

Rapport du Mini Projet Nouvelles Architectures

Réalisé par : Maleke Khemiri et Mayssa Bejaoui

Architecture

Le projet est constitué de

app.py : Un fichier Python qui comprend les diverses importations ainsi que les contrôleurs utilisés pour chaque requête HTTP.

DockerFile pour le Front et un **DockerFile** pour le Back

Svm_service.py : qui définit notre service du modèle SVM

Vgg_service.py : qui définit notre service du modèle VGG19

Svm_model : qui définit notre modèle SVM

Vgg_model : qui définit notre modèle VGG19

Dossier static : qui contient un fichier **styles.css** et dossier **photos**

Dossier templates : qui contient un fichier **index.html**

Modèle SVM :

La figure 1 montre les bibliothèques utilisées pour notre modèle SVM.



```
[1] In [5]: import librosa.display
import IPython.display as ipd
from scipy.io import wavfile as wav
import pandas as pd
import os
import numpy as np
import seaborn as sns

from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import SGDClassifier, LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier
from xgboost import XGBClassifier, XGBRFClassifier

from sklearn.metrics import confusion_matrix, accuracy_score, roc_auc_score, roc_curve
from sklearn import preprocessing
from sklearn.model_selection import train_test_split

from tensorflow.keras.callbacks import ReduceLROnPlateau, EarlyStopping, ModelCheckpoint, LearningRateScheduler
import tensorflow.keras as keras
from tensorflow.keras import Sequential
from tensorflow.keras.layers import *
```

```
[6] In [6]: from google.colab import drive
drive.mount('/content/drive', force_remount=True)

# Spécifiez le chemin du fichier dans votre Google Drive
path = '/content/drive/My Drive/BD/Data'
print(list(os.listdir(f'{path}/genres_original'))))

