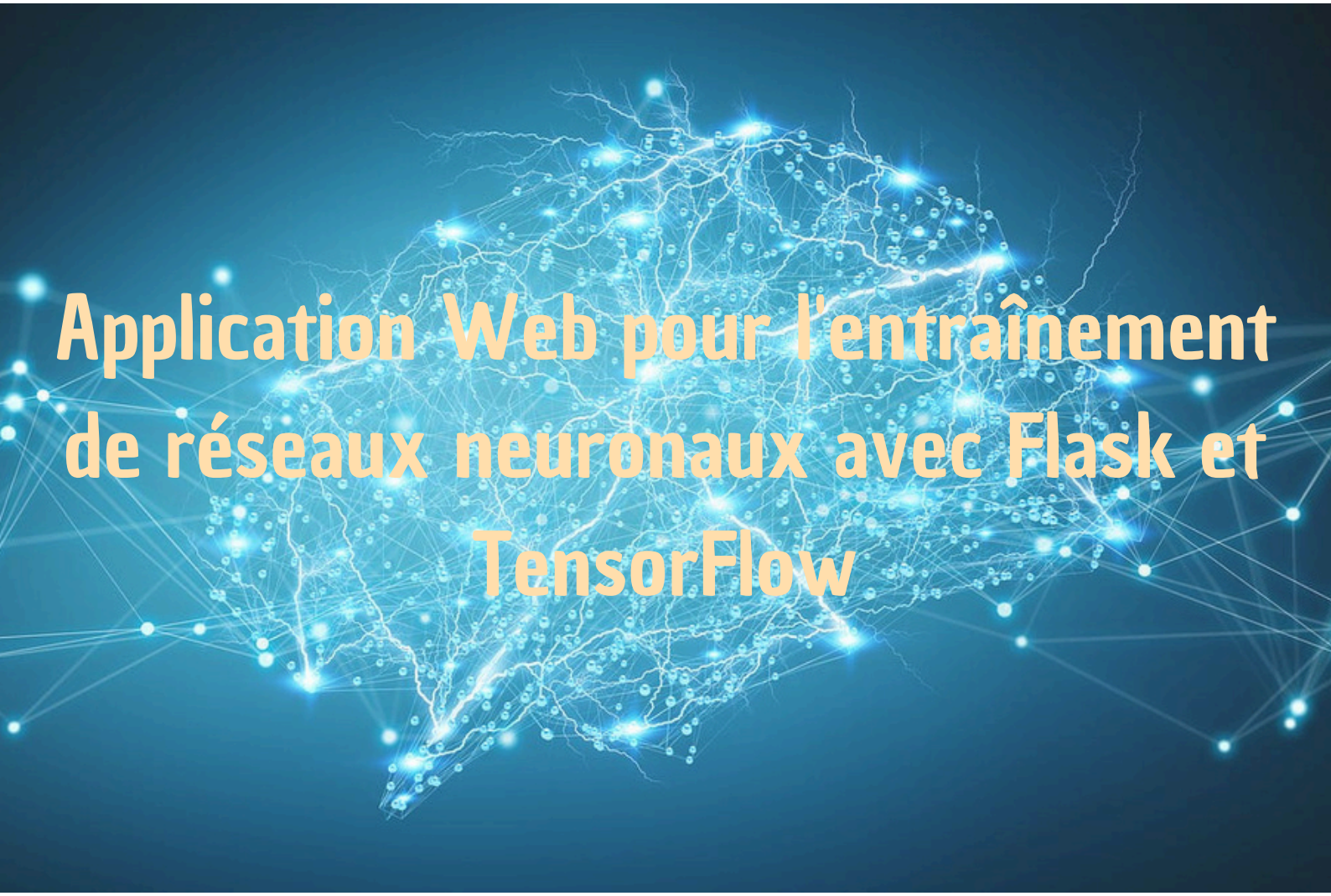
المدرسة الوطنية العليا للذكاء الاصطناعي وعلوم المعطيات - تارودانت
ÉCOLE NATIONALE SUPÉRIEURE DE L'INTELLIGENCE ARTIFICIELLE
ET SCIENCES DES DONNÉES - TAROUDANT



Application Web pour l'entraînement de réseaux neuronaux avec Flask et TensorFlow

réaliser par :

Hariri Chaimae

El Kassoiri Mohammed

IdAdoub ElKhalil

Encadré par :

Mr Kasri Mohammed

2 annee SDBDIA
2024/2025

sommaire

Introduction:

L'essor de l'intelligence artificielle et de l'apprentissage profond a profondément transformé de nombreux secteurs, allant de la reconnaissance d'image à la médecine prédictive. Au cœur de ces innovations se trouvent les réseaux de neurones, une architecture d'apprentissage automatique inspirée par le fonctionnement des neurones biologiques.

