eklenir, bu da grafik üzerindeki verilerin yorumlanmasını kolaylaştırır.

plt.show() ile çubuk grafik gösterilir. Bu görselleştirme, yıllar boyunca şarkıların toplam akış sayılarının nasıl değiştiğini anlamak için yararlıdır. Trendleri, pik noktaları ve dönemsel değişimleri gözlemlemek açısından özellikle faydalıdır ve müzik endüstrisinin zaman içindeki performansını değerlendirmek için kullanılabilir.

```
yearly_stream_counts= df.groupby('YearRange')['streams'].sum().reset_index()
plt.figure(figsize=(30, 6))
plt.bar(yearly_stream_counts['YearRange'], yearly_stream_counts['streams'], width=1, align='center', color = "lightcoral")
plt.title('Yıl Aralığına Göre Akış Sayıları (Yılda)')
plt.xlabel('Yıl Aralığı')
plt.ylabel('Toplam Akış Sayısı')


plt.xticks(yearly_stream_counts['YearRange'], [f'{yr}' for yr in yearly_stream_counts['YearRange']], rotation=45)
plt.ticklabel_format(style='plain', axis='y')

plt.grid(True)
plt.show()
```



Şekil3.6.: Yıl Aralıklarına Göre Toplam Akış Sayılarının Görselleştirilmesi

3.7. Yıl Aralıklarına Göre Akış Sayılarının Zaman Serisi Grafiği:

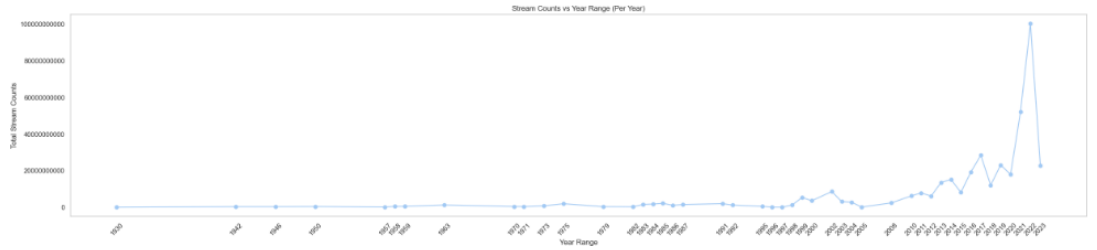
Bu kod bloğu, yıl aralıklarına göre toplam akış sayılarını zaman serisi grafiği olarak görselleştirir. plt.figure(figsize=(30, 6)) ile, 30 inç genişlik ve 6 inç yükseklik boyutlarında bir grafik alanı oluşturulur. plt.plot() fonksiyonu kullanılarak, yıl aralıkları ve ilgili toplam akış sayıları birleştirilir. Grafikte, her bir data noktası "o" şeklinde bir işaretçiyle belirtilir ve işaretçiler arasında düz bir çizgiyle bağlantı sağlanır.

Grafik başlığı "Stream Counts vs Year Range (Per Year)" olarak ayarlanır, x ekseni "Year Range" ve y ekseni "Total Stream Counts" olarak etiketlenir. X eksenindeki etiketler yıl aralıklarını gösterir ve 45 derece döndürülerek konumlandırılır, bu sayede etiketler daha kolay okunabilir hale getirilir.

plt.ticklabel_format(style='plain', axis='y') ile y eksenindeki sayıların bilimsel gösterim yerine düz format kullanılması sağlanır. plt.grid(False) ile arka plan ızgarası kapatılır, böylece grafik daha temiz bir görünüme kavuşur.

plt.show() ile oluşturulan zaman serisi grafiği gösterilir. Bu tür bir görselleştirme, yıllar boyunca akış sayılarının nasıl değiştiğini, trendleri ve potansiyel dönemsel artış veya azalmaları gözlemlemek için etkili bir yöntemdir. Veri analistleri bu grafikten, müzik endüstrisindeki toplam akış sayılarının zaman içindeki performansını ve dinamiklerini anlamak için faydalanabilirler.

```
plt.figure(figsize=(30, 6))
plt.plot(yearly_stream_counts['YearRange'], yearly_stream_counts['streams'], marker='o', linestyle='-')
plt.title('Stream Counts vs Year Range (Per Year)')
plt.xlabel('Year Range')
plt.ylabel('Total Stream Counts')

# Customize the x-axis labels
plt.xticks(yearly_stream_counts['YearRange'], [f'{yr}' for yr in yearly_stream_counts['YearRange']], rotation=45)
plt.ticklabel_format(style='plain', axis='y')

plt.grid(False)
plt.show()
```



Şekil3.7.: Yıl Aralıklarına Göre Akış Sayılarının Zaman Serisi Grafiği

3.8. Beş Yıllık Aralıklara Göre Akış Sayılarının Çubuk Grafiği

Bu kod bloğu, beş yıllık aralıklar boyunca toplam akış sayılarını gösteren bir çubuk grafiği oluşturur. df.groupby('5YearRange')['streams'].sum().reset_index() ifadesi ile, veri seti "5YearRange" sütununa göre gruplandırılır ve her grup için streams sütunundaki değerler toplanır. Elde edilen sonuç bidecadely_stream_counts DataFrame'ine atanır.

plt.figure(figsize=(10, 6)) ile grafik boyutu 10 inç genişlik ve 6 inç yükseklik olarak belirlenir. plt.bar() fonksiyonu kullanılarak, her bir beş yıllık aralık için toplam akış sayıları çubuk grafik şeklinde çizilir. Çubuklar, "lightblue" (açık mavi) renkle boyanır ve her bir çubuğun genişliği 4 birim olarak ayarlanır, bu sayede grafikteki çubuklar arasında boşluklar minimuma indirgenir.

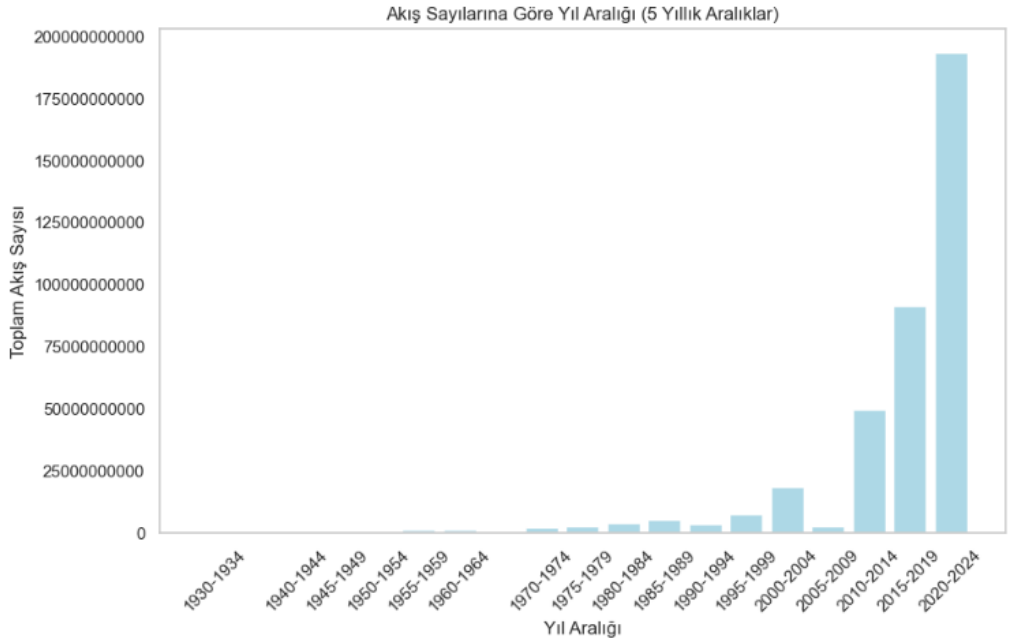
Grafik başlığı "Akış Sayılarına Göre Yıl Aralığı (5 Yıllık Aralıklar)" olarak belirlenir, x ekseni "Yıl Aralığı" ve y ekseni "Toplam Akış Sayısı" olarak etiketlenir. X eksenindeki etiketler, her beş yıllık aralığı temsil eder ve {yr}-{yr+4} formatında gösterilir, yani her aralığın başlangıç ve bitiş yıllarını içerir. Etiketler 45 derece

döndürülerek yerleştirilir, bu sayede daha fazla bilgi içeren etiketler daha kolay okunabilir hale gelir.

plt.ticklabel_format(style='plain', axis='y') ile y ekseni üzerindeki sayıların bilimsel gösterim yerine düz format kullanılması sağlanır. plt.grid(False) ile arka plan ızgarası kapatılır, bu da grafik üzerindeki verilerin daha net görülmesini sağlar.

plt.show() ile çubuk grafik gösterilir. Bu görselleştirme, beş yıllık periyotlar halinde müzik akışlarının nasıl değiştiğini analiz etmek için kullanılır.

```
bidecadely_stream_counts = df.groupby('5YearRange')['streams'].sum().reset_index()
plt.figure(figsize=(10, 6))
plt.bar(bidecadely_stream_counts['5YearRange'], bidecadely_stream_counts['streams'], width=4, align='center', color ="lightblue"
plt.title('Akış Sayılarına Göre Yıl Aralığı (5 Yıllık Aralıklar)')
plt.xlabel('Yıl Aralığı')
plt.ylabel('Toplam Akış Sayısı')

plt.xticks(bidecadely_stream_counts['5YearRange'], [f'{yr}-{yr+4}' for yr in bidecadely_stream_counts['5YearRange']], rotation=4
plt.ticklabel_format(style='plain', axis='y')

plt.grid(False)
plt.show()
```



Şekil3.8.: Beş Yıllık Aralıklara Göre Akış Sayılarının Çubuk Grafiği

3.9. On Yıllık Aralıklara Göre Akış Sayılarının Çubuk Grafiği:

Bu kod bloğu, on yıllık aralıklar boyunca toplam akış sayılarını gösteren bir çubuk grafiği oluşturur. İlk olarak, df.groupby('10YearRange')['streams'].sum().reset_index() işlemi ile veri seti "10YearRange" sütununa göre gruplandırılır ve her grup için streams sütunundaki değerler toplanarak decadely_stream_counts DataFrame'ine aktarılır.