Mounted at /content/drive
['blues', 'classical', 'reggae', 'jazz', 'hiphop', 'rock', 'disco', 'metal', 'pop', 'country']
```

Après l'entraînement de notre modèle SVM, un fichier **svm_model.pkl** a été généré



```
svm_model.pkl ((6993, 58), (2097, 58), (6993, ), (2097, ))

[23] In [23]: def model_assess(model, title = "Default"):
model.fit(X_train, y_train)
preds = model.predict(X_test)
#print(confusion_matrix(y_test, preds))
print('Accuracy', title, ':', round(accuracy_score(y_test, preds), 5), '\n')

[24] In [24]: # Support Vector Machine
svm = SVC(decision_function_shape="ovo")
model_assess(svm, "Support Vector Machine")

Accuracy Support Vector Machine : 0.74775

[25] In [25]: import joblib
joblib.dump(svm, 'svm_model.pkl')

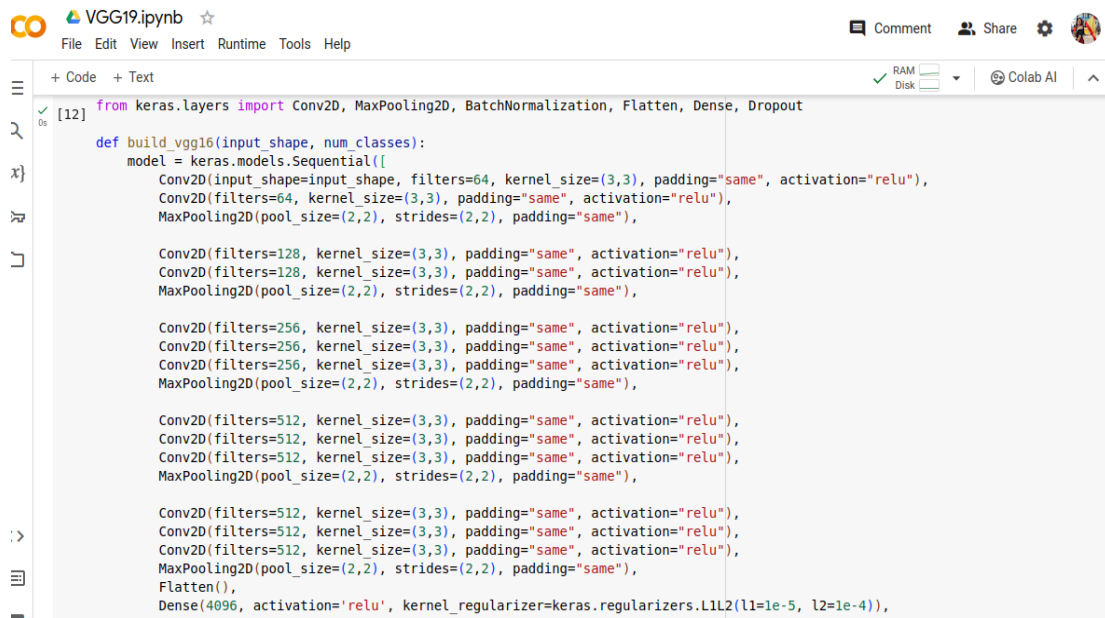
['svm_model.pkl']

[26] In [26]: # Charger le modèle depuis le fichier sauvegardé
loaded_model = joblib.load('svm_model.pkl')

[27] In [27]: from google.colab import drive
drive.mount('/content/drive')

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

La figure ci-dessous montre une partie du code de notre modèle VGG19



The screenshot shows a Jupyter Notebook titled 'VGG19.ipynb'. The code defines a function `build_vgg16` that creates a VGG16 model using Keras. The model is a `Sequential` stack of layers: two initial `Conv2D` layers (64 filters, kernel size 3x3), followed by a `MaxPooling2D` layer (pool size 2x2), then two more `Conv2D` layers (128 filters, kernel size 3x3), another `MaxPooling2D` layer, and finally three `Conv2D` layers (256, 512, and 512 filters, kernel size 3x3) followed by a final `MaxPooling2D` layer. The output is flattened and passed through a `Dense` layer with 4096 units. The model is compiled with the 'adam' optimizer and 'categorical_crossentropy' loss.

