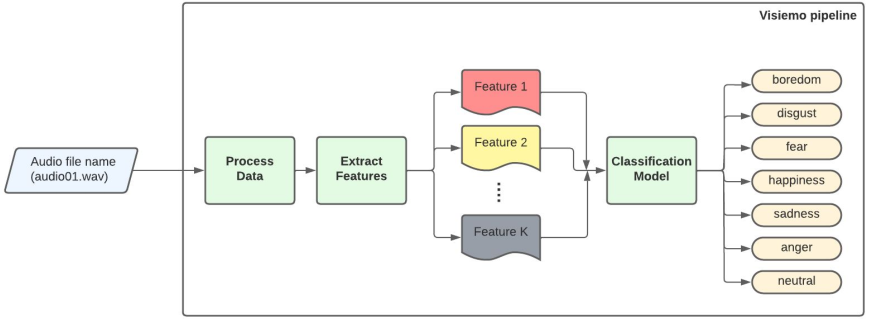
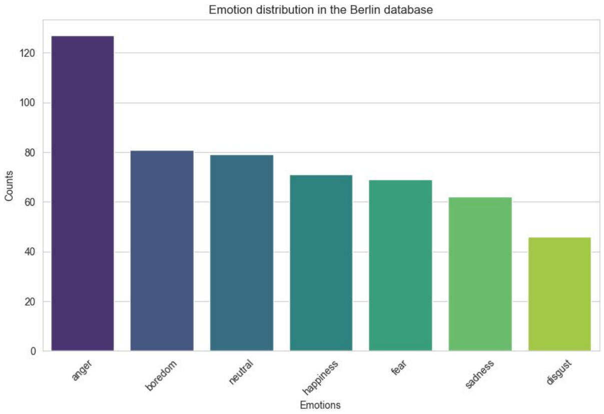
Enhancing Customer Satisfaction through Sentiment Analysis

Context and Objectives





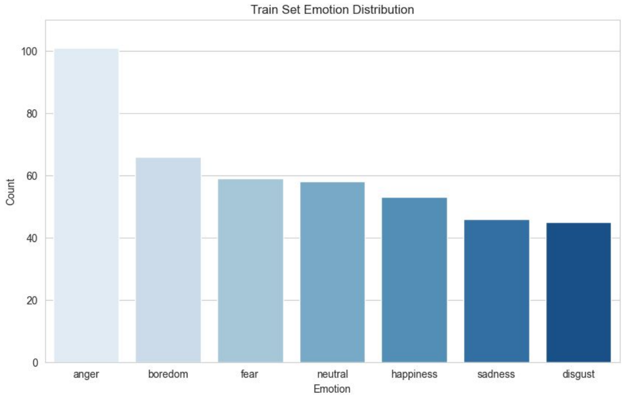
Main objectives

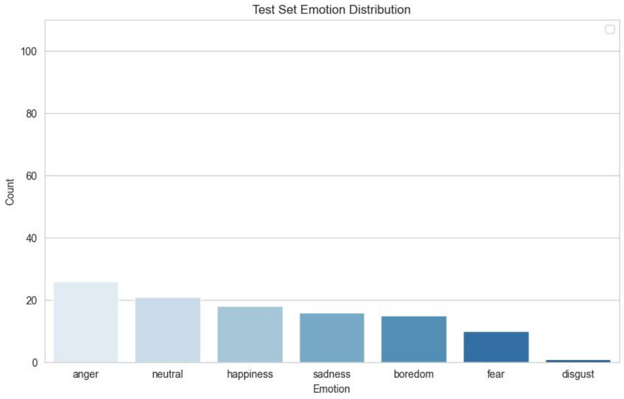
- Train a Speech Emotion Recognition using the Berlin Database
- Dockerize the project
- Construct a Flask API with 2 endpoints:
  - One for training the model
  - One for querying the last trained model with an audiofile of the data

Context and Application

- Detect the sentiment of call recordings
- The model has to analyze the satisfaction of customers