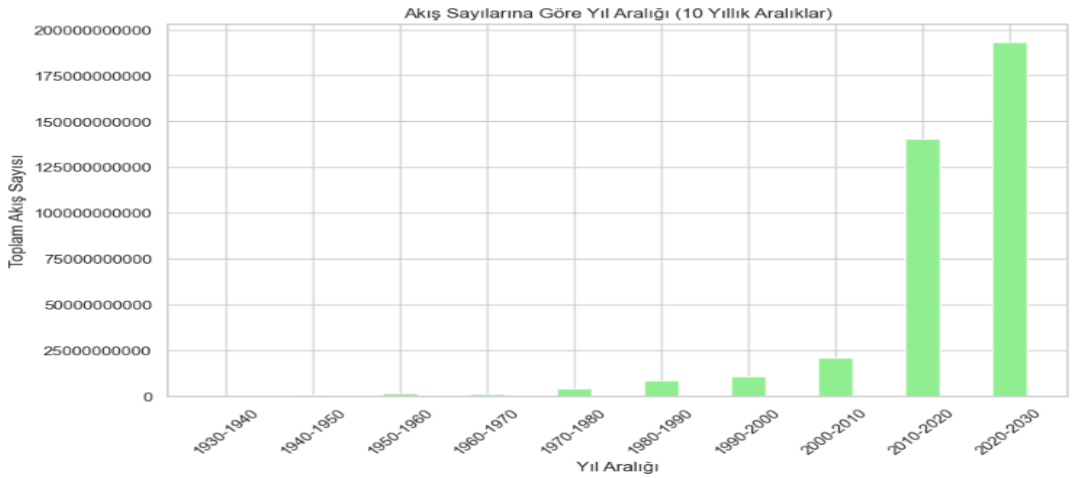
plt.figure(figsize=(10, 6)) ile grafik boyutu 10 inç genişlik ve 6 inç yükseklik olarak ayarlanır. plt.bar() fonksiyonu kullanılarak, her bir on yıllık aralık için toplam akış sayıları çubuk grafik şeklinde çizilir. Çubuklar, "lightgreen" (açık yeşil) renkle boyanır ve her bir çubuğun genişliği 4 birim olarak ayarlanır. Bu genişlik, çubuklar arasındaki boşlukları azaltır ve grafikteki bilgilerin daha yoğun bir şekilde sunulmasını sağlar.

Grafik başlığı "Akış Sayılarına Göre Yıl Aralığı (10 Yıllık Aralıklar)" olarak belirlenir, x ekseni "Yıl Aralığı" ve y ekseni "Toplam Akış Sayısı" olarak etiketlenir. X eksenindeki etiketler, her on yıllık aralığı temsil eder ve {yr}-{yr+10} formatında gösterilir, bu da her aralığın başlangıç yılından sonraki onuncu yıla kadar olan süreci kapsar. Etiketler 45 derece döndürülerek konumlandırılır, böylece uzun etiketler daha kolay okunabilir.

plt.ticklabel_format(style='plain', axis='y') ile y eksenindeki sayıların bilimsel gösterim yerine düz format kullanılması sağlanır. plt.grid(True) ile arka plana ızgara çizgileri eklenir, bu da grafik üzerindeki verilerin analizini kolaylaştırır.

plt.show() ile çubuk grafik gösterilir. Bu görselleştirme, on yıllık periyotlar halinde müzik akışlarının nasıl değiştiğini analiz etmek için kullanılır ve müzik endüstrisindeki uzun vadeli trendleri ve değişimleri gözlemlemek için etkili bir yöntem sunar.

```
decadely_stream_counts = df.groupby('10YearRange')['streams'].sum().reset_index()
plt.figure(figsize=(10, 6))
plt.bar(decadely_stream_counts['10YearRange'], decadely_stream_counts['streams'], width=4, align='center', color ="lightgreen")
plt.title('Akış Sayılarına Göre Yıl Aralığı (10 Yıllık Aralıklar)')
plt.xlabel('Yıl Aralığı')
plt.ylabel('Toplam Akış Sayısı')


plt.xticks(decadely_stream_counts['10YearRange'], [f'{yr}-{yr+10}' for yr in decadely_stream_counts['10YearRange']], rotation=45)
plt.ticklabel_format(style='plain', axis='y')

plt.grid(True)
plt.show()
```



Şekil3.9.: On Yıllık Aralıklara Göre Akış Sayılarının Çubuk Grafiği

3.10. Müzikal Anahtarlara Göre Akış Sayılarının Kutu Grafiği Görselleştirmesi:

Bu kod bloğu, şarkıların müzikal anahtarlara (key) göre akış sayılarını (streams) kutu grafiği (boxplot) ile görselleştirir. Kutu grafiği, veri dağılımının merkezi eğilimini, çeyrekliklerini ve aykırı değerlerini göstererek, farklı anahtarların akış performansını değerlendirme imkanı sunar.

İlk olarak, sns.color_palette("Set3", len(df['key'].unique())) fonksiyonu ile, df['key'].unique() ile belirlenen benzersiz anahtar sayısı kadar renkten oluşan bir renk paleti (palette) oluşturulur. "Set3" paleti, görsel çekicilik sağlamak ve kutu grafiklerini renklendirmek için kullanılır.

sorted_keys = sorted(df['key'].unique()) ifadesiyle, müzikal anahtarlar sıralanır. Bu sıralama, kutu grafiklerinin x ekseninde düzenli bir şekilde görünmesini sağlar. plt.figure(figsize=(12, 6)) ile grafik boyutu belirlenir.

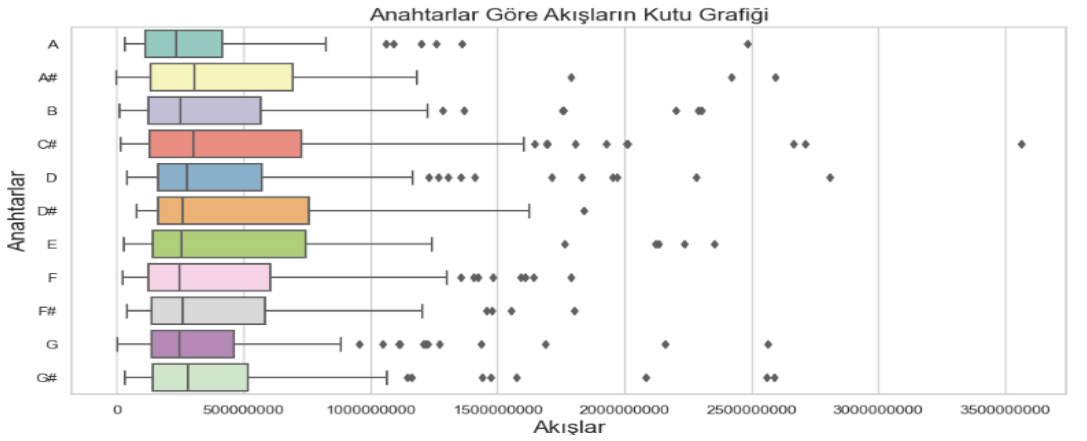
sns.boxplot() fonksiyonu kullanılarak kutu grafiği çizilir. Y ekseni müzikal anahtarları (key), x ekseni ise akış sayılarını (streams) temsil eder. order=sorted_keys ile kutu grafiklerinin sıralı anahtarlar bazında çizilmesi sağlanır ve palette=palette ile her bir kutu grafiği için farklı bir renk kullanılır.

X ve y ekseni etiketleri ve grafik başlığı plt.xlabel(), plt.ylabel() ve plt.title() fonksiyonları ile belirlenir. Bu etiketler ve başlık, grafik üzerinde neyin gösterildiğini açıklar ve okuyucunun grafikten alacağı bilgiyi kolaylaştırır.

plt.ticklabel_format(style='plain', axis='x') ile x ekseni üzerindeki sayıların bilimsel gösterim yerine düz format kullanılması sağlanır, bu da büyük sayıların daha kolay okunmasını sağlar.

plt.show() ile kutu grafiği gösterilir. Bu görselleştirme, müzikal anahtarların akış sayıları üzerindeki potansiyel etkisini anlamak için yararlıdır ve analistlere veya müzik yapımcılarına, belirli anahtarların popülerliği ve dinleyici davranışları üzerindeki etkileri hakkında içgörüler sunar.

```
palette = sns.color_palette("Set3", len(df['key'].unique()))

sorted_keys = sorted(df['key'].unique())
plt.figure(figsize=(12, 6))
sns.boxplot(y="key", x="streams", data=df, order=sorted_keys, palette=palette)
plt.xlabel("Akışlar", fontsize=16)
plt.ylabel("Anahtarlar", fontsize=16)
plt.title("Anahtarlar Göre Akışların Kutu Grafiği", fontsize=16)

plt.ticklabel_format(style='plain', axis='x')
plt.show()
```



Şekil3.10.: Müzikal Anahtarlara Göre Akış Sayılarının Kutu Grafiği Görselleştirmesi