```
[12] from keras.layers import Conv2D, MaxPooling2D, BatchNormalization, Flatten, Dense, Dropout

def build_vgg16(input_shape, num_classes):
    model = keras.models.Sequential([
        Conv2D(input_shape=input_shape, filters=64, kernel_size=(3,3), padding="same", activation="relu"),
        Conv2D(filters=64, kernel_size=(3,3), padding="same", activation="relu"),
        MaxPooling2D(pool_size=(2,2), strides=(2,2), padding="same"),

        Conv2D(filters=128, kernel_size=(3,3), padding="same", activation="relu"),
        Conv2D(filters=128, kernel_size=(3,3), padding="same", activation="relu"),
        MaxPooling2D(pool_size=(2,2), strides=(2,2), padding="same"),

        Conv2D(filters=256, kernel_size=(3,3), padding="same", activation="relu"),
        Conv2D(filters=256, kernel_size=(3,3), padding="same", activation="relu"),
        Conv2D(filters=256, kernel_size=(3,3), padding="same", activation="relu"),
        MaxPooling2D(pool_size=(2,2), strides=(2,2), padding="same"),

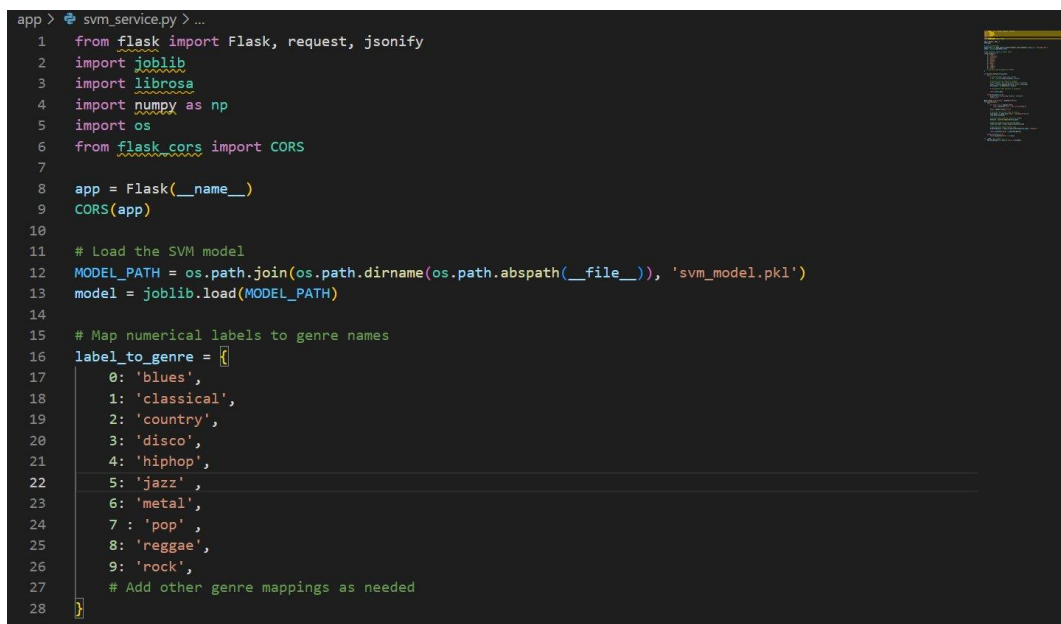
        Conv2D(filters=512, kernel_size=(3,3), padding="same", activation="relu"),
        Conv2D(filters=512, kernel_size=(3,3), padding="same", activation="relu"),
        Conv2D(filters=512, kernel_size=(3,3), padding="same", activation="relu"),
        MaxPooling2D(pool_size=(2,2), strides=(2,2), padding="same"),

        Conv2D(filters=512, kernel_size=(3,3), padding="same", activation="relu"),
        Conv2D(filters=512, kernel_size=(3,3), padding="same", activation="relu"),
        Conv2D(filters=512, kernel_size=(3,3), padding="same", activation="relu"),
        MaxPooling2D(pool_size=(2,2), strides=(2,2), padding="same"),
        Flatten(),
        Dense(4096, activation='relu', kernel_regularizer=keras.regularizers.L1L2(l1=1e-5, l2=1e-4)),
        Dense(num_classes, activation='softmax')
    ])
    model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
    return model
```

Après avoir généré nos modèles SVM et VGG19, nous passons à la création des services pour ces modèles.

Les figures ci-dessous montrent le contenu du fichier `svm_service.py` et `vgg_service.py` en utilisant Flask.

Service SVM



The screenshot shows a Python script for an SVM service using Flask. It imports necessary libraries like `flask`, `joblib`, `librosa`, `numpy`, `os`, and `flask_cors`. It initializes a Flask app with CORS support. The script loads an SVM model from a file named `svm_model.pkl`. It also defines a mapping from numerical labels to genre names: 0: 'blues', 1: 'classical', 2: 'country', 3: 'disco', 4: 'hiphop', 5: 'jazz', 6: 'metal', 7: 'pop', 8: 'reggae', 9: 'rock'. A comment indicates that other genre mappings can be added as needed.

```
app > svm_service.py > ...
1 from flask import Flask, request, jsonify
2 import joblib
3 import librosa
4 import numpy as np
5 import os
6 from flask_cors import CORS
7
8 app = Flask(__name__)
9 CORS(app)
10
11 # Load the SVM model
12 MODEL_PATH = os.path.join(os.path.dirname(os.path.abspath(__file__)), 'svm_model.pkl')
13 model = joblib.load(MODEL_PATH)
14
15 # Map numerical labels to genre names
16 label_to_genre = {
17     0: 'blues',
18     1: 'classical',
19     2: 'country',
20     3: 'disco',
21     4: 'hiphop',
22     5: 'jazz',
23     6: 'metal',
24     7: 'pop',
25     8: 'reggae',
26     9: 'rock',
27     # Add other genre mappings as needed
28 }
```