Constructing the train and test sets





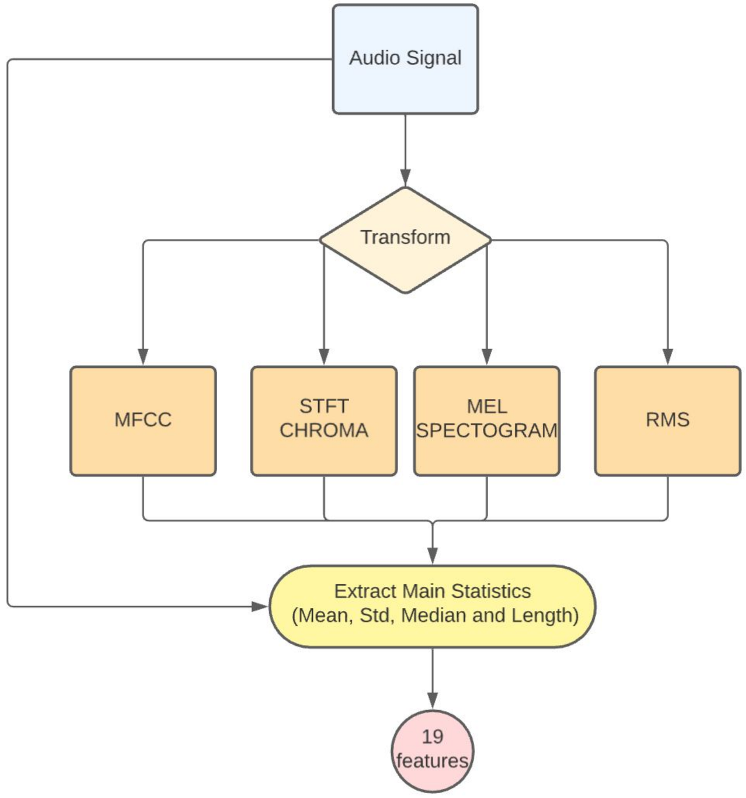
Train set has data of 8 users and contains 428 recordings

The train set is used for **selecting** the **best features, training the models** and **select the best hyperparameters** of models

Test set has data of 2 users and contains 107 recordings

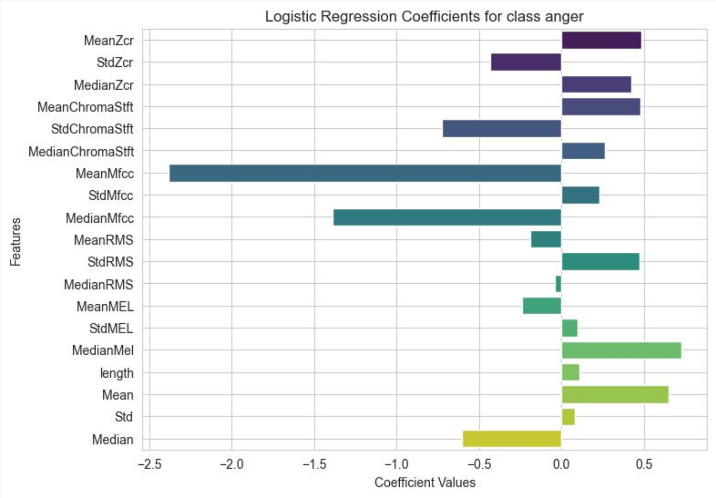
The test set is **ONLY** used to **evaluate** the performance of the **final models.**

Features of the Model



Features

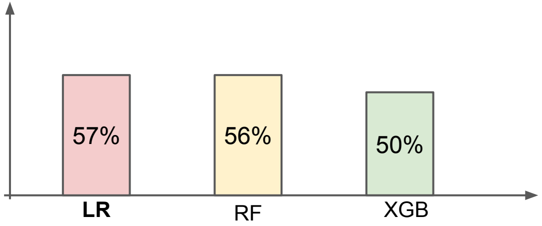
- We transform the signal and extract features widely used in speech recognition.
- For each of this features we take the main statistics as the final features.
- Lot of information is lost on the process however the PoC already provides good enough results and is more explanatory



