Group 4 - Spotify Song Recommender

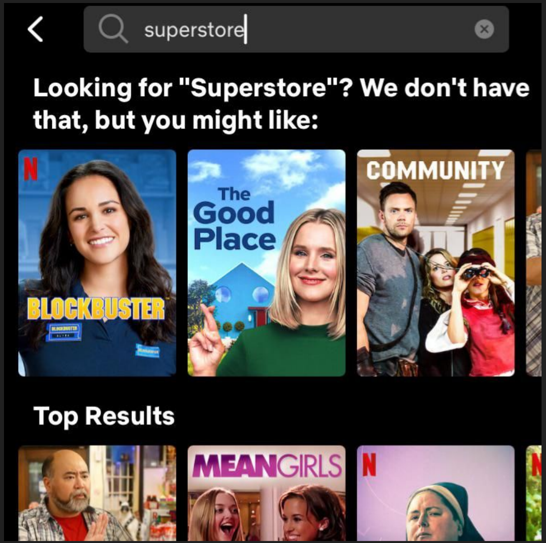
Cagan Abney, Joshua Hale, Jason Cisneros and Raheem Yusuff

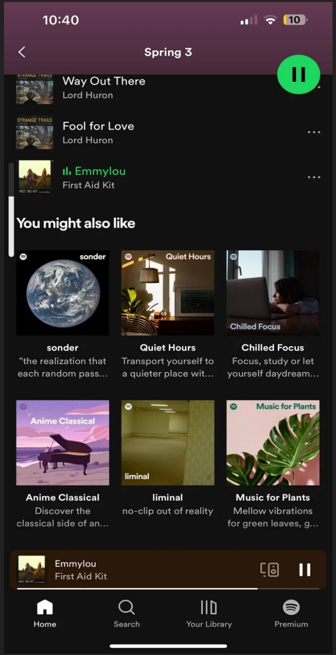
Github

Inspiration

When you are watching Netflix, how do they decide what we should watch next?

How does Spotify decide what song I might want to listen to next? As we all love listening to music and are always looking for that new song we can't stop playing or get out of our head, how can we make finding this song easier?





Dataset

Spotify Tracks Dataset : https://www.kaggle.com/datasets/maharshipandya/-spotify-tracks-dataset/data

Columns:

- **Track ID**
- **Track Name**
- **Track Artists**
- **Album Name**
- **Track Genre**
- **Popularity** : between 0 and 100, with 100 being the most popular. Calculated by algorithm on the total number of plays the track has had and how recent those plays are.
- **Audio Features** [song duration, explicitness, danceability, energy, key, loudness, mode (major or minor), speechiness, acousticness, instrumentalness, liveness, valence, tempo, time signature]

Data Engineering

Dropped one row of null values

```
In [5]:
1 df = df.dropna(subset=['album_name'])
2 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 113999 entries, 0 to 113999
Data columns (total 21 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   Unnamed: 0        113999 non-null  int64
 1   track_id          113999 non-null  object
 2   artists           113999 non-null  object
 3   album_name        113999 non-null  object
 4   track_name        113999 non-null  object
 5   popularity        113999 non-null  int64
 6   duration_ms       113999 non-null  int64
 7   explicit          113999 non-null  bool
 8   danceability      113999 non-null  float64
 9   energy            113999 non-null  float64
 10  key               113999 non-null  int64
 11  loudness          113999 non-null  float64
 12  mode              113999 non-null  int64
 13  speechiness       113999 non-null  float64
 14  acousticness      113999 non-null  float64
 15  instrumentalness  113999 non-null  float64
 16  liveness          113999 non-null  float64
 17  valence           113999 non-null  float64
 18  tempo             113999 non-null  float64
 19  time_signature    113999 non-null  int64
 20  track_genre       113999 non-null  object
dtypes: bool(1), float64(9), int64(6), object(5)
memory usage: 18.4+ MB
```

Removed all duplicate songs on track ID

```
In [6]:
1 df.drop_duplicates(subset=['track_id'], keep='first', inplace = True)
2 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 89740 entries, 0 to 113999
Data columns (total 21 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Unnamed: 0        89740 non-null  int64
 1   track_id          89740 non-null  object
 2   artists           89740 non-null  object
 3   album_name        89740 non-null  object
 4   track_name        89740 non-null  object
 5   popularity        89740 non-null  int64
 6   duration_ms       89740 non-null  int64
 7   explicit          89740 non-null  bool
 8   danceability      89740 non-null  float64
 9   energy            89740 non-null  float64
 10  key               89740 non-null  int64
 11  loudness          89740 non-null  float64
 12  mode              89740 non-null  int64
 13  speechiness       89740 non-null  float64
 14  acousticness      89740 non-null  float64
 15  instrumentalness  89740 non-null  float64
 16  liveness          89740 non-null  float64
 17  valence           89740 non-null  float64
 18  tempo             89740 non-null  float64
 19  time_signature    89740 non-null  int64
 20  track_genre       89740 non-null  object
dtypes: bool(1), float64(9), int64(6), object(5)
memory usage: 14.5+ MB
```