```

49 def svm_service():
50     try:
51         if 'file' not in request.files:
52             return jsonify({'error': 'No file provided'})
53
54         file = request.files['file']
55
56         # Save the file to the 'temp' directory
57         file_path = os.path.join('temp', 'uploaded_file.wav')
58         file.save(file_path)
59
60         # Extract audio features (modify as needed)
61         features = extract_features(file_path)
62
63         # Make the prediction with the SVM model
64         numerical_label = model.predict([features])[0]
65
66         # Map numerical label to genre name
67         predicted_genre = label_to_genre.get(numerical_label, 'Unknown')
68
69         return jsonify({'genre': predicted_genre})
70
71     except Exception as e:
72         return jsonify({'error': str(e)})
73
74 if __name__ == '__main__':
75     app.run(debug=True, host='0.0.0.0', port=5001)
76

```

```

30 def extract_features(file_path):
31     try:
32         # Load the audio file using librosa
33         y, sr = librosa.load(file_path, sr=None)
34
35         # Extract features (modify as needed)
36         # For example, let's extract the mean of the MFCCs
37         mfccs = librosa.feature.mfcc(y=y, sr=sr, n_mfcc=58)
38         mfccs_mean = np.mean(mfccs, axis=1)
39
40         # Concatenate other features if necessary
41
42         return mfccs_mean
43
44     except Exception as e:
45         print(f"Error extracting features: {str(e)}")
46         return None
47
48 @app.route('/svm_service', methods=['POST'])
49 def svm_service():
50     try:
51         if 'file' not in request.files:
52             return jsonify({'error': 'No file provided'})
53
54         file = request.files['file']
55
56         # Save the file to the 'temp' directory
57         file_path = os.path.join('temp', 'uploaded_file.wav')
58         file.save(file_path)

```

Service VGG19:

```

app > vgg_service.py > ...
1  # vgg19_service.py
2
3  from flask import Flask, request, jsonify
4  import tensorflow as tf
5  from tensorflow.keras.preprocessing import image
6  import numpy as np
7  from flask_cors import CORS
8
9
10 app = Flask(__name__)
11 CORS(app)
12
13 # Map numerical labels to genre names
14 label_to_genre = {
15     0: 'blues',
16     1: 'classical',
17     2: 'country',
18     3: 'disco',
19     4: 'hiphop',
20     5: 'jazz',
21     6: 'metal',
22     7: 'pop',
23     8: 'reggae',
24     9: 'rock',
25     # Add other genre mappings as needed
26 }
27 # Load the saved VGG19 model
28
29 model_vgg19 = tf.saved_model.load('vgg_model.pb')

```

```

28
29 model_vgg19 = tf.saved_model.load('vgg_model.pb')
30
31 def preprocess_image(img_path):
32     img = image.load_img(img_path, target_size=(224, 224))
33     img_array = image.img_to_array(img)
34     img_array = np.expand_dims(img_array, axis=0)
35     img_array = tf.keras.applications.vgg19.preprocess_input(img_array)
36     return img_array
37
38 @app.route('/classify_vgg19', methods=['POST'])
39 def classify_vgg19():
40     if 'file' not in request.files:
41         return jsonify({'error': 'No file provided'})
42
43     file = request.files['file']
44     file_path = '/tmp/uploaded_image.jpg'
45     file.save(file_path)
46
47     # Preprocess the image
48     img_array = preprocess_image(file_path)
49
50     # Make prediction using the loaded model
51     predictions = model_vgg19(img_array)
52     decoded_predictions = tf.keras.applications.vgg19.decode_predictions(predictions.numpy(), top=3)[0]
53
54     # Return predictions as JSON
55     result = [{'label': label, 'probability': float(prob)} for (_, label, prob) in decoded_predictions]
56     return jsonify({'predictions': result})

```

Après avoir créé nos services, nous écrivons un DockerFile afin de créer un conteneur Docker qui contient `svm_service.py` et `vgg_service.py`

DockerFile Front

```

1 FROM python:3.8
2
3 WORKDIR /app
4
5 COPY ./app/svm_service.py /app/
6 COPY ./app/vgg_service.py /app/
7 COPY ./app/svm_model.pkl /app/
8 COPY ./app/vgg_model.pb /app/
9
10 RUN pip install Flask flask-cors scikit-learn joblib librosa tensorflow
11
12 EXPOSE 5001 5002
13
14 CMD ["python", "svm_service.py", "vgg_service.py"]

```

La figure ci-dessous montre la création de notre conteneur avec **docker build**.

[illegible]

L'activation de notre conteneur avec **docker run**.