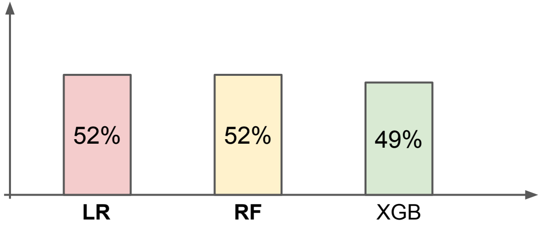
Classical ML Approach - CV Results

Main Metrics

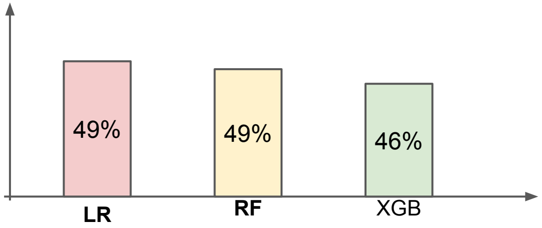
- Precision



- Recall



- F1



- Model tested:
  a. Logistic Regression
  b. Random Forest (num estimators: 50, 100, 200)
  c. Xgboost (num estimators: 50, 100, 200)

- Validation Technique:
  a. Cross Validation by user.
  b. The train set has 8 users then we iterate 8 times

- Main Results:
  a. Best results for Logistic Regression and Random Forest with 200 estimators
  b. Logistic Regression is faster to train and more explanatory

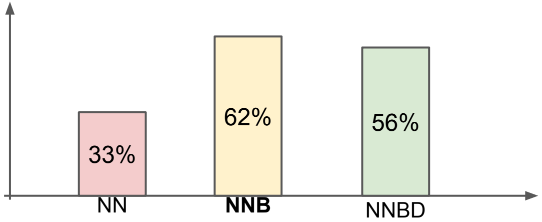
- Next steps:
  a. Repeat the experiment with more features
  b. Data augmentation technique to see if the results of Xgboost increase

- *Remark: we are showing the mean results from the CV and selecting the best results for each model type*

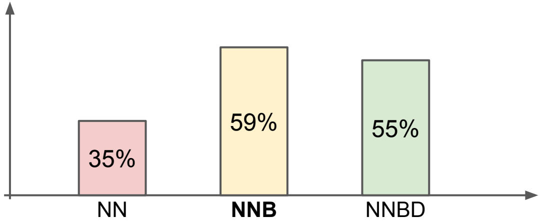
Deep Learning Approach - CV results

Main Metrics

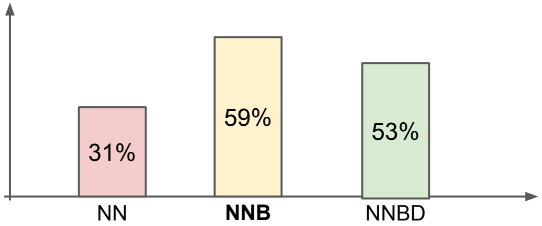
- Precision



- Recall



- F1



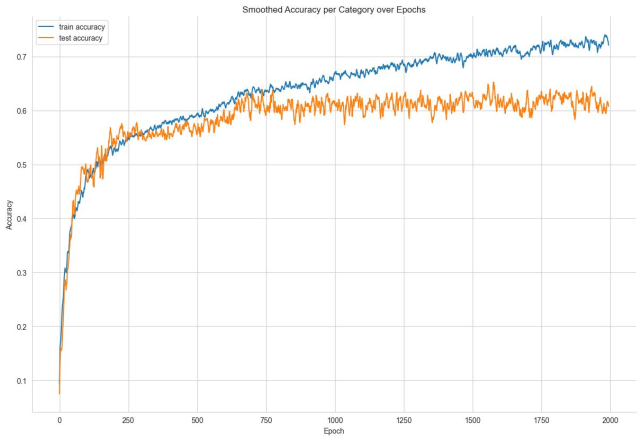
- Model tested:
  a. NN with 3 layers (hidden size: 50, 100, 200)
  b. NN with 3 layers + Batch Norm (hidden size: 50, 100, 200)
  c. NN with 3 layers + Batch Norm + Dropout (hidden size: 50, 100, 200)