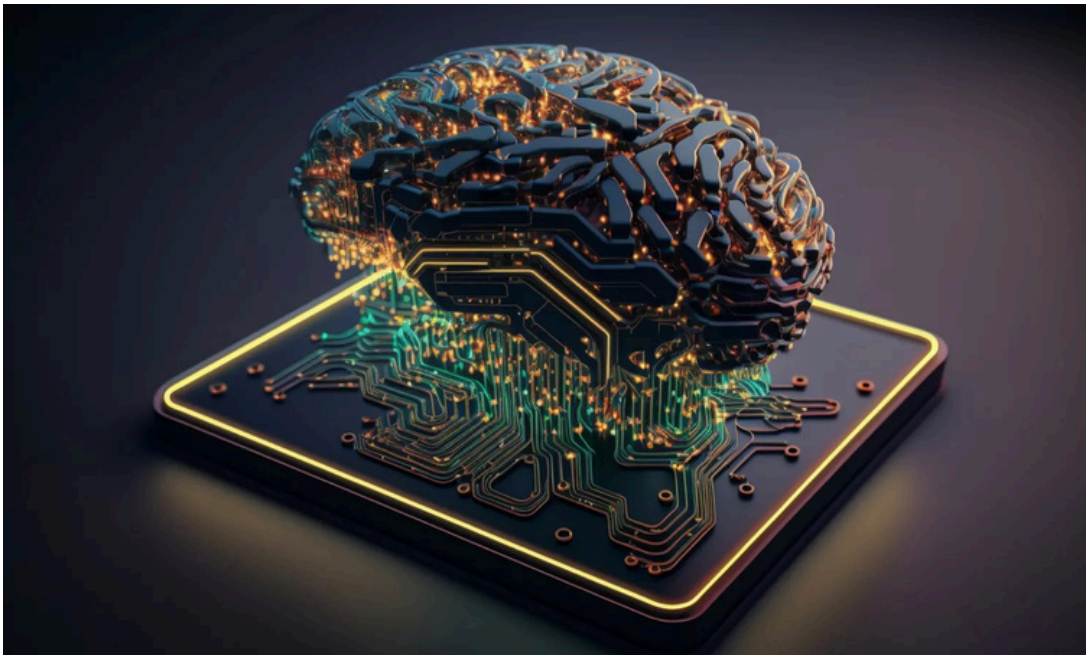
Cependant, l'entraînement et la mise en œuvre de réseaux de neurones restent souvent complexes et requièrent des compétences en programmation et en manipulation de données.

Le projet que nous présentons ici vise à simplifier le processus d'entraînement de réseaux de neurones en développant une application web intuitive. Cette application permet aux utilisateurs, même sans expertise avancée, de configurer et d'entraîner leurs propres modèles de réseaux de neurones.

En utilisant le framework Flask pour le développement du serveur web et TensorFlow pour la gestion du backend de l'apprentissage automatique, nous proposons un outil accessible, interactif et performant.

Objective de Projet :

- Faciliter l'entraînement de réseaux de neurones grâce à une interface conviviale où les utilisateurs peuvent charger leurs propres jeux de données et configurer l'architecture du réseau.
- Permettre la personnalisation des hyperparamètres du modèle, incluant le nombre de couches, les types de couches, le taux d'apprentissage, et bien plus.
- Visualiser les résultats en temps réel, notamment la courbe de perte et les performances du modèle en termes de précision.
- Entraînement de modèles sur des jeux de données divers Assurer que l'application peut gérer des jeux de données variés, allant des données tabulaires (CSV) aux images, afin de démontrer la flexibilité du modèle.



Architecture du system :

- Frontend : Interface utilisateur (HTML/CSS/JavaScript/Bootstrap)

1. Formulaire de chargement de données
2. Configuration des hyperparamètres
3. Affichage des métriques en temps réel

- Backend : Gestion des requêtes et entraînement avec TensorFlow

Flask : Ce framework léger a été choisi pour créer le serveur web

TensorFlow : Utilisé pour la création et l'entraînement des réseaux de neurones

- Communication Frontend-Backend

L'architecture suit une logique client-serveur où :

Le frontend envoie des requêtes HTTP au serveur Flask pour charger des fichiers, lancer l'entraînement ou récupérer les métriques.