```
PS C:\Users\maiss\OneDrive\Bureau\projetDevOps\app> docker run -p 5001:5001 -p 5002:5002 back1:latest
* Serving Flask app 'svm_service'
* Debug mode: on
/usr/local/lib/python3.8/site-packages/sklearn/base.py:348: InconsistentVersionWarning: Trying to unpickle estimator SVC from version 1.2.2 when using version 1.3.2. This might lead to breaking code or invalid results. Use at your own risk. For more info please refer to:
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
warnings.warn(
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5001
* Running on http://172.17.0.2:5001
```

L'étape suivante consiste à créer notre page web, un code en HTML et en CSS sera écrit pour notre interface web.

Les deux figures ci-dessous montrent une partie du contenu du fichier **index.html**

```

app > templates > index.html > html > head > script > predictImage
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4    <meta charset="UTF-8">
5    <meta name="viewport" content="width=device-width, initial-scale=1.0">
6    <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
7    <script src="https://code.jquery.com/jquery-3.6.4.min.js"></script>
8    <script>
9      function predictAudio() {
10        // Get the selected audio file
11        var audioInput = $("#audioInput")[0].files[0];
12
13        if (audioInput) {
14          // Create a FormData object to send the file
15          var formData = new FormData();
16          formData.append("file", audioInput);
17
18          // Make an AJAX request to the svm_service endpoint
19          $.ajax({
20            type: "POST",
21            url: "http://localhost:5001/svm_service",
22            data: formData,
23            contentType: false,
24            processData: false,
25            success: function(response) {
26              // Update the content with the predicted genre
27              console.log(response); // Log the entire response object
28              $("#r").text("Predicted Genre: " + response.genre);
29
30              // Navigate to the result section
31              window.location.href = "#result";
32            },
33            error: function(error) {
34              console.error("Error predicting genre:", error.responseText);
35              $("#r").text("Error predicting genre. Check console for details.");
36            }
37          });
38        } else {
39          alert("Please choose an audio file.");
40        }
41      }
42      function predictImage() {
43        var imageInput = $("#imageInput")[0].files[0];
44
45        if (imageInput) {
46          var formData = new FormData();
47          formData.append("file", imageInput);
48
49          $.ajax({
50            type: "POST",
51            url: "http://localhost:5002/classify_vgg19", // Assuming your VGG19 service is running on
52            data: formData,
53            contentType: false,
54            processData: false,
55            success: function(response){
56              console.log(response);
57              $("#imageResult").text("Predicted Genre: " + response.genre);
58              window.location.href = "#image"; // Scroll to the image result section

```

Front CSS


```

app > static > # style.css > {} @media only screen and (min-width: 768px) > 🚧 h1.section-title
1  @import 'https://fonts.googleapis.com/css?family=Montserrat:300, 400, 700&display=swap';
2  * {
3      padding: 0;
4      margin: 0;
5      box-sizing: border-box;
6  }
7  html {
8      font-size: 10px;
9      font-family: 'Montserrat', sans-serif;
10     scroll-behavior: smooth;
11 }
12 a {
13     text-decoration: none;
14 }
15 .container {
16     min-height: 100vh;
17     width: 100%;
18     display: flex;
19     align-items: center;
20 }
21 img {
22     height: 100%;
23     width: 100%;
24     object-fit: cover;
25 }
26 p {
27     color: black;
28     font-size: 1.4rem;
29     margin-top: 5px;
30     line-height: 2.5rem;

```