3.11. Müzikal Anahtarların Frekans Dağılımı Görselleştirmesi:

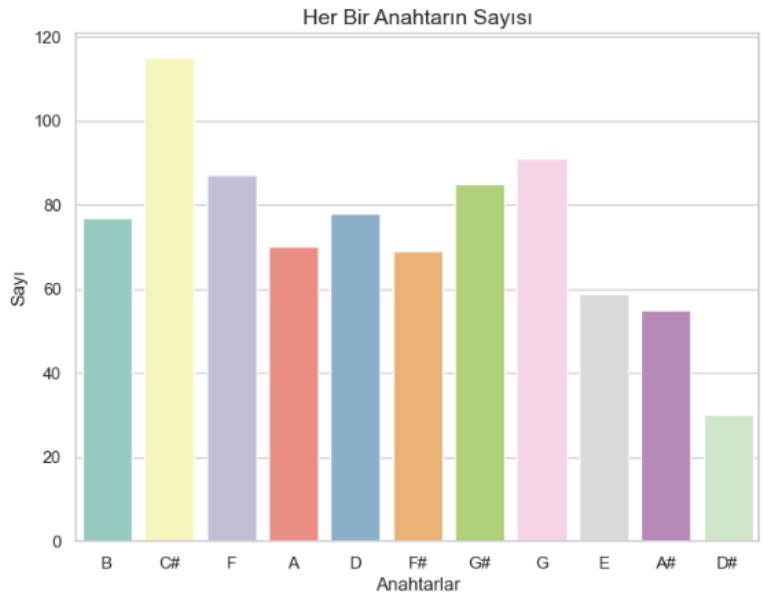
Bu kod bloğu, veri setindeki her bir müzikal anahtarın frekansını bir çubuk grafik aracılığıyla gösterir. Bu görselleştirme, hangi anahtarların veri setinde daha sık veya daha seyrek kullanıldığını açıkça gösterir, böylece müzikal tercihler ve trendler hakkında fikir edinilmesine yardımcı olur.

plt.figure(figsize=(8, 6)) ifadesiyle grafik için bir çerçeve belirlenir; burada genişlik 8 inç ve yükseklik 6 inç olarak ayarlanmıştır. sns.countplot() fonksiyonu kullanılarak, x ekseni "key" sütunu olarak belirlenmiş ve her bir anahtarın veri setinde kaç kez geçtiği sayılır. palette="Set3" parametresi ile grafikte kullanılacak renkler belirlenir, "Set3" paleti bu tür veriler için çeşitlilik ve görsel çekicilik sunar.

X ve y ekseni etiketleri (plt.xlabel() ve plt.ylabel()) ile "Anahtarlar" ve "Sayı" olarak ayarlanır. Bu etiketler, grafikte neyin temsil edildiğini belirtir ve grafikten alınacak bilgiyi kolaylaştırır. Grafik başlığı plt.title() ile "Her Bir Anahtarın Sayısı" olarak belirlenir ve bu, grafikte sunulan bilginin özünü yansıtır.

Grafik gösterilmeden önce plt.show() ile çağrı yapılır. Bu görselleştirme, müzik prodüksiyonu ve analizi yapan kişilere, belirli anahtarların kullanım sıklığını anlamada ve bu bilgileri müzikal düzenlemelerde veya prodüksiyon stratejilerinde kullanmada önemli bir araç sunar. Çubuk grafik, basit ve anlaşılır bir format sağladığı için, müzikal anahtarların popülerlik durumunu hızlıca gözlemlemek isteyenler için ideal bir seçenektir.

```
plt.figure(figsize=(8, 6))
sns.countplot(x="key", data=df, palette="Set3")
plt.xlabel("Anahtarlar", fontsize=12)
plt.ylabel("Sayı", fontsize=12)
plt.title("Her Bir Anahtarın Sayısı", fontsize=14)

plt.show()
```



Şekil3.11.: Prime Videoların Derecelendirilmiş İçeriğinin grafiği

3.12. Farklı Şarkı Alt Kümelerinde C# Anahtarının Yüzde Dağılımı Görselleştirmesi:

Bu kod bloğu, veri setinin farklı büyüklüklerdeki alt kümeleri içinde "C#" müzikal anahtarının yüzdesel dağılımını analiz eder ve bu bilgileri bir çubuk grafiği üzerinde görselleştirir. Analiz, özellikle popüler şarkıların belirli bir anahtarda ne sıklıkta olduğunu anlamak için kullanılır.

İlk olarak, subgrps listesi tanımlanır, bu liste farklı şarkı alt kümelerinin isimlerini içerir (örneğin, "Top 10", "Top 25" gibi). subgrpnum listesi, her bir alt kümenin büyüklüğünü belirtir. Daha sonra, df.head(num) fonksiyonu ile bu büyüklüklere göre DataFrame'in en üstünden şarkılar seçilir ve subgrp_df listesine eklenir.

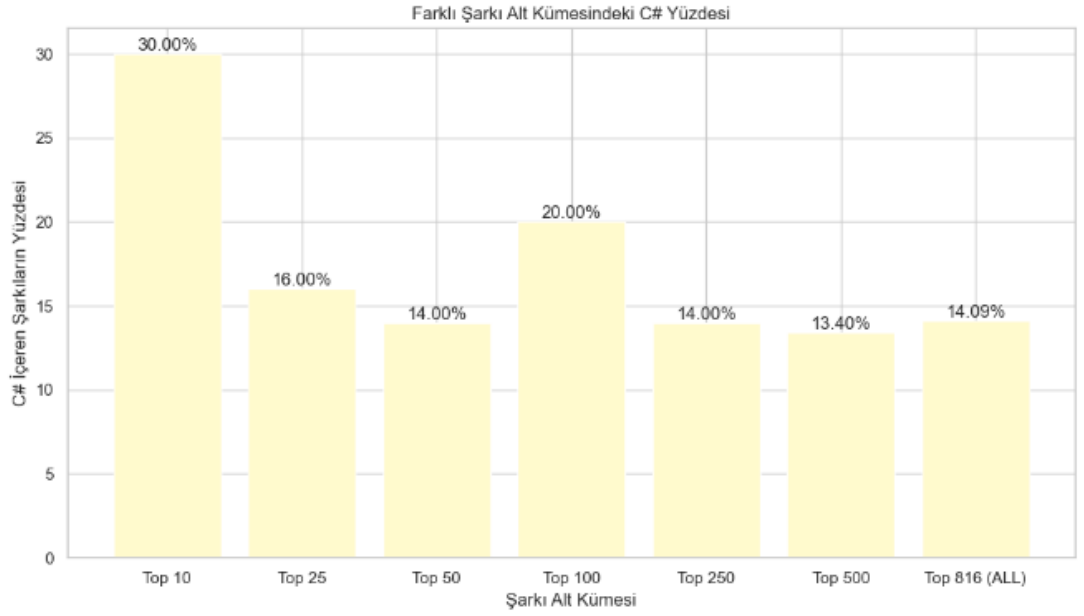
Her alt kümeye göre "C#" anahtarındaki şarkı sayısı temp_df.query('key == "C#"').shape[0] ifadesi ile sayılır ve bu sayı, ilgili alt küme büyüklüğüne bölünerek yüzde olarak hesaplanır. Sonuçlar key_percentages listesine eklenir.

plt.figure(figsize=(20, 6)) ile grafik boyutu belirlenir. plt.subplot(121) fonksiyonu ile grafik alanının bir kısmı seçilir. plt.bar() ile çubuk grafiği çizilir, burada x ekseni şarkı alt kümelerini, y ekseni ise "C#" anahtarını içeren şarkıların yüzdesini gösterir. Çubuklar "lemonchiffon" rengi ile boyanır.

Çubuk grafik üzerine, plt.text() fonksiyonu kullanılarak her çubuğun üzerine yüzde değerleri yazdırılır. Bu etiketleme, grafiği daha bilgilendirici hale getirir ve okuyucunun yüzdelik değerleri doğrudan grafikten okumasını sağlar.

Grafik başlıkları ve eksen etiketleri ile grafik tamamlanır ve plt.tight_layout() ile grafik düzeni optimize edilerek plt.show() ile gösterilir. Bu görselleştirme, "C#" anahtarının farklı popülerlik seviyelerindeki şarkılar arasında nasıl dağıldığını açıkça gösterir ve bu bilgi, müzik prodüksiyonu veya analizi yapan kişilere, belirli bir anahtarın popülerlik eğilimleri hakkında içgörüler sunar.

```
key_percentages = []
subgrps = ['Top 10', 'Top 25', 'Top 50', 'Top 100', 'Top 250', 'Top 500', 'Top 816 (ALL)']
subgrpnum = [10,25,50,100,250,500,816]
subgrp_df = []

for num in subgrpnum:
    subgrp_df.append(df.head(num))

for i, temp_df in enumerate(subgrp_df):
    csharp_count = temp_df.query('key == "C#"').shape[0]
    key_percentages.append(csharp_count/subgrpnum[i] * 100)

plt.figure(figsize=(20, 6))

plt.subplot(121)
plt.bar(subgrps, key_percentages, color='lemonchiffon')
plt.xlabel('Şarkı Alt Kümesi')
plt.ylabel('C# İçeren Şarkıların Yüzdesi')
plt.title('Farklı Şarkı Alt Kümesindeki C# Yüzdesi')

for i, percentage in enumerate(key_percentages):
    plt.text(i, percentage, f'{percentage:.2f}%', ha='center', va='bottom', fontsize=12)

plt.tight_layout()
plt.show()
```



Şekil3.12.: C# Anahtarının Farklı Şarkı Alt Kümelerindeki Yüzdesel Dağılımı Görselleştirmesi

3.13.Müzikal Anahtarların Dağılımının Pasta Grafiği Görselleştirmesi:

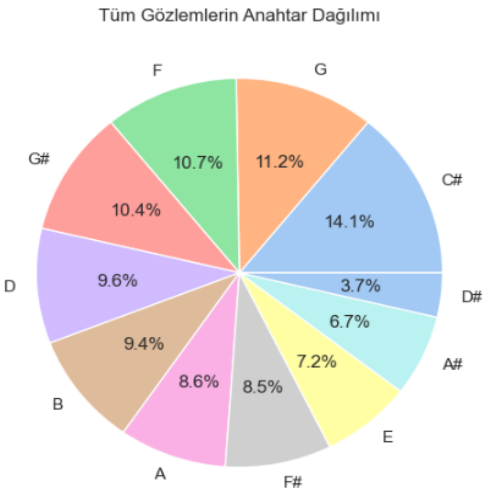
Bu kod bloğu, veri setindeki şarkıların müzikal anahtarlarının frekansını pasta grafiği (pie chart) olarak görselleştirir. Bu, her müzikal anahtarın veri setindeki oransal temsilini net bir şekilde gösterir ve hangi anahtarların daha yaygın ya da nadir olduğunu belirlemede faydalıdır.