Étapes du projet

Le développement de l'application s'est déroulé en plusieurs étapes clés, allant de la conception du frontend à l'intégration de l'entraînement du modèle de réseau de neurones, en fonction des données et paramètres fournis par l'utilisateur.es lignes dans le corps du texte

1. Développement du Frontend (HTML/CSS/JavaScript/Bootstrap)

La première étape a consisté à créer une interface utilisateur intuitive qui permet aux utilisateurs de configurer le réseau de neurones. Nous avons utilisé HTML, CSS et Bootstrap pour concevoir une interface responsive, et JavaScript pour ajouter des fonctionnalités interactives.

- **Formulaire de téléchargement de données** : Permettre aux utilisateurs de charger leurs propres jeux de données (fichiers CSV, images, etc.).

- **Formulaire de configuration du modèle** : L'utilisateur peut définir les caractéristiques du modèle, comme le nombre de couches, le type de couches (Dense, Convolutionnelle, etc.), le nombre de neurones, et d'autres hyperparamètres tels que le taux d'apprentissage.

- **Affichage dynamique** : Une fois les paramètres saisis, ils sont envoyés au backend pour le traitement, et les résultats sont visualisés sous forme de courbes et graphiques, notamment via Chart.js pour afficher la courbe de perte et la précision en temps réel.

2. Entraînement du modèle avec TensorFlow

- **Réception des données et paramètres** : Le backend, développé avec Flask, reçoit les données téléchargées et les paramètres du modèle (nombre de couches, type de couches, etc.).

- **Création du modèle dynamique avec TensorFlow** : En fonction des paramètres de configuration, un modèle de réseau de neurones est créé à la volée. Par exemple, si l'utilisateur a défini un réseau avec 3 couches denses, celles-ci sont générées selon les spécifications données.

- **Entraînement du modèle** : Après la création du modèle, le processus d'entraînement commence. Les données sont normalisées et prétraitées pour être compatibles avec TensorFlow.

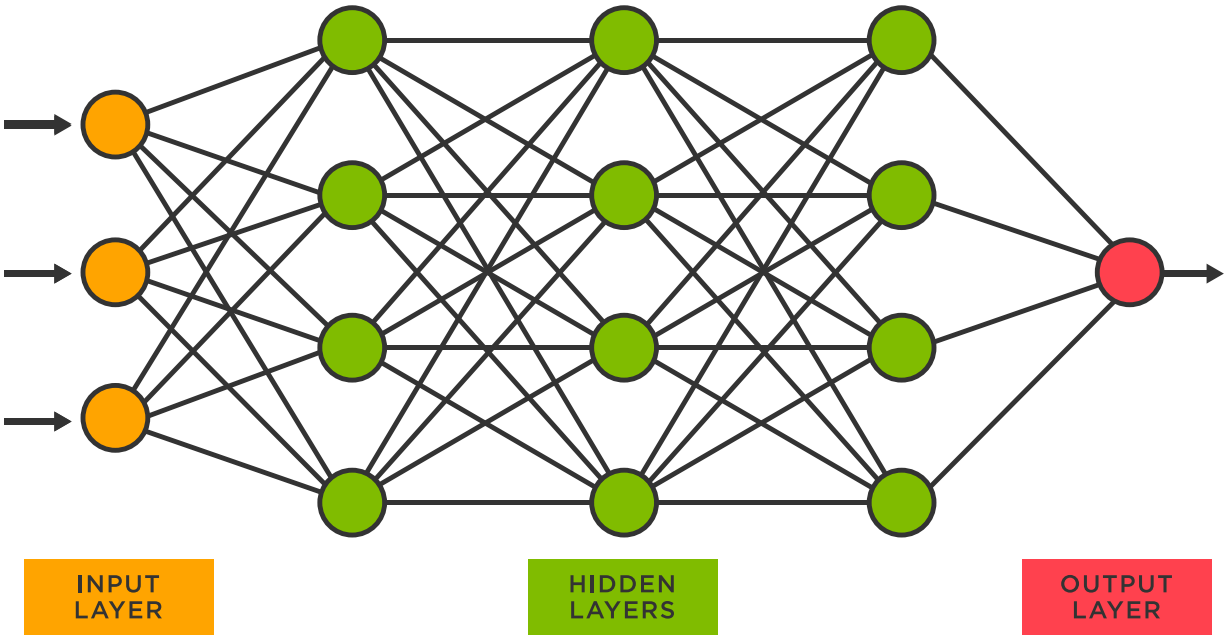
3. Visualisation des résultats

- **Courbe de perte** : Montre comment la perte diminue au fur et à mesure que le modèle s'entraîne.