```

app > static > # style.css > {} @media only screen and (min-width: 768px) > 🚧 h1.section-title
30     line-height: 2.5rem;
31     font-weight: 300;
32     letter-spacing: 0.05rem;
33 }
34 .section-title {
35     font-size: 4rem;
36     font-weight: 300;
37     color: black;
38     margin-bottom: 10px;
39     text-transform: uppercase;
40     letter-spacing: 0.2rem;
41     text-align: center;
42 }
43 .section-title span {
44     color: crimson;
45 }
46
47 .cta {
48     display: inline-block;
49     padding: 10px 30px;
50     color: white;
51     background-color: transparent;
52     border: 2px solid crimson;
53     font-size: 2rem;
54     text-transform: uppercase;
55     letter-spacing: 0.1rem;
56     margin-top: 30px;
57     transition: 0.3s ease;
58     transition-property: background-color, color;

```

DockerFile Front

L'étape suivante est la création du DockerFile pour notre partie Front


```

app > Dockerfile > CMD
1  FROM nginx:alpine
2
3  RUN rm -rf /usr/share/nginx/html/*
4
5  COPY ./static /usr/share/nginx/html/static
6
7  COPY ./templates/index.html /usr/share/nginx/html/index.html
8
9  EXPOSE 80
10
11 CMD ["nginx", "-g", "daemon off;"]
12

```

La figure ci-dessous montre la création de notre conteneur avec **docker build**.

```

PROBLEMS 10 OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\maiss\OneDrive\Bureau\projetDevOps\app> docker build -t front3:latest .
[+] Building 1.4s (9/9) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 259B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/nginx:alpine
=> [1/4] FROM docker.io/library/nginx:alpine@sha256:3923f8de8d2214b9490e68fd6ae63ea604deddd166df2755b788bef04848b9bc
=> [internal] load build context
=> => transferring context: 297B
=> CACHED [2/4] RUN rm -rf /usr/share/nginx/html/*
=> [3/4] COPY ./static/* /usr/share/nginx/html/static
=> [4/4] COPY ./templates/index.html /usr/share/nginx/html/index.html
=> exporting to image
=> => exporting layers
=> => writing image sha256:29ec3033b9b8c9b94a8b0b00d812b705f068f0a2680c0fc721d43455736948f3
=> => naming to docker.io/library/front3:latest

What's Next?
View a summary of image vulnerabilities and recommendations -> docker scout quickview

```

L'activation de notre conteneur avec **docker run**.

```

PS C:\Users\maiss\OneDrive\Bureau\projetDevOps\app> docker run -p 8080:80 --name container3_front front3:latest
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2023/12/13 21:49:17 [notice] 1#1: using the "epoll" event method
2023/12/13 21:49:17 [notice] 1#1: nginx/1.25.3
2023/12/13 21:49:17 [notice] 1#1: built by gcc 12.2.1 20220924 (Alpine 12.2.1_git20220924-r10)
2023/12/13 21:49:17 [notice] 1#1: OS: Linux 5.15.133.1-microsoft-standard-WSL2
2023/12/13 21:49:17 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2023/12/13 21:49:17 [notice] 1#1: start worker processes
2023/12/13 21:49:17 [notice] 1#1: start worker process 30
2023/12/13 21:49:17 [notice] 1#1: start worker process 31
2023/12/13 21:49:17 [notice] 1#1: start worker process 32

```

Docker-Compose

Afin de gérer notre application multi-conteneur. On crée un fichier YAML pour configurer les services afin de faciliter le déploiement et la gestion des conteneurs.

```

🔥 docker-compose.yml
1  version: '3'
2  services:
3    frontend:
4      build:
5        context: ./app
6        dockerfile: Dockerfile
7      ports:
8        - "80:80"
9      volumes:
10     - ./app:/app
11   backend:
12     build:
13       context: ./
14       dockerfile: Dockerfile
15     ports:
16       - "5001:5001"
17       - "5002:5002"
18     volumes:
19       - ./app:/app
20

```

On démarre les conteneurs grâce à **docker-compose up**