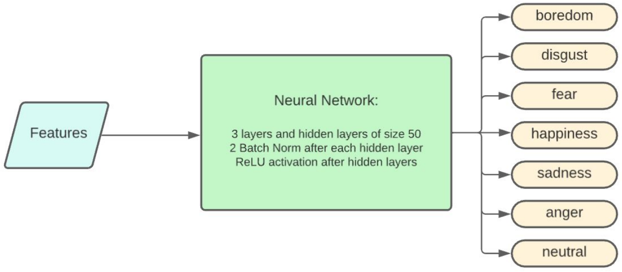
- Validation Technique:
  a. Cross Validation by user.
  b. The train set has 8 users then we iterate 8 times

- Main Results:
  a. Best results for Neural Network with Batch Norm and hidden size 50
  b. The training time of the best model during CV and 2000 epochs is 30s

- Next steps:
  a. Expand the features and try other architectures like CNN or RNN
  b. Data Augmentation

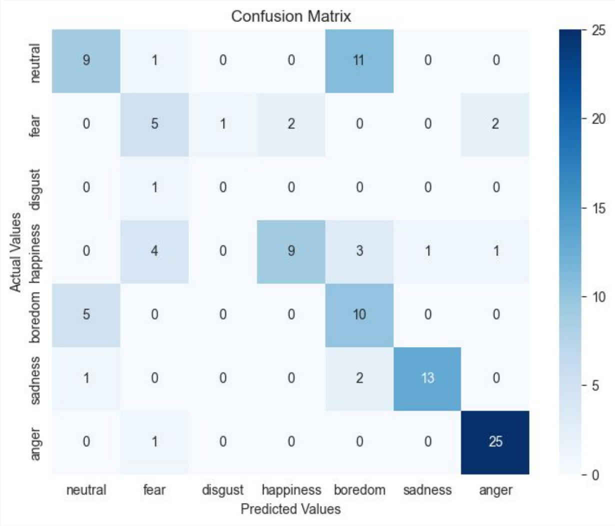
Best Model





Main Results

- Model training lasts 39 seconds
- The model classifies 6604 samples per second
- The model detects well **Angry** and struggles a lot with **Disgust** samples, it also confounds neutral with sadness.
- There are very few big mistakes (ex: misclassifying angry by happy)
- In this PoC we putted the same weight for each class it would be great in the future to know what are the most important class to detect.