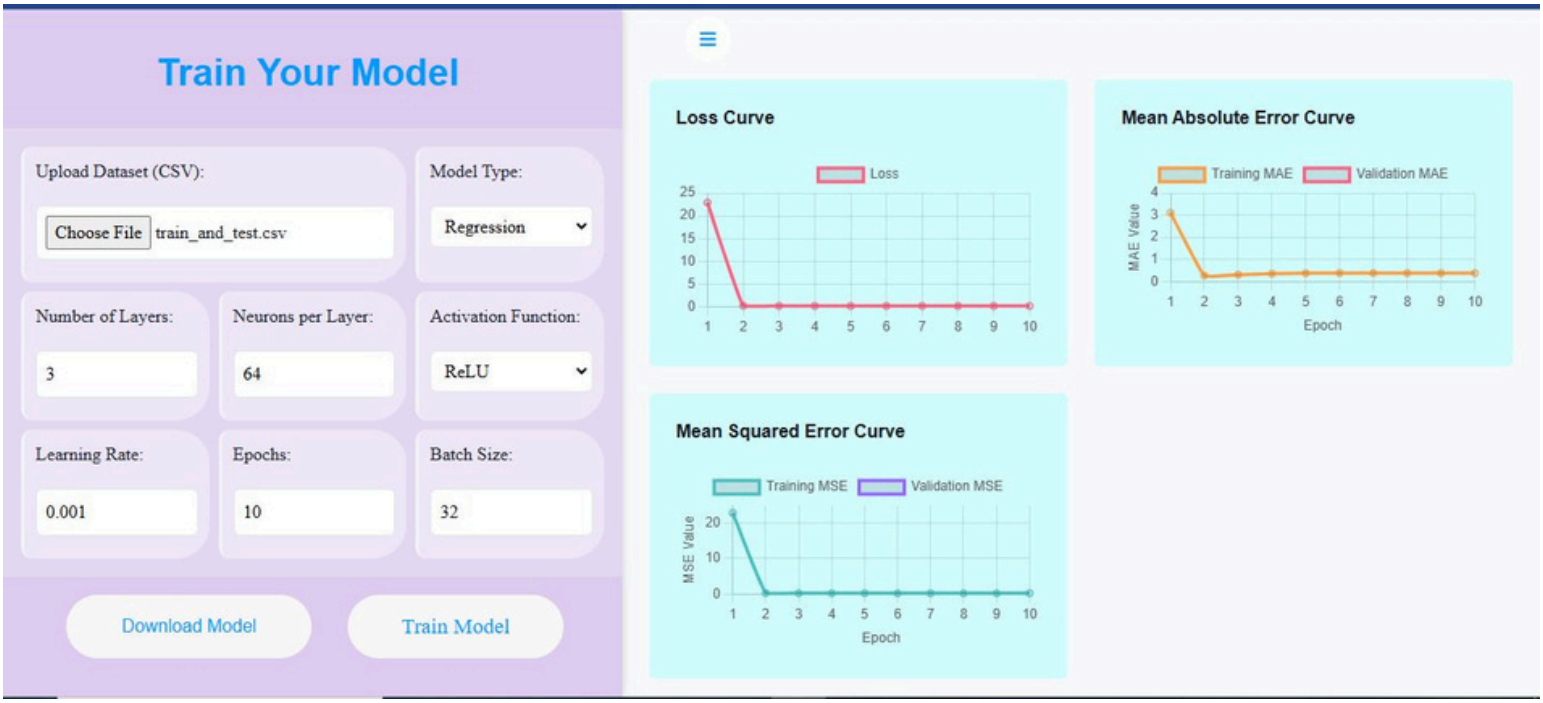
- **Précision** : Indique le taux de précision du modèle sur les données d'entraînement et de validation.