Dataset Pt. 2

Most Streamed Spotify Songs 2023 Dataset:
https://www.kaggle.com/datasets/nelgiriyewithana/top-spotify-songs-2023

Columns:

- **Track name,**
- **Artist(s) name,**
- **Released year**
- **In spotify playlists**
- **In spotify charts**
- **streams**
- **in_apple_charts**
- **danceability_%**
- **energy_%**
- **acousticness_%**
- **instrumentalness_%**

Data Engineering

This dataset was already very clean, we simply dropped 2 columns that had null values and saved to a new CSV.

```python
# Check for null values and data types
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
```

```python
# drop columns with null values that we will not use
df = df.drop('key', axis=1)
```

```python
df = df.drop('in_shazam_charts', axis=1)
```

Machine Learning: k-Nearest Neighbors (kNN)

- Supervised Machine Learning Model used for Classification or Regression
- kNN was first developed by Evelyn Fix and Joseph Hodges in **1951** in the context of research performed for the US military.

- **How does it work:**
  - Uses proximity to make classifications or predictions about the grouping around an individual data point
  - In order to determine which data points are closest to a given query point, the distance between the query point and the other data points will need to be calculated
  - Based on the k value is how many nearest neighbors the model will predict

kNN Distance Metrics

Euclidean - most commonly used distance measure, measures a straight line between the query point and the other point being measured.

$$d(x,y) = \sqrt{\sum_{i=1}^{n} (y_i - x_i)^2}$$

Manhattan - referred to as taxicab distance or city block distance as it is commonly visualized with a grid to navigate from one address to another via city streets

$$d(x,y) = \left(\sum_{i=1}^{m} \left| x_i - y_i \right| \right)$$

Minkowski - generalized form of Euclidean and Manhattan distance metrics

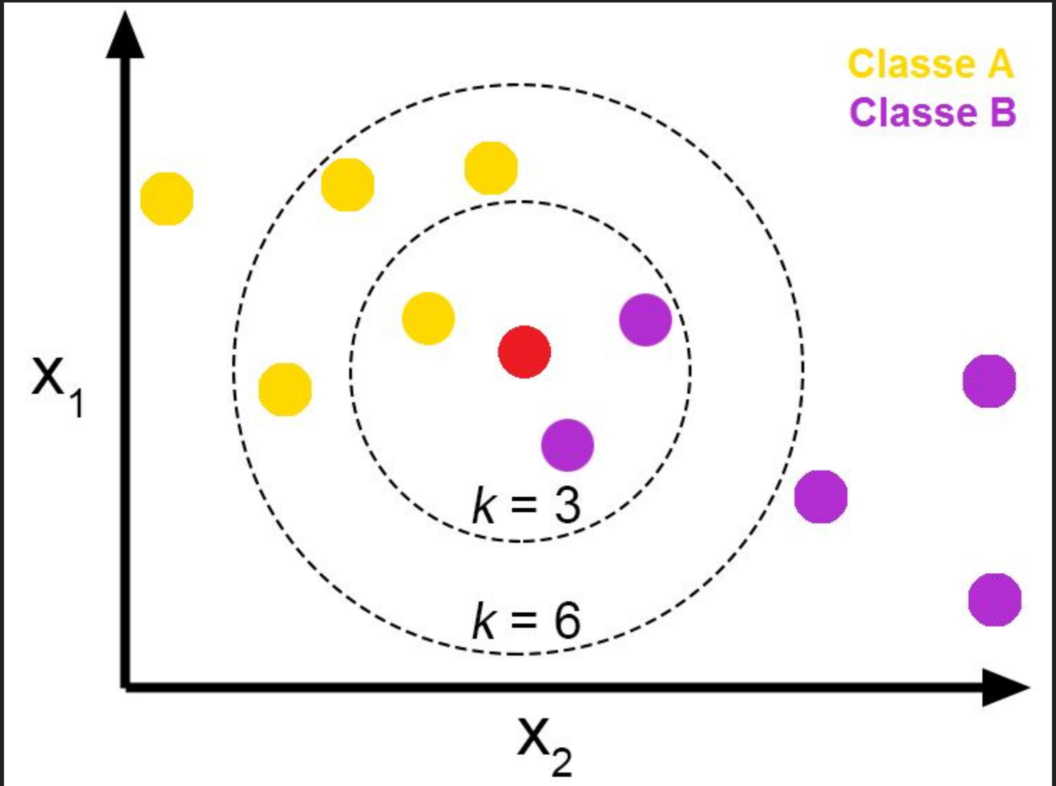
$$\left(\sum_{i=1}^{n} \left| x_i - y_i \right| \right)^{1/p}$$