API

Instructions Build Docker

1. Go into your terminal
2. Enter in the directory of the project
3. Make sure docker is running
4. Write the command: `docker-compose build`
5. You should see the following logs
6. Write the command: `docker-compose up`
7. You should see the following logs:
   a. Address of the notebook
   b. Address flask API

```
(speech_recognition) (miniforge3-4.10.3-10) alejandrobonell@Alejandros-MacBook-Air visium_homework % docker-compose build
[+] Building 0.8s (16/16) FINISHED
 => [internal] load build definition from Dockerfile                                         0.0s
 => => transferring dockerfile: 32B                                                          0.0s
 => [internal] load .dockerignore                                                            0.0s
 => => transferring context: 2B                                                              0.0s
 => [internal] load metadata for docker.io/library/python:3.9.16-slim-bullseye               0.6s
 => [ 1/11] FROM docker.io/library/python:3.9.16-slim-bullseye@sha256:5cde4e147c4165ad8dbf8a4df9  0.0s
 => [internal] load build context                                                            0.0s
 => => transferring context: 3.31kB                                                          0.0s
 => CACHED [ 2/11] RUN pip install --no-cache-dir pip==22.2.2                                0.0s
 => CACHED [ 3/11] WORKDIR /home                                                             0.0s
 => CACHED [ 4/11] RUN apt-get update && apt-get install -y gcc python3-dev                  0.0s
 => CACHED [ 5/11] RUN apt-get update -y && apt-get install -y --no-install-recommends build-ess  0.0s
 => CACHED [ 6/11] COPY requirements.txt .                                                   0.0s
 => CACHED [ 7/11] RUN pip install --upgrade pip &&     pip install --no-cache-dir -r requiremen  0.0s
 => CACHED [ 8/11] COPY src src/                                                             0.0s
 => CACHED [ 9/11] COPY img img/                                                             0.0s
 => CACHED [10/11] COPY Report_SER.ipynb .                                                   0.0s
 => [11/11] COPY application.py .                                                            0.0s
 => exporting to image                                                                       0.0s
 => => exporting layers                                                                      0.0s
 => => writing image sha256:590cb67b5636ba532e9a00e465dec60166b6385a64f45b791bbeb6689fb831e2  0.0s
 => => naming to docker.io/library/speech-emotion-recognition                                0.0s
(speech_recognition) (miniforge3-4.10.3-10) alejandrobonell@Alejandros-MacBook-Air visium_homework % docker-compose up
[+] Running 3/3
 ⠿ Container visium_homework-jupyter_notebook-1            Recreated                         0.1s
 ⠿ Container visium_homework-flask_api-1                   Recreated                         0.1s
 ⠿ Container visium_homework-speech-emotion-recognition-1  Recreated                         0.1s
Attaching to visium_homework-flask_api-1, visium_homework-jupyter_notebook-1, visium_homework-speech-emotion-recognition-1
visium_homework-speech-emotion-recognition-1 exited with code 0
visium_homework-jupyter_notebook-1          | [I 20:31:26.783 NotebookApp] Writing notebook server cookie secret to /root/.local/share/jupyter/runtime/notebook_cookie_secret
visium_homework-jupyter_notebook-1          | [I 20:31:27.052 NotebookApp] Serving notebooks from local directory: /home
visium_homework-jupyter_notebook-1          | [I 20:31:27.052 NotebookApp] Jupyter Notebook 6.5.4 is running at:
visium_homework-jupyter_notebook-1          | [I 20:31:27.052 NotebookApp] http://11b6a31cec83:5001/?token=...
visium_homework-jupyter_notebook-1          | [I 20:31:27.053 NotebookApp]  or http://127.0.0.1:5001/?token=...
visium_homework-jupyter_notebook-1          | [I 20:31:27.053 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
visium_homework-flask_api-1                 |  * Serving Flask app 'application'
visium_homework-flask_api-1                 |  * Debug mode: off
visium_homework-flask_api-1                 | WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
visium_homework-flask_api-1                 |  * Running on all addresses (0.0.0.0)
visium_homework-flask_api-1                 |  * Running on http://127.0.0.1:5000
visium_homework-flask_api-1                 |  * Running on http://172.21.0.4:5000
visium_homework-flask_api-1                 | Press CTRL+C to quit
visium_homework-flask_api-1                 | 172.21.0.1 - - [15/Jun/2023 20:33:00] "GET /train HTTP/1.1" 200 -
```

Flask API

- For training the model send the following request:

```
alejandrobonell@Alejandros-MacBook-Air visium_homework % curl --request GET --url 'http://0.0.0.0:5000/train'
model trained with final acc in train: 0.705607476635514 and in test: 0.6074766355140186%
```

- For predicting then send a request like:

```
curl --request
POST 'http://0.0.0.0:5000/pred' \
--header 'Content-Type: application/json' \
-d '{"id":"03a01Fa" }'
```

```
alejandrobonell@Alejandros-MacBook-Air visium_homework % curl --request POST 'http://0.0.0.0:5000/pred' \
--header 'Content-Type: application/json' \
-d '{
    "id":"03a01Fa" }'
{"class":{"code":"N","emotion":"neutral","label":0}}
alejandrobonell@Alejandros-MacBook-Air visium_homework %
```

Jupyter Notebook

1. To access the notebook go into your browser to the URL: http://0.0.0.0:5001/
2. Then you will be redirected to a website and ask you to put a password, write: visiumSER

Conclusions and Next steps

Conclusions:

1. Created a functional model with 64% of accuracy (Random gives 14.25% of Accuracy)

2. The final model has been trained on 490 samples

3. The model **detects** very well **sadness** and **anger** and **struggles** with **disgust** and **boredom**

4. The model classifies **6604** samples per second

Next Steps:

1. Understand what are the most important classes to detect to get better results on those.

2. Try more complex features and architectures, we can add all the information from the MFCC and then use a CNN

3. Data augmentation:
    a. Other public datasets
    b. Label data using active learning strategies
    c. Soft labelling