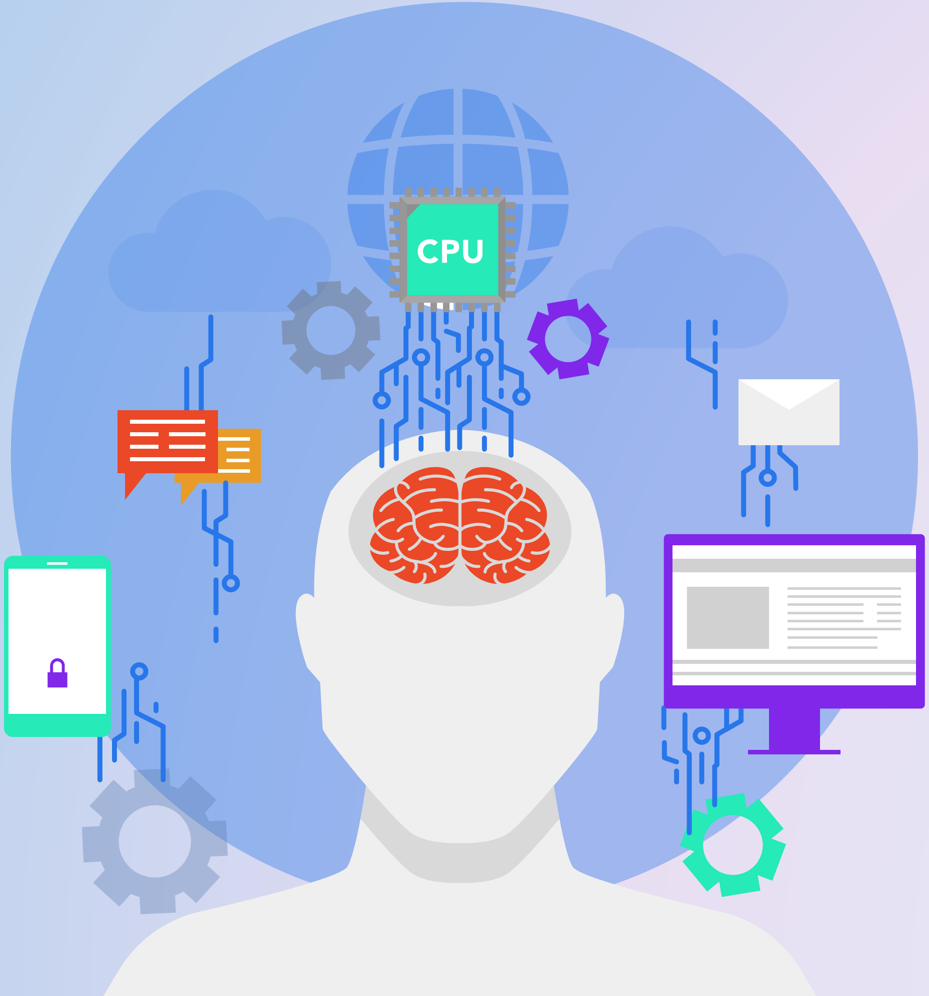


Machine Learning Team

Lily Djami, Numan Tok

Overview

1 → 2 → 3

1 Team Structure & Responsibilities

2 ML Nutrition Project

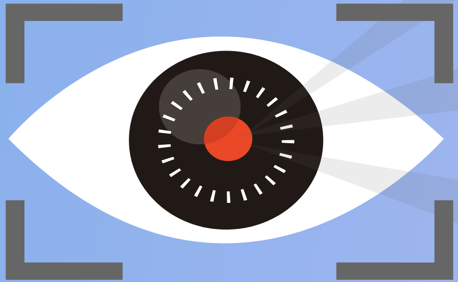
I. Requirements
II. System Architectures - Faster R-CNN
III. Initial Results
IV. Retraining and Final Results