Hamming - typically used with Boolean or string vectors, identifying the points where the vectors do not match.

$$D_H = \left(\sum_{i=1}^{k} \left| x_i - y_i \right| \right)$$

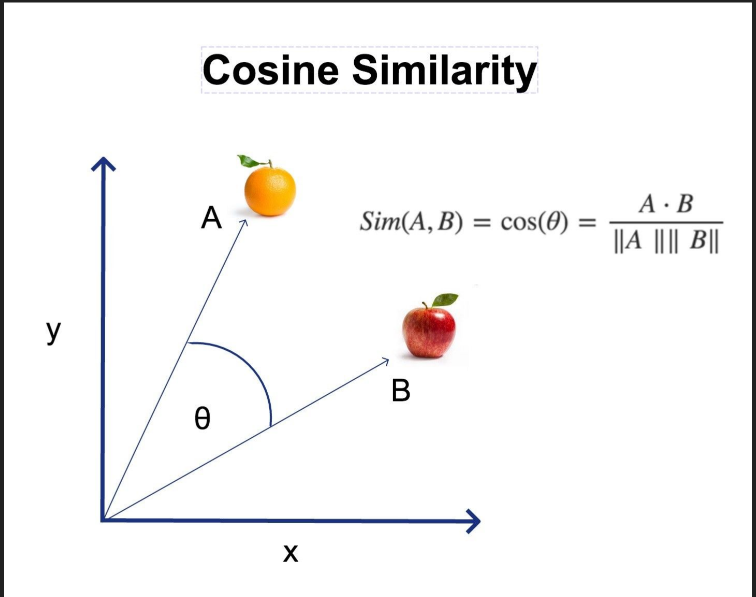
$$\begin{array}{ll} x=y & D=0 \\ x\neq y & D\neq 1 \end{array}$$

kNN Made Simple



kNN: Cosine Similarity

- While the other metrics use distance, Cosine Similarity will use the cosine of the angle between two non-zero vectors



Feature Engineering

Isolated numeric features

In [8]:
```
numeric_df['explicit'] = numeric_df['explicit'].astype(float)
```

In [9]:
```
# Assuming 'df' is your DataFrame
int_columns = numeric_df.select_dtypes(include=['int64']).columns
numeric_df[int_columns] = numeric_df[int_columns].astype('float64')

# Verify the data types after conversion
print(numeric_df.dtypes)
```

```
popularity          float64
duration_ms         float64
explicit            float64
danceability        float64
energy              float64
key                 float64
loudness            float64
mode                float64
speechiness         float64
acousticness        float64
instrumentalness    float64
liveness            float64
valence             float64
tempo               float64
time_signature      float64
dtype: object
```

Encoded Track Genre

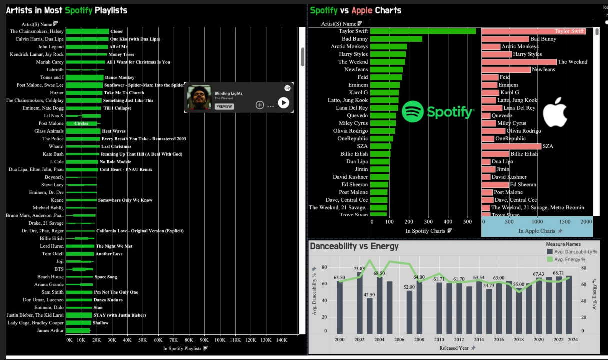
In [18]:
```
genre_df = pd.get_dummies(df,
                          columns = ['track_genre'],
                          prefix = ['track_genre'])
genre_df.head()
```

Out[18]:

rack_genre_alt-rock	track_genre_alternative	track_genre_ambient	track_genre_anime
False	False	False	False
False	False	False	False
False	False	False	False
False	False	False	False
False	False	False	False

Application Demo: https://jabney12.pythonanywhere.com/

Tableau Engineering



Our first tableau dashboard was created with the Most Streamed Spotify Songs 2023 Dataset.