```

PS C:\Users\maiss\OneDrive\Bureau\projetDevOps\app> docker-compose up -d
[+] Building 4.4s (21/21) FINISHED
=> [frontend internal] load build definition from Dockerfile
=> => transferring dockerfile: 259B
=> [frontend internal] load .dockerignore
=> => transferring context: 2B
=> [backend internal] load .dockerignore
=> => transferring context: 2B
=> [backend internal] load build definition from Dockerfile
=> => transferring dockerfile: 352B
=> [backend internal] load metadata for docker.io/library/python:3.8
=> [frontend internal] load metadata for docker.io/library/nginx:alpine
=> [backend internal] load build context
=> => transferring context: 5.04MB
=> [backend 1/7] FROM docker.io/library/python:3.8@sha256:7264a50439679c2868a99f71a5c9b9831cc082b1d3f05c8643788e1
=> [frontend 1/4] FROM docker.io/library/nginx:alpine@sha256:3923f8de8d2214b9490e68fd6ae63ea604deddd166df2755b788
=> [frontend internal] load build context
=> => transferring context: 2.18MB
=> CACHED [frontend 2/4] RUN rm -rf /usr/share/nginx/html/*
=> CACHED [frontend 3/4] COPY ./static/* /usr/share/nginx/html/static
=> [frontend 4/4] COPY ./templates/index.html /usr/share/nginx/html/index.html
=> [frontend] exporting to image
=> => exporting layers
=> => writing image sha256:910b094b95b6268285f8e5ba348f1fd6863fe606c99b9795272418ea8e07ccd2
=> => naming to docker.io/library/app-frontend
=> CACHED [backend 2/7] WORKDIR /app
=> CACHED [backend 3/7] COPY ./app/svm_service.py /app/
=> CACHED [backend 4/7] COPY ./app/vgg_service.py /app/
=> CACHED [backend 5/7] COPY ./app/svm_model.pkl /app/
=> CACHED [backend 6/7] COPY ./app/vgg_model.pb /app/
=> CACHED [backend 7/7] RUN pip install Flask flask-cors scikit-learn joblib librosa tensorflow
=> [backend] exporting to image
=> => exporting layers
=> => writing image sha256:4e39ba6631f7598af8989cc49a3c26113decc738e7dcc6944642deabea987a79
=> => naming to docker.io/library/app-backend
[+] Running 3/3
✔ Network app_default      Created
✔ Container app-backend-1  Started
✔ Container app-frontend-1 Created
Error response from daemon: Ports are not available: exposing port TCP 0.0.0.0:80 -> 0.0.0.0:0: listen tcp 0.0.0.0:80:
attempt was made to access a socket in a way forbidden by its access permissions.
PS C:\Users\maiss\OneDrive\Bureau\projetDevOps\app>

```

Notre interface web permet de prédire le genre musical à travers un son ou une image.



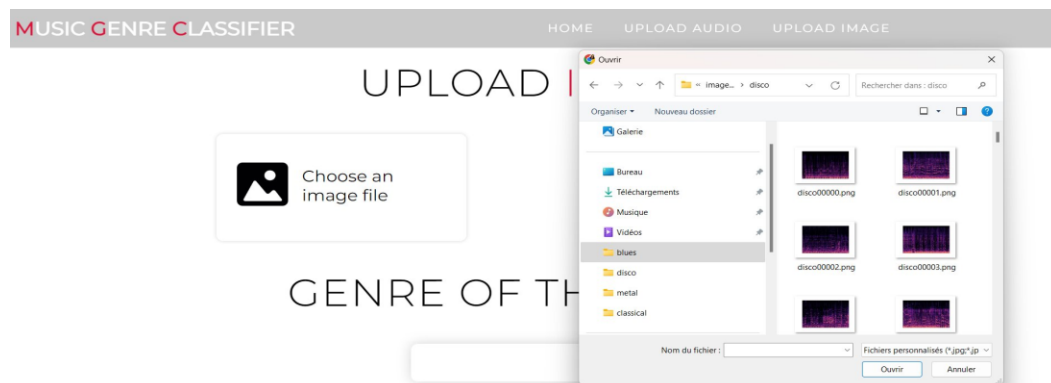
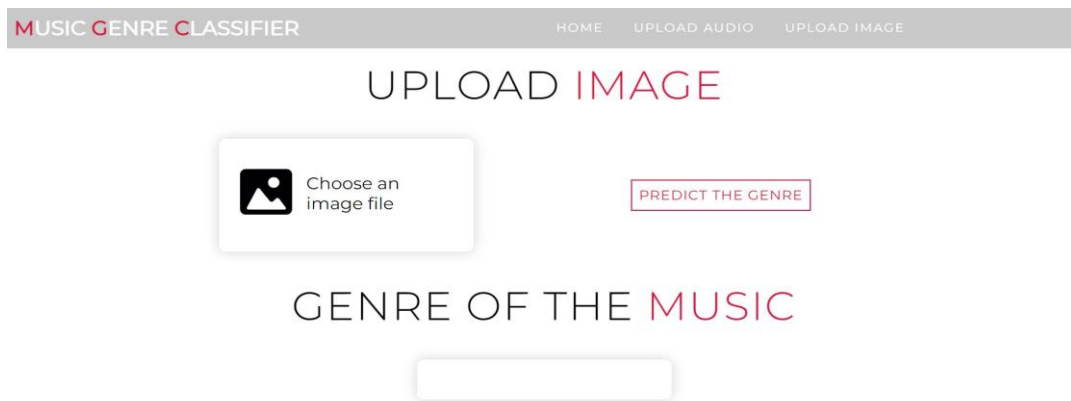
Il suffit de télécharger le son.

A screenshot of the 'MUSIC GENRE CLASSIFIER' website's 'UPLOAD MUSIC' section. The header includes the site name and navigation links. The main heading is 'UPLOAD MUSIC'. Below it, there is a box with a headphones icon and the text 'Choose an audio file'. To the right is a button labeled 'PREDICT THE GENRE'. Underneath, the heading 'GENRE OF THE MUSIC' is followed by an empty text input field.

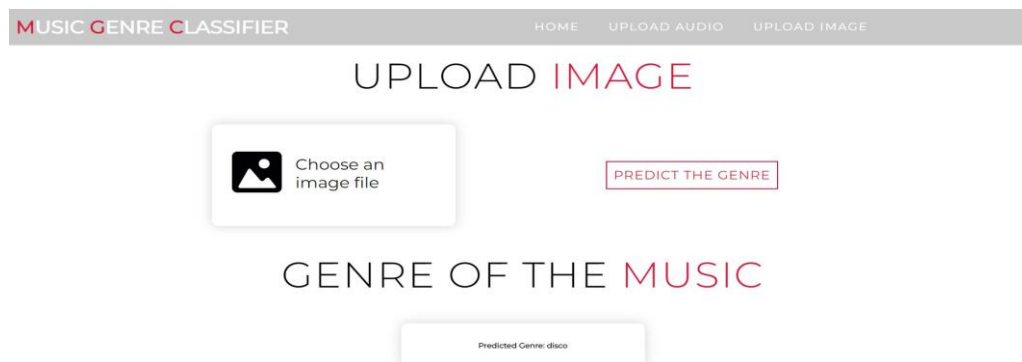
Le résultat sera affiché comme suit.

A screenshot of the 'MUSIC GENRE CLASSIFIER' website's 'UPLOAD MUSIC' section, showing the result of a prediction. The layout is identical to the previous screenshot, but the text input field under 'GENRE OF THE MUSIC' now displays 'Predicted Genre: classical'.

On peut aussi prédire en cliquant sur « choose an image file »





Le résultat sera affiché comme suit.





Pour la partie tests, nous avons choisi Jenkins comme plateforme, la figure ci-dessous montre les tests effectués.

+ Nouveau Item

 Utilisateurs

 Historique des constructions

 Administrer Jenkins

 Mes vues

File d'attente des constructions

▼

File d'attente des constructions vide

État du lanceur de compilations

▼










1 Au repos

2 Au repos

Tous

+

Ajouter une description

| S | M | Nom du projet | Dernier succès | Dernier échec | Dernière durée | |
|---|---|---------------|------------------------------|------------------------------|----------------|---|
|  |  | projetDocker | 4 mn 26 s #1 | s. o. | 0,38 s |  |
|  |  | projetDocker1 | 9 h 45 mn #1 | s. o. | 1 s |  |
|  |  | projetDocker2 | s. o. | 9 h 42 mn #5 | 23 ms |  |


Icône:


S


M

L

Légende

 Atom feed for all

 Atom feed for failures

 Atom feed for just latest builds