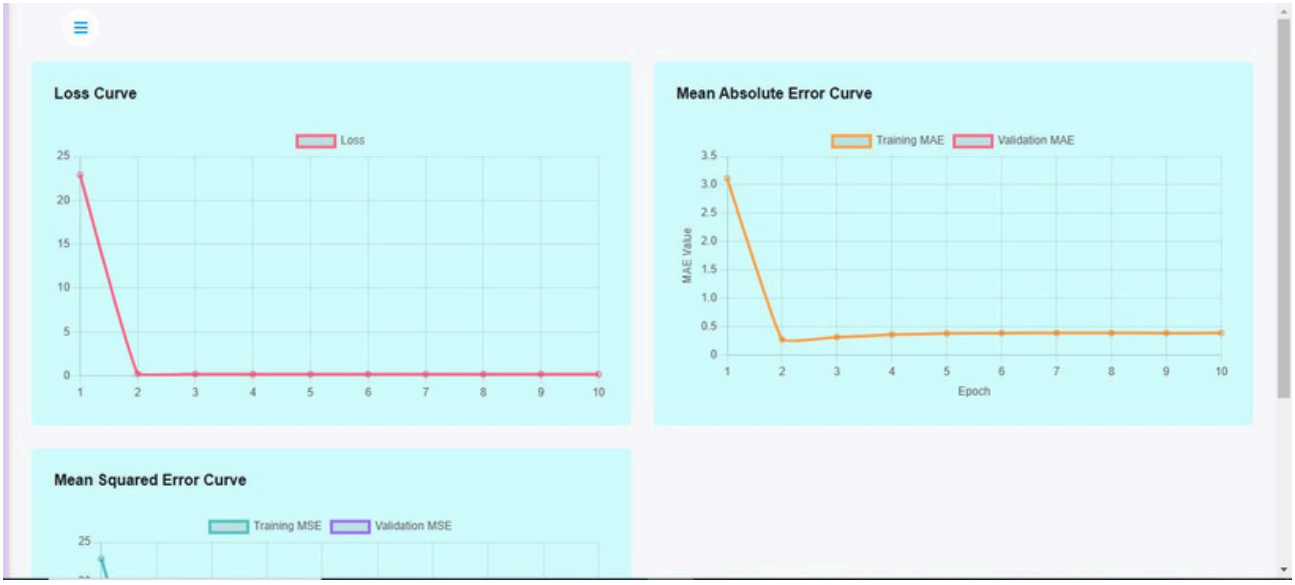
- **La matrice de confusion:** est un tableau qui évalue la performance d'un modèle de classification en montrant les prédictions correctes et incorrectes par classe. Elle est particulièrement utile pour visualiser la capacité d'un modèle à distinguer différentes classes, en fournissant des informations détaillées sur les types d'erreurs commises.

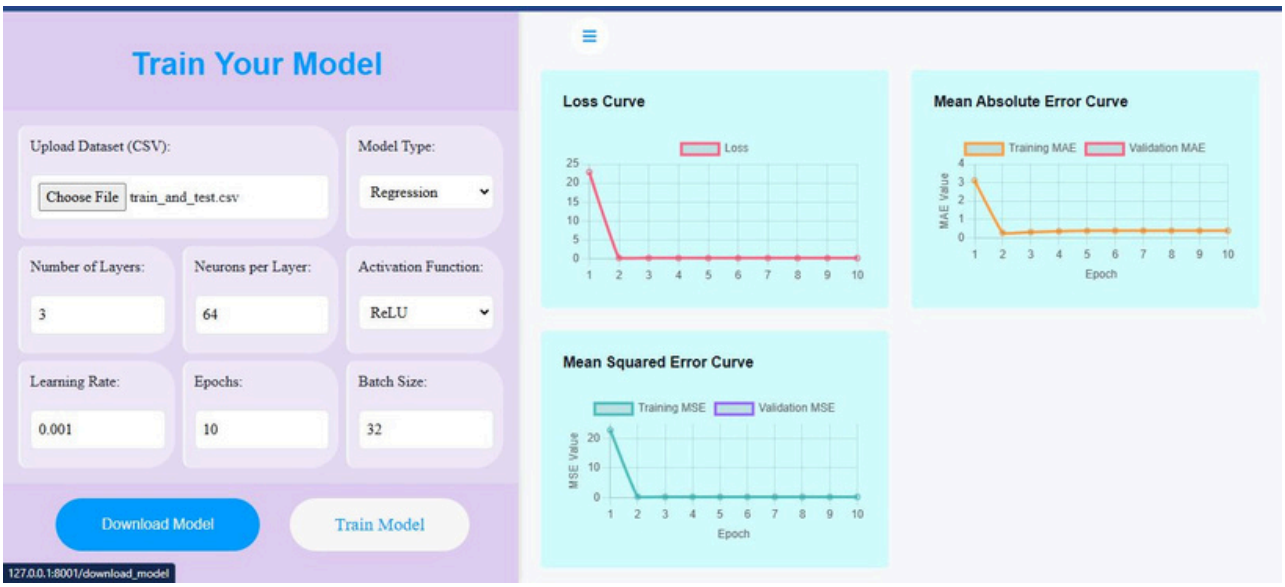
- **La courbe ROC (Receiver Operating Characteristic)** :est un graphique utilisé pour évaluer la performance d'un modèle de classification binaire. Elle permet de visualiser le compromis entre le taux de vrais positifs (rappel) et le taux de faux positifs pour différents seuils de décision. La courbe ROC est particulièrement utile lorsque les classes sont déséquilibrées, car elle montre comment le modèle gère les erreurs de classification.