3 ML Spotify Project

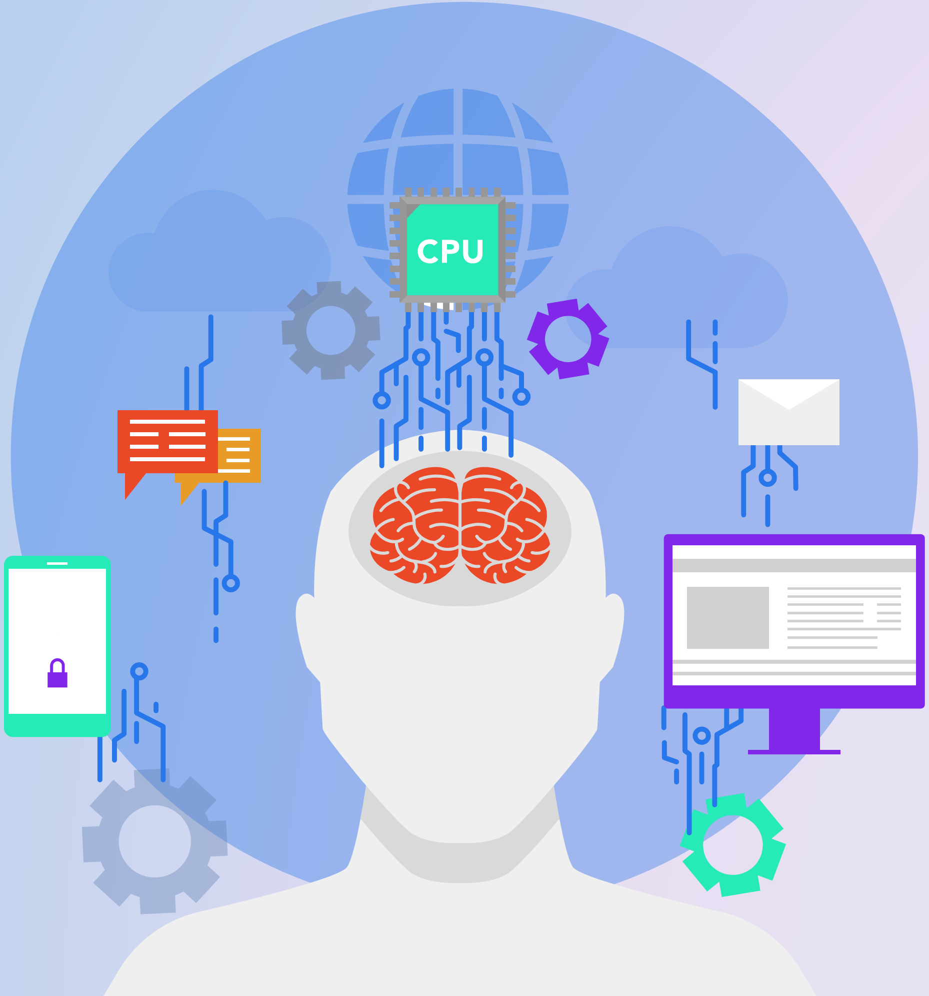
I. Requirements
II. Understanding the Fundamentals
III. Data Fundament & API possibilities
IV. Introducing 2 Solutions
V. Testing & Optimization
VI. Code Walkthrough

Team Structure and Responsibilities

Specialization in supporting different Projects



News Summary	Ilir Hajrullahu
Nutrition Project	Lily Djami
Spotify Project	Numan Tok



Nutrition Group Object Detection

Machine Learning Model

Requirement Questions for Nutrition Project

General Questions:

- What dataset are you going to use?
- What is the expected input?
- How accurate should the model be?
- What is the output that you want, what do you want to predict?
- What is the desired speed of the output creation?
- How do you judge if a predicted output is good or bad?

Specific for Nutrition Project:

- Is there enough image data, or would simulated data be needed?
- Would the model be used to only detect food from a specific cuisine (Asian/Western/etc.)?
- Will there be multiple objects (different fruits) in the picture or just one? Do you want to detect them one after another or as a group?

Requirements - Nutrition Group

- **Problem**: detection of food ingredients from an image (object detection)
- **Input**: Image with multiple ingredients
- **Output**: List of detected ingredients
- **Good prediction**: all food items are correctly identified.
- **Accuracy**: 90%, but user can delete bad predictions
- **Speed**: less than 1 minute
- **Type of food**: no information
- **Datasets**: food image datasets based on initial results (fruits and vegetables, sausages, bread, grocery items)

Figure: Sample input image

System Architecture - Faster R-CNN

- One of the most popular class of algorithms for object detection
- Works similarly to CNN in image classification, however the classification is region based.
- Searches through image for possible regions of interests.
- Extract regions using their bounding boxes and classify using CNN.
- Faster R-CNN the latest version with faster region search algorithm