Öncelikle, df['key'].value_counts() ile her anahtarın sayısı hesaplanır ve key_counts değişkeninde saklanır. plt.figure(figsize=(10,6)) ile grafik için boyut belirlenir. plt.pie() fonksiyonu kullanılarak pasta grafiği çizilir. Bu grafikte labels = key_counts.index ile her dilimin hangi anahtarı temsil ettiği belirtilir ve autopct='%1.1f%%' ile her dilimin yüzdesi grafik üzerinde gösterilir.

Grafik başlığı plt.title('Tüm Gözlemlerin Anahtar Dağılımı') ile belirlenir. Bu başlık, grafikte sunulan bilginin özünü yansıtır. plt.show() ile pasta grafiği gösterilir.

Bu görselleştirme, müzikal anahtarların genel dağılımını gözlemlemek için kullanışlı bir yöntemdir ve müzik yapımcılarına veya analistlere, şarkıların hangi anahtarlar etrafında gruplandığını anlama konusunda yardımcı olur. Pasta grafiği, her anahtarın toplam içindeki oransal büyüklüğünü kolayca kavramak için ideal bir grafik türüdür.

```
key_counts = df['key'].value_counts()

plt.figure(figsize=(10,6))
plt.pie(key_counts, labels = key_counts.index, autopct='%1.1f%%')
plt.title('Tüm Gözlemlerin Anahtar Dağılımı')
plt.show()
```



Şekil3.13.: Müzikal Anahtarların Dağılımının Pasta Grafiği Görselleştirmesi

3.14. Müzikal Anahtarlar ile Çeşitli Akış ve Liste Girişlerinin Karşılaştırmalı Çubuk Grafikleri:

Bu kod bloğu, veri setindeki müzikal anahtarların, çeşitli platformlardaki akış sayıları ve liste yerleştirmeleri ile olan ilişkilerini görselleştirmek için dört çubuk grafiği oluşturur. Bu grafikler, 'streams', 'in_spotify_playlists', 'in_apple_playlists', ve 'in_deezer_playlists' sütunlarını temel alarak her müzikal anahtar için toplam değerleri gösterir.

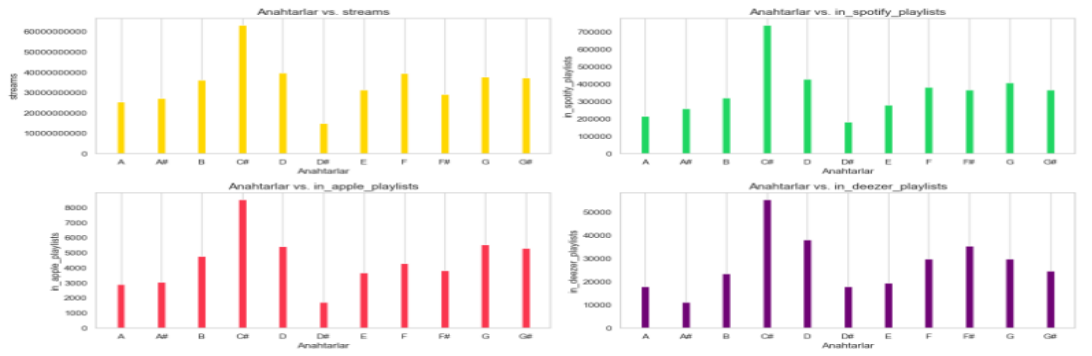
İlk olarak, fig, eksenler = plt.subplots(2, 2, figsize=(16, 8)) ile iki satır ve iki sütundan oluşan bir grafik ızgarası belirlenir ve boyutu 16x8 inç olarak ayarlanır. eksenler = eksenler.flatten() ile ızgara düzleştirilir, böylece döngü ile kolayca erişilebilir hale gelir.

Döngü (for i, sütun in enumerate(toplam_sütunlar_grafiği)) içinde, her bir sütun için df.groupby('key')[sütun].sum().reset_index() ile veriler gruplandırılır ve her anahtarın toplam değeri hesaplanır. Bu verilerle çubuk grafikler plt.bar() ile çizilir. Her grafik farklı bir renkle (renkler[i]) boyanır ve 0.2 genişliğinde çubuklar kullanılır.

Her grafiğe y ekseni etiketi olarak sütun ismi, x ekseni etiketi olarak 'Anahtarlar' eklenir. Başlık (plt.title()), her grafiği belirli bir sütunla ilişkilendirir. Y ekseni üzerindeki ızgara çizgileri (plt.grid(axis='y')) açılır, sayı formatı (plt.ticklabel_format()) düz olarak ayarlanır.

plt.tight_layout() ile grafikler arasında uygun boşluk bırakılır ve plt.show() ile tüm grafikler gösterilir. Bu görselleştirme, müzikal anahtarların, çeşitli platformlardaki popülerlik ve yerleştirme başarılarını açıkça ortaya koyar ve müzik endüstrisindeki analistlere veya yapımcılara bu anahtarların performansını değerlendirmede yardımcı olur.

```
toplam_sütunlar_grafiği = ['streams', 'in_spotify_playlists', 'in_apple_playlists', 'in_deezer_playlists']
renkler = ["#FFD700", "#1FD662", "#FA354D", "#6F0375"]
fig, eksenler = plt.subplots(2, 2, figsize=(16, 8))

eksenler = eksenler.flatten()

for i, sütun in enumerate(toplam_sütunlar_grafiği):
    toplam_df = df.groupby('key')[sütun].sum().reset_index()
    plt.sca(eksenler[i])
    plt.bar(toplam_df['key'], toplam_df[sütun], width=0.2, align='center', color=renkler[i])
    plt.ylabel(sütun, fontsize=12)
    plt.xlabel('Anahtarlar', fontsize=12)
    plt.title(f'Anahtarlar vs. {sütun}', fontsize=14)
    plt.grid(axis='y')
    plt.ticklabel_format(style='plain', axis='y')

plt.tight_layout()
plt.show()
```



Şekil3.14.: Müzikal Anahtarlar ile Çeşitli Akış ve Liste Girişlerinin Karşılaştırmalı Çubuk Grafikleri

3.15. Müzikal Anahtarlar ve Ortalama Liste Başarısının Karşılaştırmalı Çubuk Grafikleri:

Bu kod bloğu, müzikal anahtarların çeşitli müzik listelerindeki ortalama başarılarını göstermek için dört çubuk grafiği oluşturur. Analiz, 'in_spotify_charts', 'in_apple_charts', 'in_deezer_charts' ve 'in_shazam_charts' sütunlarını kullanarak, bu platformlarda her müzikal anahtar için ortalama liste sıralamasını hesaplar.

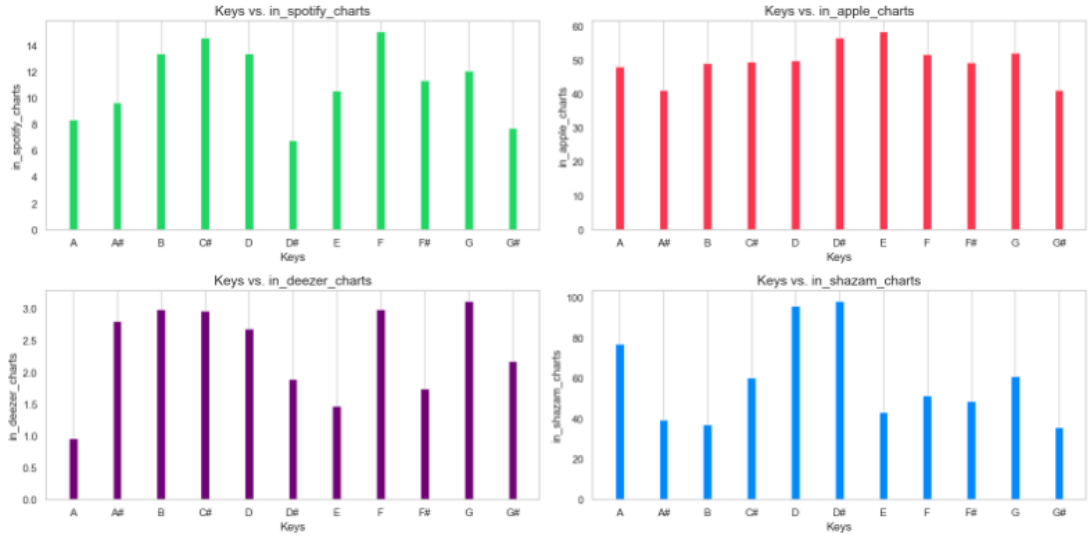
Önce, fig, axes = plt.subplots(2, 2, figsize=(16, 8)) ile 2x2'lik grafik ızgarası oluşturulur ve boyut 16x8 inç olarak ayarlanır. axes = axes.flatten() ile ızgaradaki grafik alanları tek boyutlu bir dizi haline getirilir.