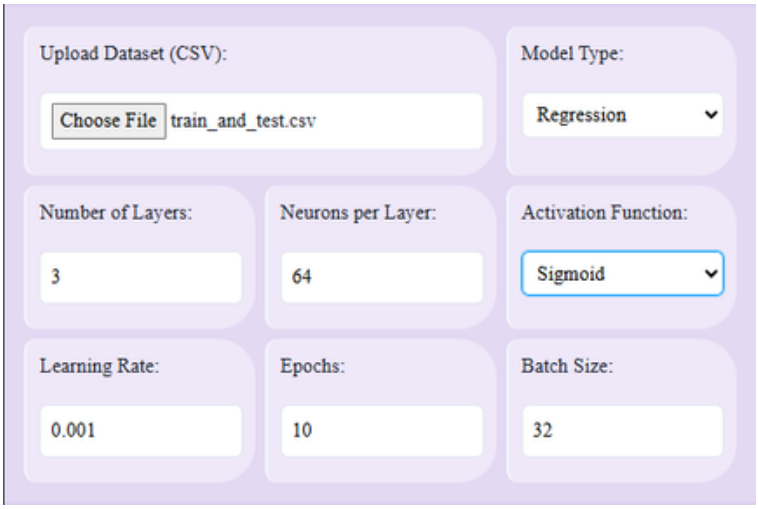
simulation

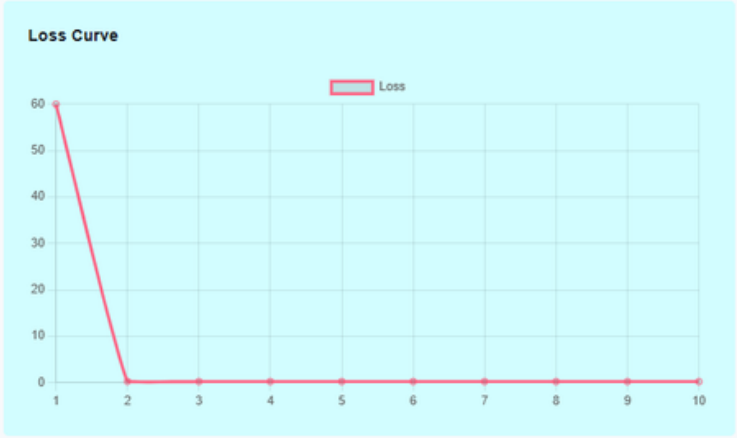

Train Your Model
Upload Dataset (CSV):
Choose File
train_and_test.csv
Model Type:
Regression
Number of Layers:
3
Neurons per Layer:
64
Activation Function:
ReLU
Learning Rate:
0.001
Epochs:
10
Batch Size:
32
Download Model
Train Model
Loss Curve
Loss
Mean Absolute Error Curve
Training MAE
Validation MAE
MAE Value
Epoch
Mean Squared Error Curve
Training MSE
Validation MSE
MSE Value
Epoch