System Architecture - Faster RCNN

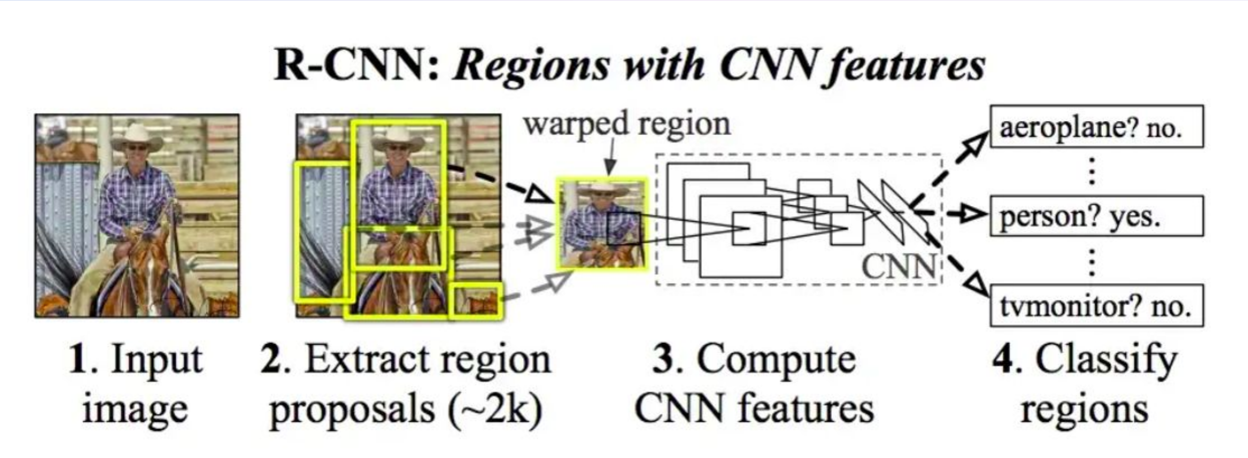


Figure: The R-CNN architecture

Initial Results

Figure: Initial results. Items on the top row was detected, but not items on the bottom row

- Base model: Faster R-CNN trained with ResNet-50-FPN from PyTorch, minimum size 4096
- Detected: cheese, orange (lemon), egg
- Not detected: milk, bread, peppers

→ Retraining to improve results

Datasets for Retraining

After discussion with the nutrition group, the following datasets are chosen:

- Sausage
- Bread
- Fruits and vegetables
- Freiburg Groceries

Training:

- Training on Google Colab
- Label set collected from all datasets (66 labels)
- The same label set used for all training
- Base model trained on 50 epochs/dataset

Figure: Sample images from the datasets.