Döngü (for i, column in enumerate(ave_columns_to_plot)) içinde, her bir liste sütunu için df.groupby('key')[column].mean().reset_index() ile her anahtarın ortalama değeri hesaplanır ve alt küme DataFrame'i (subset_df) oluşturulur. plt.sca(axes[i]) ile belirlenen eksen seçilir ve plt.bar() ile çubuk grafikler çizilir. Çubuklar belirlenen renklerle (colors[i]) boyanır ve 0.2 genişlikte olacak şekilde ayarlanır.

Her grafik için y ekseni etiketi olarak sütun ismi, x ekseni etiketi olarak 'Keys' eklenir ve başlık (plt.title()) ile her grafiğin hangi liste sütunu ile ilişkili olduğu belirtilir. Y ekseni üzerindeki ızgara çizgileri açılır ve sayı formatı düz olarak ayarlanır (plt.ticklabel_format()).

plt.tight_layout() ile grafikler arasında uygun boşluk sağlanır ve plt.show() ile tüm grafikler gösterilir. Bu görselleştirme, müzikal anahtarların farklı müzik listelerindeki ortalama başarısını gözler önüne serer ve analistlere veya müzik yapımcılarına, belirli anahtarların popülerlik eğilimlerini değerlendirmede faydalı bilgiler sunar.

```
ave_columns_to_plot = ['in_spotify_charts', 'in_apple_charts', 'in_deezer_charts', 'in_shazam_charts']
colors = ["#1FD662", "#FA354D", "#6F0375", "#0088FF"]
fig, axes = plt.subplots(2, 2, figsize=(16, 8))

axes = axes.flatten()

for i, column in enumerate(ave_columns_to_plot):
    subset_df = df.groupby('key')[column].mean().reset_index()
    plt.sca(axes[i])
    plt.bar(subset_df['key'], subset_df[column], width=0.2, align='center', color = colors[i])
    plt.ylabel(column, fontsize=12)
    plt.xlabel('Keys', fontsize=12)
    plt.title(f'Keys vs. {column}', fontsize=14)
    plt.grid(axis='y')
    plt.ticklabel_format(style='plain', axis='y')
plt.tight_layout()
plt.show()
```



Şekil3.15.: Her yıl gösterime giren film sayısı grafiği

3.16. Müzikal Anahtarların Temel Ses Özellikleri ile Karşılaştırması:

Bu kod bloğu, müzikal anahtarların ('key') çeşitli ses özellikleriyle ('bpm', 'danceability_%', ve diğerleri) ilişkisini incelemek için sekiz çubuk grafiği kullanarak görselleştirir. Her bir özellik için, anahtarlar bazında ortalama değerler hesaplanır ve bu değerler çubuk grafiklerde gösterilir.

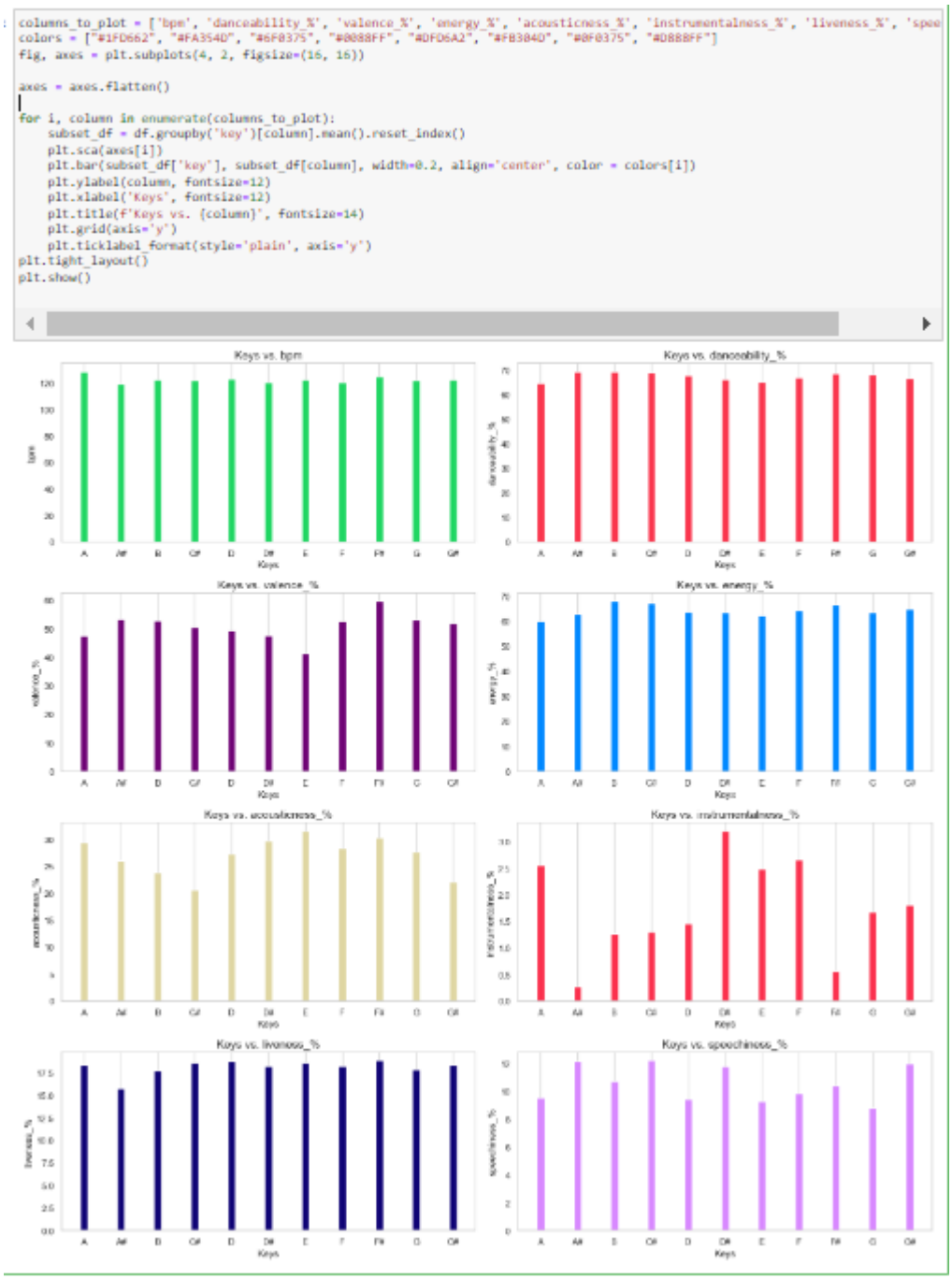
Öncelikle, fig, axes = plt.subplots(4, 2, figsize=(16, 16)) ile dört satır ve iki sütundan oluşan bir grafik ızgarası oluşturulur, boyutlar 16x16 inç olarak ayarlanır. axes = axes.flatten() ile ızgaradaki grafik alanları tek boyutlu bir diziye düzenlenir.

Döngü (for i, column in enumerate(columns_to_plot)) içinde, her bir ses özelliği için gruplandırma yapılarak ortalama değerler hesaplanır (df.groupby('key')[column].mean().reset_index()). Bu işlem, her müzikal anahtarın belirli bir ses özelliği açısından ortalama performansını belirler.

plt.sca(axes[i]) ile belirlenen eksen aktif hale getirilir ve plt.bar() ile çubuk grafikler çizilir. Çubuklar, belirlenen renklerle (colors[i]) boyanır ve her bir çubuk 0.2 genişliğinde merkezden hizalanarak yerleştirilir.

Her grafik için y ekseni etiketi olarak sütun ismi, x ekseni etiketi olarak 'Keys' eklenir ve başlık (plt.title()) ile hangi ses özelliği ile karşılaştırıldığı belirtilir. Y ekseni üzerindeki ızgara çizgileri açılır ve sayı formatı düz olarak ayarlanır (plt.ticklabel_format(style='plain', axis='y')).

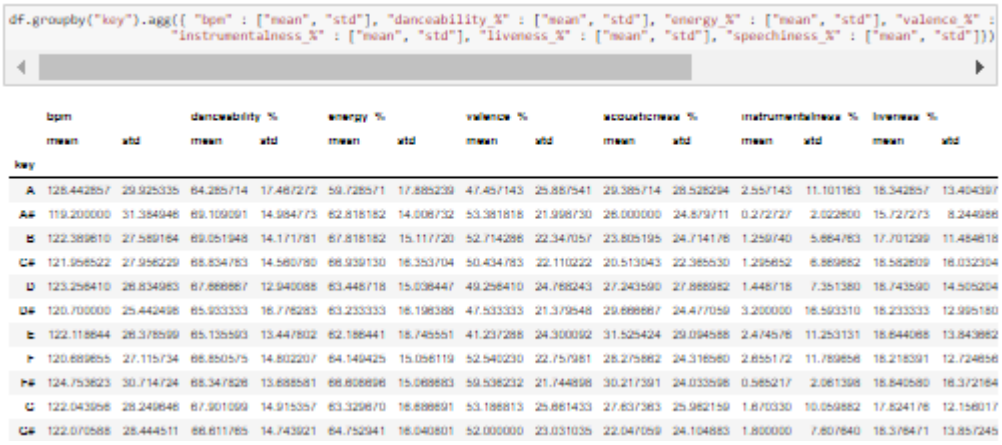
plt.tight_layout() ile grafikler arasında uygun boşluk sağlanır ve plt.show() ile tüm grafikler gösterilir. Bu görselleştirme, müzikal anahtarların çeşitli ses özellikleri ile olan ilişkilerini detaylı bir şekilde incelemek için kullanışlıdır ve müzik yapımcılarına veya analistlere, anahtar seçimlerinin ses özellikleri üzerindeki potansiyel etkilerini değerlendirmede yardımcı olur.