Visualization Titles

- Artists in Most Spotify Playlists
- Spotify vs Apple Charts
- Danceability vs Energy

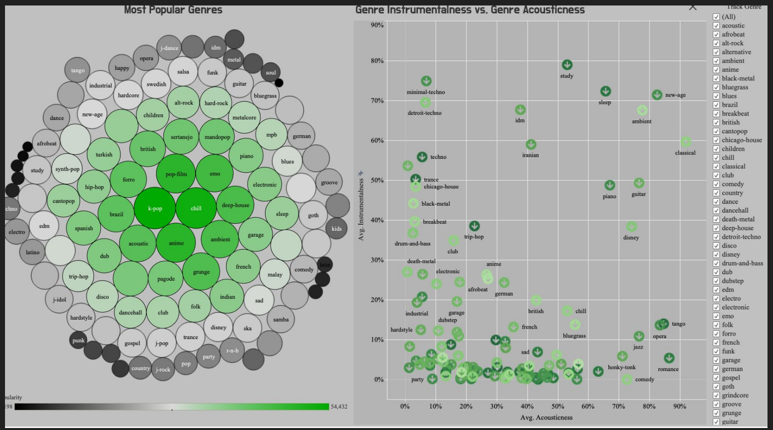
For the “Artists in Most Spotify Playlists”, the Artist’s bar changes in size based on # of streams per song. We also embedded Spotify to showcase the #1 song.

We added a global filter to change the released years for the Playlist and Charts bar graphs.

By filtering the Danceability vs Energy graph by songs released after the year 2000, there is a clear correlation between the two features.

Playlist Dashboard

Tableau Engineering



Our second dashboard was created with the Spotify Tracks Dataset

Visualization Titles

- Most Popular Genres
- Genre Instrumentalness vs Genre Acousticness

The Most Popular Genres are shown by bubble size and color, with green being most popular and black being least popular.

Genre Instrumentalness vs Genre Acousticness is shown in a scatter plot with a global Genre filter.

[Genre Dashboard]

Conclusions

- Taylor Swift is the most popular artists right now
- K-pop is most popular genre as of 2023 on Spotify
- Recommender systems in our everyday life likely use a supervised classification machine learning.
- kNN is a simple classification model good for beginners to learn the complexed world of machine learning.
- Endless amounts of opportunities to optimize machine learning models to achieve your expected outcome.
- Data Science is pretty cool.

Limitations/Bias

Some limitations included our newly introduced machine learning knowledge. With more experience we would be able to use more complex models or better optimize our feature engineering.

Another limitation was the dataset, of course the dataset could not obtain every song in Spotify due to size limitations, so the user can not just input any song.

Limitations also included the data columns, one data set did not have a genre column and one didn’t have a time column (Released day, month, year).

Bias also ties into the song limitation, since the number and choice of songs were limited, there is a bias against smaller, less known artists, as well as older songs. Especially since one dataset is based on 2023 popularity

Future Work

- Use other classification machine learning models if KNN truly is the best for a recommendation system.
  - try other distance metrics to see how different our recommendations would be
- Adding more user customization to the application
  - Allow for minor errors in user input (mismatch capitalization or no space)
  - input multiple songs user likes
  - input only an Artist the user likes
  - select what distance metric you would like model to use
  - Auto finish or text box suggestion
- Add more songs

Works Cited

- Algolia. “Cosine Similarity: What Is It and How Does It Enable Effective and Profitable Recommendations?” Algolia Blog, www.algolia.com/blog/ai/cosine-similarity-what-is-it-and-how-does-it-enable-effective-and-profitable-recommendations/.

- IBM. “K-Nearest Neighbors (KNN).” IBM, www.ibm.com/topics/knn.
- Kaggle. “Spotify Tracks Dataset.” Kaggle, www.kaggle.com/datasets/maharshipandya/-spotify-tracks-dataset/data.
- Kaggle. “Top Spotify Songs 2023.” Kaggle, www.kaggle.com/datasets/nelgiriyewithana/top-spotify-songs-2023.
- "K-Nearest Neighbors (KNN)." Elastic, www.elastic.co/what-is/knn.
- Scikit-learn. “sklearn.neighbors.NearestNeighbors.” Scikit-learn Documentation, scikit-learn.org/stable/modules/generated/sklearn.neighbors.NearestNeighbors.html.