Results

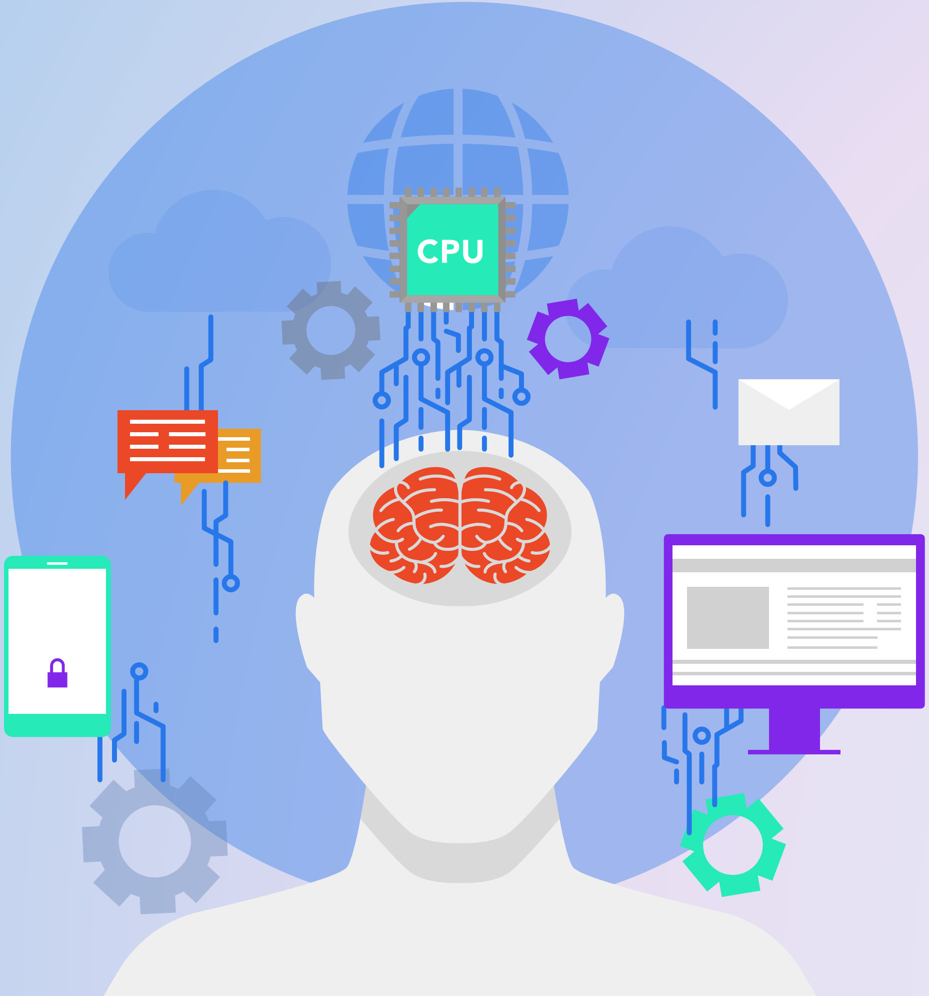
Retraining resulted in worse results

- Mislabeling, less detected objects

Possible causes:

→ Label set misalignment
→ Could be fixed with further re-training with correct label set, but there was no time.
→ Used the rest of the time to help integration with Nutrition group.

Figure: Sample images from the datasets.



Spotify Group Recommendation

Machine Learning Model

Requirement Questions for Spotify Project

General Questions:

- What dataset are you going to use?
- What is the expected input?
- How accurate should the model be?
- What is the output that you want, what do you want to predict?
- What is the desired speed of the output creation?
- How do you judge if a predicted output is good or bad?

Specific for Spotify Project:

- Do you want a feedback loop to the model?
- Which features do you want to use for the prediction? Length of song, mood, …?
- Do we have access to the Blend feature functions to possibly build on? (e.g. in the Spotify API)
- Do we also have access to functions of the "Recommended" feature?
- Is there a group size limit or other restrictions?

Requirements - Spotify Group

- **Problem**: Create a group playlist that satisfies the music taste of all group members.
- **Input**: Can be defined by ML Group. Spotify can format/parse it accordingly etc.
- **Output**: List of matching tracks (Track IDs).
- **Good prediction**: No specification
- **Accuracy**: No specification
- **Speed**: No specification
- **Feedback loop:** Users will be able to rate playlists
- **Features:** Free to choose for ML group.
- **Blend feature:** Does not seem to be available.
- **"Recommended" feature:** Available
- **Limits or other restrictions:** No specification
- **Datasets**: Data about the songs and the playlists (from API). Among other things, genre, abstract popularity info and some user-related information are available.

Understanding the Fundamentals

Problem:

Create a group playlist that satisfies the music taste of all group members.

To solve this, we should think about:

- What is a Music Track?
- What is a Playlist?

What is a Music Track?

1) Instruments: A track usually includes different kinds of instruments or electronic sounds that form the basis of a track.
2) Vocals: A track can contain vocals from one or many persons that may be detailed lyrics or just single words.
3) Effects: Auto tune, echos, fade in/out, delays,…
4) Tempo: The track speed that is measured in BPM (beats per minute).
5) Composition: A track may include several parts with different characteristics
6) Mode, Key & time signatures: Track features that influence the overall mood
7) … Has many characteristics

What is a Music Playlist?