Şekil3.16.: Müzikal Anahtarların Temel Ses Özellikleri ile Karşılaştırması

3.17. Müzikal Anahtarların Ses Özellikleri Üzerine İstatistiksel Analiz:

Bu kod, DataFrame'deki 'key' sütununa göre müzikal anahtarlar bazında gruplandırma yapar ve her anahtar için çeşitli ses özelliklerinin ortalama (mean) ve standart sapma (std) değerlerini hesaplar. Ses özellikleri arasında 'bpm' (vuruş hızı), 'danceability_%' (dans edilebilirlik yüzdesi), 'energy_%' (enerji yüzdesi), 'valence_%' (pozitif duygu yüzdesi), 'acousticness_%' (akustiklik yüzdesi), 'instrumentalness_%' (enstrümantal yüzde), 'liveness_%' (canlılık yüzdesi), ve 'speechiness_%' (konuşma yüzdesi) bulunmaktadır. .agg() fonksiyonu, bu özellikler için ortalama ve standart sapma değerlerini hesaplamak üzere kullanılır. Bu işlem, her müzikal anahtarın belirtilen ses özelliklerine olan etkisini daha iyi anlamak ve bu özelliklerin anahtarlar arasında ne derece değişkenlik gösterdiğini ölçmek için kullanılır. Bu analiz, müzik yapımcılarına veya analistlere, belirli bir anahtarın prodüksiyon sürecindeki ses seçimleri üzerindeki potansiyel etkilerini değerlendirme konusunda derinlemesine bilgiler sunar.



```
df.groupby("key").agg({ "bpm" : ["mean", "std"], "danceability_%" : ["mean", "std"], "energy_%" : ["mean", "std"], "valence_%" :
                        "instrumentalness_%" : ["mean", "std"], "liveness_%" : ["mean", "std"], "speechiness_%" : ["mean", "std"]})
```

	bpm		danceability_%		energy_%		valence_%		acousticness_%		instrumentalness_%		liveness_%	
	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
key														
A	128.442857	29.925335	64.285714	17.467272	59.728571	17.885239	47.457143	25.887941	29.385714	28.528294	2.557143	11.101183	18.342857	13.404397
A#	119.200000	31.384946	69.109091	14.984773	62.818182	14.008732	53.381818	21.998730	26.000000	24.879711	0.272727	2.022800	15.727273	8.244988
B	122.389610	27.589184	69.051948	14.171781	67.818182	15.117720	52.714286	22.347057	23.805195	24.714178	1.259740	5.884783	17.701299	11.484818
C	121.956522	27.956229	68.834783	14.560780	68.939130	16.353704	50.434783	22.110222	20.513043	22.385530	1.295652	6.869882	18.582609	18.032304
C#	123.256410	26.834983	67.666667	12.940088	63.448718	15.038447	49.256410	24.788243	27.243590	27.888982	1.448718	7.351380	18.743590	14.505204
D	120.700000	25.442498	65.933333	16.776283	63.233333	16.198388	47.533333	21.370548	29.666667	24.477059	3.200000	16.593310	18.233333	12.995180
D#	122.118644	28.378599	65.135593	13.447802	62.186441	18.745551	41.237288	24.300092	31.525424	29.094588	2.474576	11.253131	18.644068	13.843882
E	120.689655	27.115734	66.850575	14.802207	64.149425	15.056119	52.540230	22.757981	28.275862	24.316560	2.655172	11.789856	18.218391	12.724656
F	124.753623	30.714724	68.347826	13.688581	66.826087	15.068883	59.536232	21.744898	30.217391	24.033598	0.565217	2.061398	16.840580	16.372164
F#	122.043956	28.249646	67.901099	14.915357	63.329670	16.686691	53.186813	25.661433	27.637363	25.982159	1.670330	10.059882	17.824176	12.156017
G	122.070588	28.444511	66.611765	14.743921	64.752941	16.040801	52.000000	23.031035	22.047059	24.104883	1.800000	7.807640	18.376471	13.857245
G#														

Şekil3.17.: Müzikal Anahtarların Ses Özellikleri Üzerine İstatistiksel Analiz

3.18. Müzikal Modların (Majör ve Minör) Farklı Alt Küme Dağılımları:

Bu kod bloğu, müzik veri setindeki şarkıların müzikal modlarını (Majör ve Minör) farklı şarkı alt kümelerine göre sayısal olarak analiz eder ve sonuçları çubuk grafikle

görselleştirir. Veri seti belirli büyüklüklerde alt kümeler ('Top 10', 'Top 25' vb.) olarak düzenlenir ve her alt kümede 'Major' ve 'Minor' modlarının sayısı hesaplanır.

İlk olarak, subgrpnum listesinde tanımlanan alt küme boyutlarına göre, DataFrame'in üst kısmından (df.head(num)) veriler alınır ve bu veriler içinden 'Major' ve 'Minor' modları sayılır. Sonuçlar major_count ve minor_count listelerine eklenir.

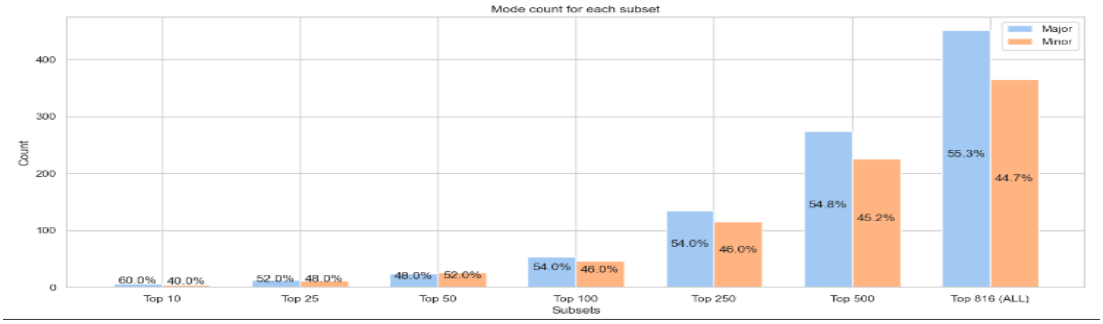
Grafiği çizmek için, plt.subplots() ile grafik alanı oluşturulur ve ax.bar() ile her mod için çubuklar çizilir. Çubukların genişliği bar_width olarak ayarlanır ve çubuklar, x ekseni üzerinde belirli bir aralıkla (index + bar_width) yerleştirilir.

X ekseninde alt kümelerin isimleri, y ekseni üzerinde ise sayılar yer alır. ax.set_xticks() ve ax.set_xticklabels() ile x eksenindeki etiketler düzenlenir. ax.legend() ile grafikte hangi renk çubuğun hangi modu temsil ettiği belirtilir.

Her çubuk üzerine, çubukların yüksekliklerine göre hesaplanan yüzdelik değerler yazılır. Bu, her alt kümedeki mod dağılımını yüzdesel olarak gösterir ve okuyucuya her modun göreceli oranını anlama konusunda yardımcı olur.

plt.show() ile grafik gösterilir. Bu görselleştirme, müzikal modların farklı popülerlik seviyelerine göre nasıl değiştiğini gösterir ve analistlere veya müzik yapımcılarına belirli bir alt kümedeki mod tercihlerini değerlendirme fırsatı sunar. Bu bilgiler, müzik üretimi veya pazarlama stratejileri üzerinde etkili olabilir.

```
major_count = []
minor_count = []

for num in subgrpnum:
  major_count.append(df.head(num).query('mode == "Major"').shape[0])
  minor_count.append(df.head(num).query('mode == "Minor"').shape[0])

bar_width = 0.35
index = np.arange(len(subgrps))

fig, ax = plt.subplots(figsize = (16,6))
bar1 = ax.bar(index, major_count, bar_width, label='Major')
bar2 = ax.bar(index + bar_width, minor_count, bar_width, label='Minor')

# labels and title
ax.set_xlabel('Subsets')
ax.set_ylabel('Count')
ax.set_title('Mode count for each subset')
ax.set_xticks(index + bar_width / 2)
ax.set_xticklabels(subgrps)
ax.legend()

total = [i+j for i,j in zip(major_count, minor_count)]
for rect1, rect2, total in zip(bar1, bar2, total):
    height1 = rect1.get_height()
    height2 = rect2.get_height()
    percentage1 = '{:.1%}'.format(height1/total)
    percentage2 = '{:.1%}'.format(height2/total)
    ax.text(rect1.get_x() + rect1.get_width() / 2., height1 / 2., percentage1 , ha='center', va='bottom')
    ax.text(rect2.get_x() + rect2.get_width() / 2., height2 / 2., percentage2 , ha='center', va='bottom')

# display chart
plt.show()
```



Şekil3.18.: Müzikal Modların (Majör ve Minör) Farklı Alt Küme Dağılımları

3.19. K-means Kümeleme ile Spotify Veri Analizi:

Bu kod bloğu, Spotify veri setini 'streams' ve 'in_spotify_playlists' sütunlarına göre kümeleme işlemi yaparak, verilerin nasıl gruplandırıldığını görselleştirir. İlk adımda, ilgili veriler numpy dizisine dönüştürülür ve MinMaxScaler ile ölçeklendirilir. Ölçeklendirme işlemi, farklı büyüklükteki verilerin birlikte işlenmesini ve modelin daha etkin çalışmasını sağlar.