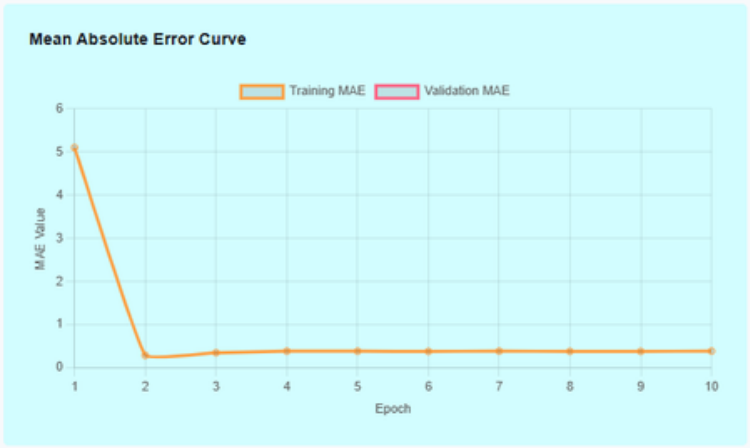

Loss Curve
Loss
Mean Absolute Error Curve
Training MAE
Validation MAE
MAE Value
Epoch
Mean Squared Error Curve
Training MSE
Validation MSE

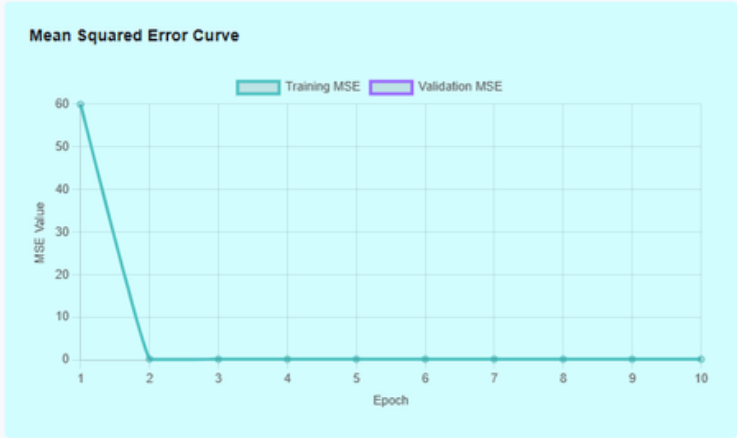

Train Your Model
Upload Dataset (CSV):
Choose File
train_and_test.csv
Model Type:
Regression
Number of Layers:
3
Neurons per Layer:
64
Activation Function:
ReLU
Learning Rate:
0.001
Epochs:
10
Batch Size:
32
Download Model
Train Model
127.0.0.1:8001/download_model
Loss Curve
Loss
Mean Absolute Error Curve
Training MAE
Validation MAE
MAE Value
Epoch
Mean Squared Error Curve
Training MSE
Validation MSE
MSE Value
Epoch

les résultats qu'on a obtenus lors on utilisons la regression:



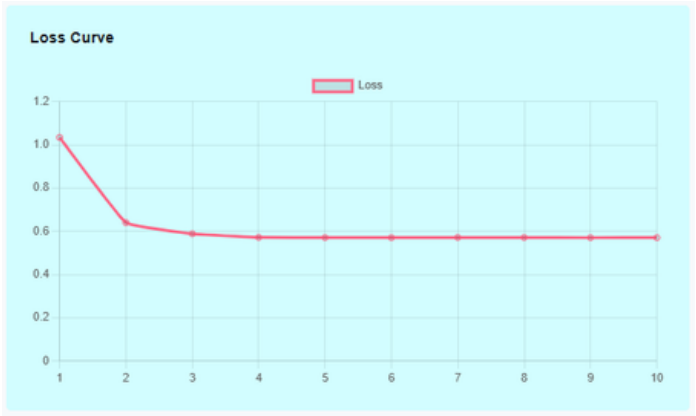


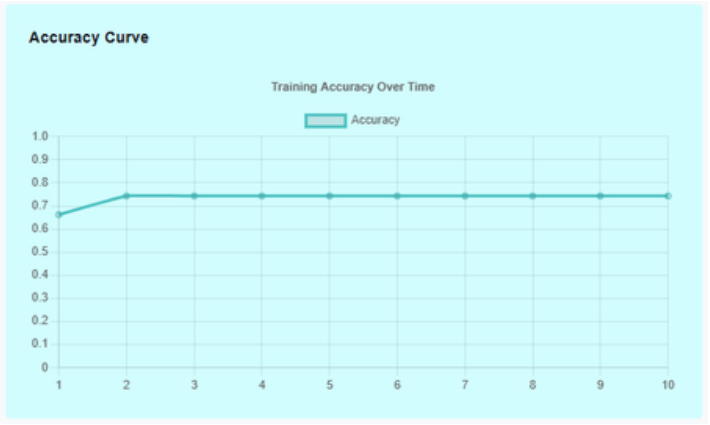


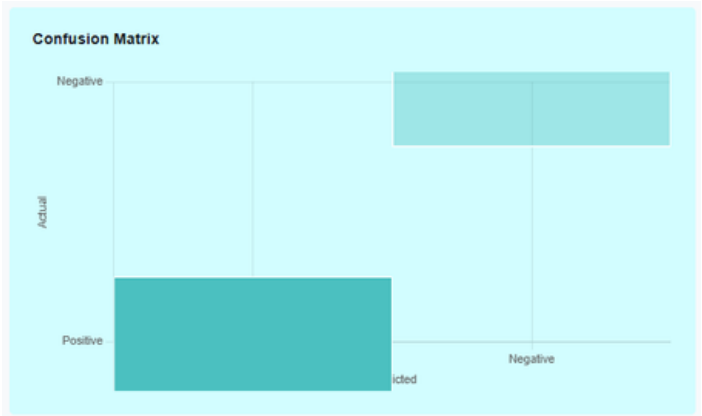


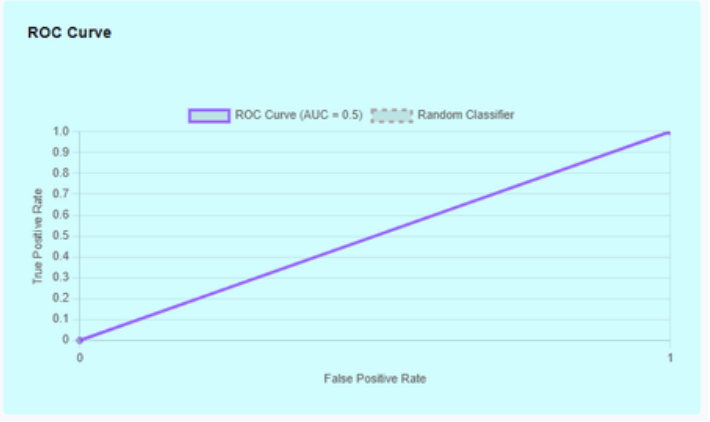
Apres upload la DataSet.

Voici les résultats qu'on a obtenus pour la classification :









Conclusion

Ce projet a permis de concevoir et de développer une application web d'entraînement de réseaux de neurones en utilisant Flask et TensorFlow. L'objectif principal était de fournir une interface conviviale permettant aux utilisateurs de charger leurs propres données, de configurer l'architecture de leur modèle et de visualiser les résultats de l'entraînement en temps réel.

Nous avons réussi à créer une interface intuitive grâce à l'utilisation de HTML, CSS, et JavaScript, intégrant des éléments de Bootstrap pour assurer une expérience utilisateur fluide sur divers appareils. Les fonctionnalités clés, telles que le formulaire de chargement de données et la configuration des hyperparamètres, ont été mises en œuvre avec succès, offrant ainsi une grande flexibilité aux utilisateurs.

L'entraînement du modèle a été géré efficacement par TensorFlow, permettant une personnalisation des paramètres du réseau. Les résultats, visualisés via des graphiques interactifs, ont permis aux utilisateurs de suivre les performances de leur modèle en temps réel, renforçant ainsi l'interaction et l'engagement.

Cependant, plusieurs défis ont été rencontrés tout au long du projet, notamment dans la gestion des différents formats de données et l'optimisation des performances du modèle. Ces obstacles ont été surmontés grâce à des recherches approfondies et des ajustements itératifs dans le code.

Pour l'avenir, plusieurs améliorations sont envisageables. Nous pourrions intégrer de nouveaux types de modèles d'apprentissage automatique, élargir le support pour différents formats de données, et améliorer l'interface utilisateur pour la rendre encore plus intuitive.

En somme, ce projet a non seulement renforcé nos compétences techniques en développement web et en apprentissage automatique, mais a également ouvert des perspectives intéressantes pour des développements futurs dans le domaine de l'intelligence artificielle. Nous sommes convaincus que cette application pourra avoir un impact positif sur ceux qui souhaitent explorer et utiliser les réseaux de neurones pour divers projets.