1) A sequence of tracks
2) In most cases including tracks with similar characteristics
3) The tracks may or may not be in a harmonic order
   (Sorted in a way that 2 consecutive tracks have a low difference of their characteristics)

Data Fundament & API possibilities

- Get Track's Audio Features
  - "danceability": 0.696
  - "energy": 0.905
  - "key": 2
  - "loudness": -2.743
  - "mode": 1
  - "acousticness": 0.011
  - "instrumentalness": 0.000905
  - "liveness": 0.302
  - "valence": 0.625
  - "tempo": 114.944
  - "duration_ms": 207960
  - "time_signature": 4

Data Fundament & API possibilities

- Get Track’s Audio Features
  - "danceability": 0.696
  - "energy": 0.905
  - "key": 2
  - "loudness": -2.743
  - "mode": 1
  - "acousticness": 0.011
  - "instrumentalness": 0.000905
  - "liveness": 0.302
  - "valence": 0.625
  - "tempo": 114.944
  - "duration_ms": 207960
  - "time_signature": 4

- Get Track
  - Release_date
  - Duration
  - Popularity
- Get Track‘s Audio Analysis
  - Offset_seconds
  - Window_seconds
  - End_of_fade_in / Start_of_fade_out
  - … More detailed info about some audio features
- Get user‘s saved tracks: (limit 20)
- Get a list of Spotify featured playlists (limit 50)
- Get user‘s playlists: (limit 50)
- Get user‘s top tracks : (limit 20)

Solution 1 – Clustering + ANN (hybrid model)

Clustering

- Try to find the most common similarities of the tracks that the different group members listen to:
  1. Group tracks into clusters based on the similarity of the audio features
  2. Find the most promising cluster to use it as an “ideal characteristics representative” by statistical criteria like
     - Percentage of covered tracks
     - Balancing of user representation
     - Additional ranking of tracks (e.g. by recency of when the track was saved to the library; initially a use of listening counts was planned)

Solution 1 – Clustering + ANN (hybrid model)

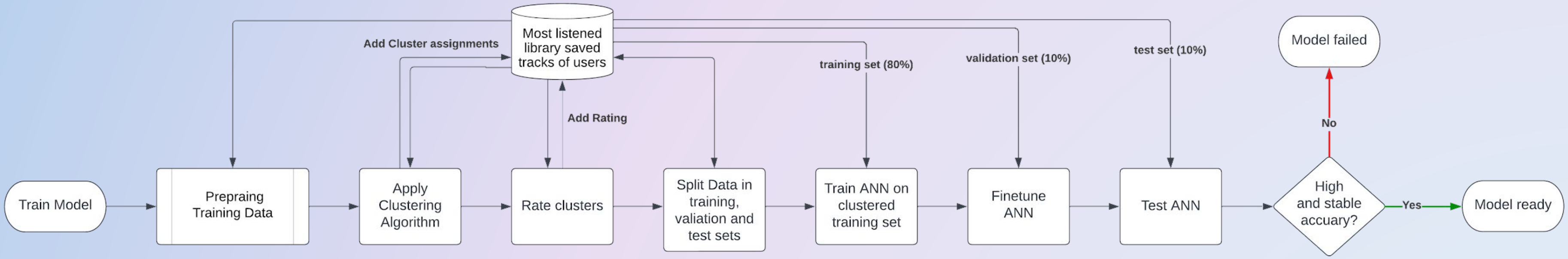
Clustering

- Try to find the most common similarities of the tracks that the different group members listen to:
  1. Group tracks into clusters based on the similarity of the audio features
  2. Find the most promising cluster to use it as an “ideal characteristics representative” by statistical criteria like
     - Percentage of covered tracks
     - Balancing of user representation
     - Additional ranking of tracks (e.g. by recency of when the track was saved to the library; initially a use of listening counts was planned)