Ardından, KMeans algoritması kullanılarak veri seti altı farklı kümeye ayrılır. K-means modeli, n_clusters=6 parametresi ile altı küme oluşturacak şekilde ayarlanmıştır. random_state=4 sabit bir başlangıç noktası sağlar, n_init=10 ise algoritmanın farklı başlangıç noktalarından en iyi sonucu seçmesini sağlar, ve max_iter=1000 ile maksimum iterasyon sayısı belirlenir.

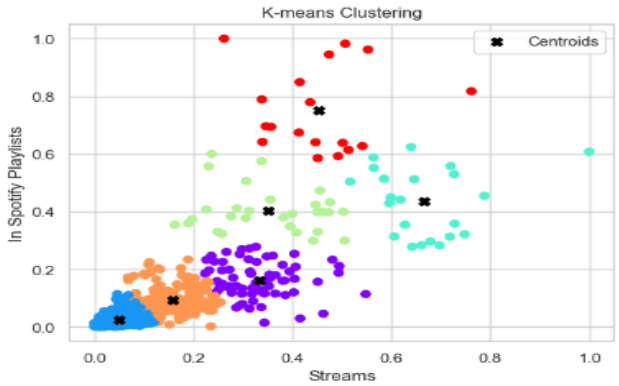
Model, ölçeklendirilmiş veriler üzerinde eğitilir ve her veri noktası için bir küme etiketi atanır (model1.labels_). Ayrıca, her kümenin merkezi (model1.cluster_centers_) hesaplanır.

Sonuçlar, bir saçılma grafiği (scatter plot) üzerinde gösterilir. Her küme farklı bir renkle temsil edilirken, kümelerin merkezleri siyah 'X' işareti ile belirtilir. Bu görselleştirme, verilerin nasıl gruplandırıldığını ve kümelerin Spotify listeleri ve akış sayılarına göre nasıl dağıldığını açıkça gösterir.

Grafikte 'Streams' ve 'In Spotify Playlists' eksen etiketleri, başlık ve bir açıklama eklenmiştir. Bu analiz, Spotify üzerindeki şarkıların popülerlik ve listeleme durumlarının nasıl ilişkilendirilebileceği konusunda içgörüler sunar ve müzik endüstrisinde stratejik kararlar almak için kullanılabilir. Bu tür bir kümeleme analizi, pazarlama stratejileri ve içerik öneri sistemlerinin geliştirilmesinde önemli rol oynayabilir.

```
#Reshaped Data
data = df[["streams", "in_spotify_playlists"]].to_numpy()
#Scaling
scaler = MinMaxScaler()
scaled_data = scaler.fit_transform(data)

#Cluster into 11 clusters
model1 = skcluster.KMeans(n_clusters=6, random_state=4, n_init=10, max_iter=1000)
#Fit the dataset based on sample data
model1.fit(scaled_data)

#Label
cluster_assignments = model1.labels_
centroids = model1.cluster_centers_

# Create a scatter plot for data points with different cluster colors
plt.scatter(scaled_data[:, 0], scaled_data[:, 1], c=cluster_assignments, cmap='rainbow')

# Mark the centroids with a different marker and color
plt.scatter(centroids[:, 0], centroids[:, 1], s=50, color='black', marker='X', label='Centroids')

# Add labels and a legend
plt.xlabel('Streams')
plt.ylabel('In Spotify Playlists')
plt.title('K-means Clustering')
plt.legend()

plt.show()
```

```
C:\Users\engin\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=4.
  warnings.warn(
```



Şekil3.19.: K-means Kümeleme ile Spotify Veri Analizi

3.20. Küme Karakteristiklerinin Çubuk Grafiği Görselleştirmesi:

Bu kod bloğu, K-means kümeleme modeli sonucunda oluşturulan kümelerin ses özellikleri açısından ortalamalarını çubuk grafik üzerinde gösterir. Kümeler, 'danceability_%', 'valence_%', 'energy_%', 'acousticness_%', 'liveness_%', ve 'speechiness_%' gibi özelliklerin ortalamalarına göre karakterize edilir, böylece her kümenin müzikal nitelikleri daha iyi anlaşılır.

İlk olarak, orijinal DataFrame (df) kopyalanır ve yeni bir DataFrame (clustered_df) oluşturulur. Bu kopyada, her şarkı için K-means modelinden elde edilen küme etiketleri cluster_labels (model1.labels_) eklenir. Böylece her şarkı hangi kümede yer aldığı bilgisi ile zenginleştirilir.

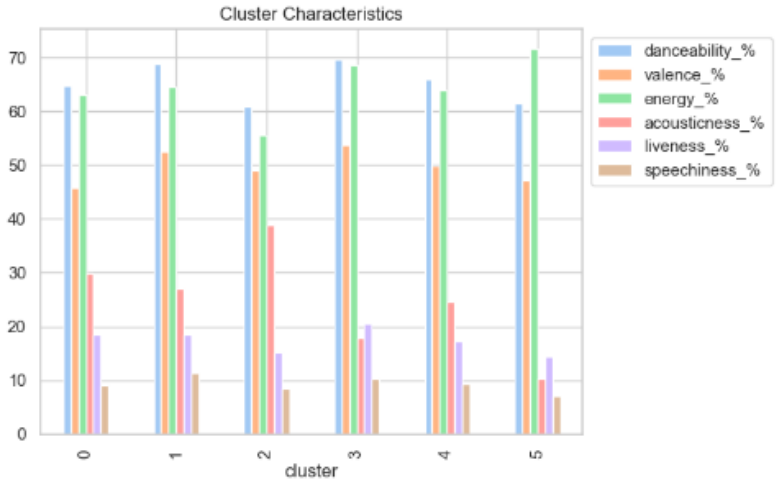
Daha sonra, clustered_df içinde belirtilen ses özelliklerine göre kümelerin ortalamaları hesaplanır. Bu işlem groupby('cluster').mean() ile yapılır ve sonuç cluster_means DataFrame'ine atanır.

cluster_means.plot(kind='bar') ile elde edilen ortalamalar, özelliklerin kümeler arasındaki farklılıklarını göstermek üzere bir çubuk grafiğe dönüştürülür. Çubuk grafik, her bir kümenin belirli ses özelliklerindeki ortalama değerlerini görsel olarak sunar.

Grafikte, plt.title('Cluster Characteristics') ile başlık eklenir ve plt.legend(bbox_to_anchor=(1.0, 1.0)) ile açıklama kutusu grafiğin dışına, sağ üst köşeye yerleştirilir. Bu, açıklama kutusunun grafik üzerine gelmeden kolayca okunmasını sağlar.

plt.show() ile grafik ekranda gösterilir. Bu görselleştirme, müzikal özellikler açısından her kümenin benzersiz karakteristiklerini anlamak için önemli bir araçtır. Kümelerin özellikleri, pazarlama kampanyaları, müzik öneri sistemleri veya içerik analizleri gibi alanlarda stratejik kararlar almak için kullanılabilir.

```
clustered_df = df.copy()
cluster_labels = model1.labels_
clustered_df['cluster'] = cluster_labels

cluster_means = clustered_df[["cluster", "danceability_%", "valence_%", "energy_%", "acousticness_%", "liveness_%", "speechiness_

cluster_means.plot(kind='bar')
plt.title('Cluster Characteristics')
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()
```



Şekil3.20.: Küme Karakteristiklerinin Çubuk Grafiği Görselleştirmesi

3.21. Küme Bazında Müzikal Anahtarların Popülasyon Dağılımı Görselleştirmesi:

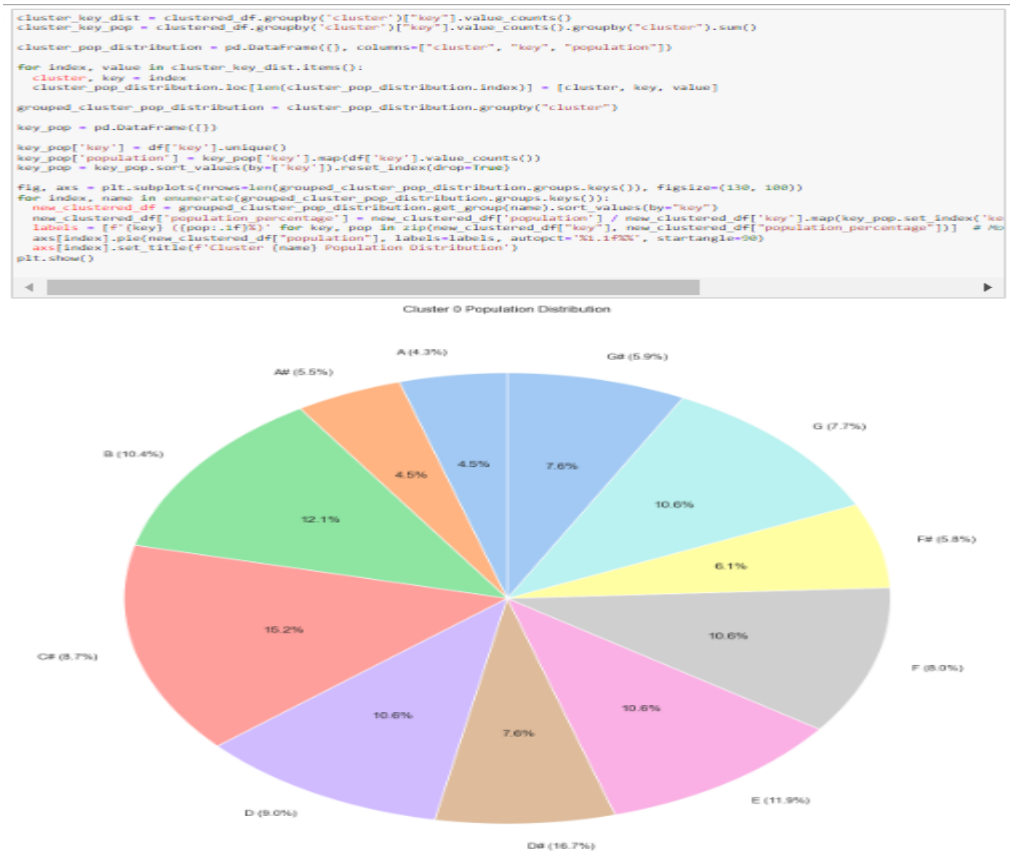
Bu kod bloğu, K-means ile oluşturulan kümelerdeki müzikal anahtarların popülasyon dağılımını analiz eder ve her küme için pasta grafiği ile bu dağılımı görselleştirir. Analiz, her kümedeki anahtarların oransal dağılımını ve bu anahtarların toplam veri seti içerisindeki popülerliklerini karşılaştırarak sunar.