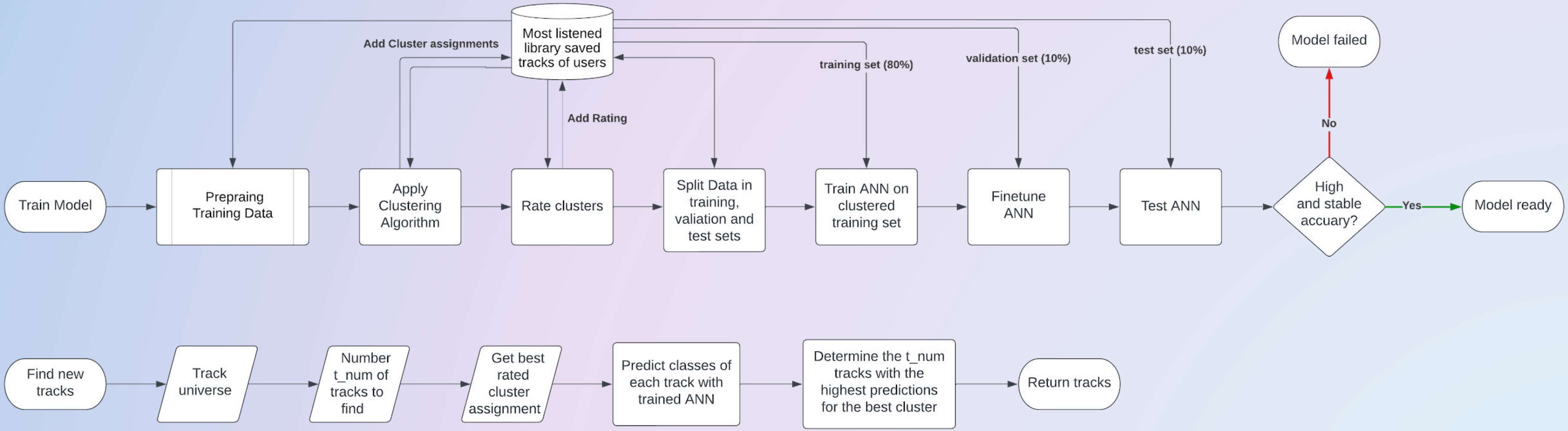
ANN

- Train an ANN to predict if a song matches the “ideal characteristics representative”
  1. Use the track’s audio features and cluster labels as training input
  2. Predict the class of new songs
     - The higher the predicted value for the most promising cluster, the more suitable it is for the playlist

Solution 1 – Clustering + ANN (hybrid model)



Solution 1 – Clustering + ANN (hybrid model)



Solution 2 – Graph model

 Find out how the audio features of consecutive tracks change in the playlists of all group members. Determine a weight vector as importance measure for the audio features:

1. Calculate the differences between the audio features of each pair of consecutive tracks
2. Sum the differences per audio feature = Vector of summed feature distances (“overall_distances”)
3. A lower difference is interpreted as a higher importance ⇒ get a higher weight

   (feature_weights = 1/ overall_distances)

Solution 2 – Graph model

 Find out how the audio features of consecutive tracks change in the playlists of all group members. Determine a weight vector as importance measure for the audio features:

1. Calculate the differences between the audio features of each pair of consecutive tracks
2. Sum the differences per audio feature = Vector of summed feature distances (“overall_distances”)
3. A lower difference is interpreted as a higher importance ⇒ get a higher weight

   (feature_weights = 1/ overall_distances)

 Create an undirected complete graph for all selectable tracks with

- Nodes = Tracks
- Edge weights = Weighted difference between the audio features of the connected Nodes

Solution 2 – Graph model

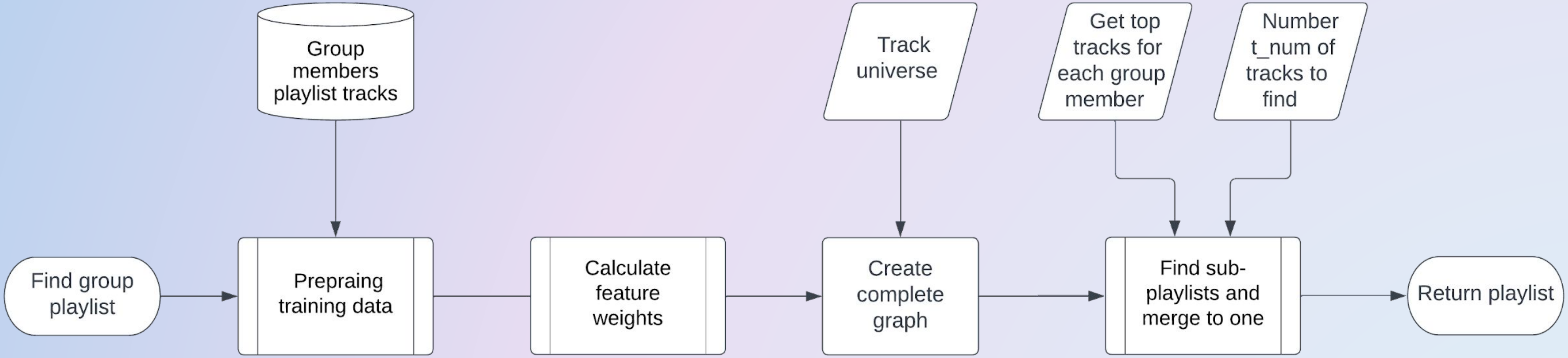
- Find out how the audio features of consecutive tracks change in the playlists of all group members. Determine a weight vector as importance measure for the audio features:
    1. Calculate the differences between the audio features of each pair of consecutive tracks
    2. Sum the differences per audio feature = Vector of summed feature distances (“overall_distances”)
    3. A lower difference is interpreted as a higher importance ⇒ get a higher weight