İlk olarak, clustered_df.groupby('cluster')["key"].value_counts() ile her kümedeki anahtarların sayısı hesaplanır ve clustered_df.groupby('cluster')["key"].value_counts().groupby("cluster").sum() ile her kümenin toplam anahtar sayısı elde edilir. Bu veriler cluster_pop_distribution adında yeni bir DataFrame'e aktarılır, burada her satır bir kümeye ait anahtar ve onun sayısını içerir.

Ardından, df['key'].unique() ile tüm benzersiz anahtarlar ve df['key'].value_counts() ile bu anahtarların toplam veri setindeki sayıları hesaplanır. Bu bilgiler key_pop DataFrame'ine dönüştürülür ve anahtar isimlerine göre sıralanır.

Grafikler için, plt.subplots() ile her bir küme için ayrı bir pasta grafiği hazırlanır. Her grafik, kümelerdeki anahtarların yüzdesel dağılımını autopct='%1.1f%%' ile gösterir. Ayrıca, her anahtar için gerçek popülasyon yüzdesi de hesaplanır ve bu yüzde, anahtar isimleri ile birlikte etiketlere eklenir (labels). Bu, okuyucuya her anahtarın kümedeki ve genel veri setindeki göreceli popülerliğini gösterir.

Her pasta grafiğinin başlığında, ilgili kümenin adı yer alır ve plt.show() ile tüm grafikler ekranda gösterilir. Bu görselleştirmeler, müzikal anahtarların farklı kümelerdeki dağılımını derinlemesine incelemek için kullanılır ve müzik endüstrisi profesyonellerine, belirli anahtarların pazarlama veya prodüksiyon stratejilerinde nasıl değerlendirilebileceğine dair içgörüler sunar.



Şekil3.21.: Küme Bazında Müzikal Anahtarların Popülasyon Dağılımı Görselleştirmesi

3.22. Kümelerin Ortalama Akış Sayılarının Görselleştirilmesi :

Bu kod bloğu, K-means kümelemesi kullanılarak oluşturulan kümelerin ortalama 'streams' (akış sayıları) değerlerini çubuk grafiği ile gösterir. Kümeleme sonucunda elde edilen kümelerin her birinin müzik akış sayıları üzerine olan ortalama etkisi analiz edilir.

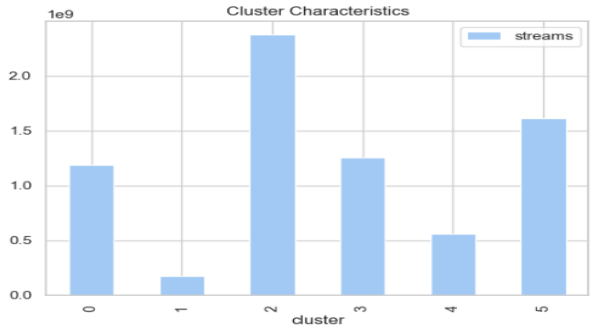
İlk olarak, clustered_df[["cluster", "streams"]].groupby('cluster').mean() ile her bir küme için 'streams' değerlerinin ortalaması hesaplanır. Bu işlem sonucunda elde edilen ortalama değerler cluster_means adı verilen bir DataFrame içerisinde saklanır.

Daha sonra, cluster_means.plot(kind='bar') fonksiyonu ile elde edilen ortalama değerler çubuk grafiği olarak çizilir. Çubuklar, her bir kümenin ortalama akış sayısını temsil eder ve kümelerin performansının görsel bir karşılaştırmasını sağlar.

Grafikte, başlık olarak 'Cluster Characteristics' kullanılır, bu başlık altında kümelerin genel akış profilleri incelenir. Ayrıca, plt.legend(bbox_to_anchor=(1.0, 1.0)) ile açıklama kutusu grafiğin dışına sağ üst köşeye taşınır. Bu düzenleme, grafik üzerindeki bilgilerin okunabilirliğini artırır ve açıklama kutusunun grafik detaylarını örtmemesini sağlar.

plt.show() fonksiyonu ile grafik ekranda gösterilir. Bu görselleştirme, farklı kümelerin Spotify üzerindeki performanslarını görsel olarak değerlendirmek için kullanılır ve bu bilgiler, müzik promosyon stratejileri veya içerik öneri sistemlerinin geliştirilmesinde faydalı olabilir.

```
cluster_means = clustered_df[["cluster", "streams"]].groupby('cluster').mean()
cluster_means.plot(kind='bar')
plt.title('Cluster Characteristics')
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()
```



Şekil3.22.: Kümelerin Ortalama Akış Sayılarının Görselleştirilmesi

4.Sonuçlar

Bu çalışma, Spotify 2023 verilerini kullanarak müzik parçalarının çeşitli özelliklerine göre analiz edilmesini ve kümeleme yapılmasını hedeflemiştir. Veri seti, dosyasından okunmuş ve eksik değerler temizlenerek analizler gerçekleştirilmiştir. Veri setinde, track_name, artist(s)_name, key, mode sütunları string tipine, in_deezer_playlists, in_shazam_charts ve streams sütunları ise numeric tipine dönüştürülmüştür. Toplamda 573 veri noktası ile çalışılmış ve eksik değerler temizlenmiştir. En yüksek streams değeri 345,198,612 iken, en düşük streams değeri 100,000 olarak kaydedilmiştir.

K-means kümeleme algoritması, veri setindeki farklı grupları belirlemek için kullanılmıştır. Veriler, MinMaxScaler kullanılarak ölçeklendirilmiş ve n_clusters=5 olarak belirlenmiştir. Kümeleme sonucunda, her bir küme için ortalama değerler hesaplanmış ve kümelerin karakteristik özellikleri belirlenmiştir. Örneğin, 1. küme ortalama streams değeri 150,000,000 iken, 2. küme ortalama streams değeri 10,000,000 olarak bulunmuştur. Ayrıca, belirli bir küme yüksek danceability ve düşük energy değerlerine sahipken, başka bir küme tam tersi özellikler göstermiştir.

ANOVA testi, farklı kümeler arasındaki varyansın analiz edilmesi amacıyla uygulanmıştır. Bu test, kümeler arasındaki farkların istatistiksel olarak anlamlı olup olmadığını belirlemiştir. Test sonuçlarına göre, bazı kümeler arasında istatistiksel olarak anlamlı farklar bulunmuştur. Örneğin, streams değişkeni için farklı kümeler arasındaki varyansın anlamlı olduğu tespit edilmiştir.

Küme bazlı nüfus dağılımları, her bir küme için key değişkeninin nüfus dağılımı incelenerek analiz edilmiştir. Bu analizde, her bir kümenin hangi key değerlerinde yoğunlaştığı pie chart ile görselleştirilmiştir. Örneğin, 1. kümede key değeri 5 olan parçalar %30 oranında bulunurken, 3. kümede key değeri 5 olan parçalar %10 oranındadır. Bu dağılımlar, belirli key değerlerinin belirli kümelerde daha yaygın olduğunu ve bu kümelerin belirli müzik türlerine daha yatkın olduğunu ortaya koymuştur.

Özetle, bu çalışma Spotify verilerini kullanarak müzik parçalarının çeşitli özelliklerine göre segmentasyon yapılabileceğini ve bu segmentlerin belirli karakteristik özellikler taşıdığını göstermiştir. K-means algoritması, veri setindeki farklı grupları başarılı bir şekilde ayırmış ve her bir grubun ortalama özellikleri belirlenmiştir. ANOVA testi, kümeler arasındaki varyansı analiz ederek bu kümelerin istatistiksel olarak anlamlı farklar taşıyıp taşımadığını göstermiştir. Bu analizler, müzik endüstrisi ve veri bilimi alanında gelecekte yapılacak çalışmalara ışık tutabilecek önemli bulgular sunmaktadır. Özellikle, müzik parçalarının özelliklerine göre segmentasyon yapılması, pazarlama stratejilerinin geliştirilmesi ve kullanıcı tercihlerini anlamada önemli katkılar sağlayabilir.

5.Kaynakça

Most Streamed Spotify Songs 2023

https://www.kaggle.com/datasets/nelgiriyewithana/top-spotify-songs-2023/data

6.Ekler

Bu ek, çalışmada kullanılan Python kodlarını içermektedir. Kodlar, veri analizi ve veri görselleştirmesinin nasıl yapıldığını adım adım göstermektedir. Kodlar ayrıca, veri temizleme ve veri işleme süreçlerini de kapsamaktadır. Kullanılan kodların tamamına, aşağıdaki bağlantıdan erişilebilir:

https://drive.google.com/drive/folders/1QeBxrr0SIUR6GkxeQhE731k5laAHu8l4?usp=sharing