       (feature_weights = 1/ overall_distances)

- Create an undirected complete graph for all selectable tracks with
    - Nodes = Tracks
    - Edge weights = Weighted difference between the audio features of the connected Nodes

- Create a playlist by finding paths through the graph
    1. Specify a set of starting points: For each user, select his/her 3 top tracks (available through Spotify API)
    2. For each start track, find a sub-playlist (aka path) by successively adding the nearest neighbor in the graph
    3. Concatenate the sub-playlists that together form the overall result

Solution 2 – Graph model



Model Comparison

Clustering + ANN

- Higher complexity / more parts
- More influencing parameters that need to be carefully chosen (e.g. selection of Clustering algorithm, Sizes and numbers of layers)
- Depends on well-chosen cluster goodness measure
- Depends on finding a good cluster
- Lower Transparency of Decisioning
- Higher computational cost

Model Comparison

Clustering + ANN

- Higher complexity / more parts
- More influencing parameters that need to be carefully chosen (e.g. selection of Clustering algorithm, Sizes and numbers of layers)
- Depends on well-chosen cluster goodness measure
- Depends on finding a good cluster
- Lower Transparency of Decisioning
- Higher computational cost

Graph model

- Quality of playlist depends on start songs (but 3 top songs for each should be a good basis)
- There may be interruptions of the playlist flow because of the sub-playlist approach
- There could be too much similarity of the tracks

Could be countered by adding random noise, prefiltering the track universe or other techniques

Testing

- Unit Tests for each method
- Integration Tests: for combinations of methods, modules and classes and for the whole ML part
- Static testing: walkthroughs with the Spotify project team
- Performance Tests
  - optimization through:
    1. Replacing for loops by list comprehensions
    2. Minimizing API calls by retrieving the audio features and playlists for whole chunks at once (API limits of 100 tracks and 50 playlists)

Runtime Optimization

```
--- retrieve_audio_features: 1.6789839267730713 seconds ---
--- retrieve_audio_features: 2.0816640853881836 seconds ---
--- retrieve_audio_features: 1.311901569366455 seconds ---
--- retrieve_audio_features: 3.132132053375244 seconds ---
--- retrieve_audio_features: 4.034107208251953 seconds ---
--- retrieve_audio_features: 2.705784797668457 seconds ---
--- retrieve_audio_features: 3.4747555255889893 seconds ---
--- custom_audio_features_playlists: 142.38457870483398 seconds ---
--- prepare_user_playlists: 142.[illegible]0485696792603 seconds ---
```

```
--- retrieve_audio_features: 0.08209395408630371 seconds ---
--- retrieve_audio_features: 0.12010526657104492 seconds ---
--- retrieve_audio_features: 0.09958815574645996 seconds ---
--- retrieve_audio_features: 0.0886235237121582 seconds ---
--- retrieve_audio_features: 0.10134363174438477 seconds ---
--- retrieve_audio_features: 0.2505049705505371 seconds ---
--- retrieve_audio_features: 0.1434917449951172 seconds ---
--- custom_audio_features_playlists: 13.96532940864563 seconds ---
--- prepare_user_playlists: 14.1[illegible]85040473937988 seconds ---
```

1. Optimization
   - On retrieve_audio_features
   - -90% runtime

```
ode\ML model\graph-solution> python ml_main.py
--- retrieve_audio_features: 0.13012146949768066 seconds ---
--- prepare_track_universe: 19.1[illegible]0339822769165 seconds ---
```

```
 model\graph-solution> python ml_main.py
--- retrieve_audio_features: 1.1218020915985107 seconds ---
--- prepare_track_universe: 6.92[illegible]5781688690186 seconds ---
```

2. Optimization
   - On prepare_track_universe
   - -64% runtime

MANY THANKS!

Questions?

Systems & Software Engineering I

Lectured by:
Prof. Dr. Visvanathan Ramesh

Presentation by:
Lily Djami
Numan Tok

Frankfurt, March 16th 2023