

PROJET DE FIN De Module

Deep Learning Master MBD

Sujet :

Titanic - Machine Learning from Disasterc

Réalisé PAR :

- AFILAL TRIBAK Firdaous
- MECHHEDAN INSAFE

Sous l'encadrement de :

- Pr. Lotfi EL AACHAK

Année Universitaire : 2022/2023

Introduction générale

Dans le cadre de notre formation en 2^{ème} année Master Big Data et Mobilité au sein de la Faculté des Sciences et Techniques de Tanger, l'établissement cherche à faire évoluer les compétences de ses étudiants par divers moyens tels que les travaux pratiques , et particulièrement les projets encadrés, pour cela, nous avons été amené à réaliser un projet de fin de module Deep Learning qui a pour thème " la réalisation d'une plateforme qui vise à organiser l'entrée et la sortie d'un modèle machine Learning bien traité et choisi par l'une des compétitions kaggle " afin de concrétiser nos acquis durant notre formation.

Le but de la construction de cette application est de prédire les passagers survivants du Titanic à travers l'Atlantique.

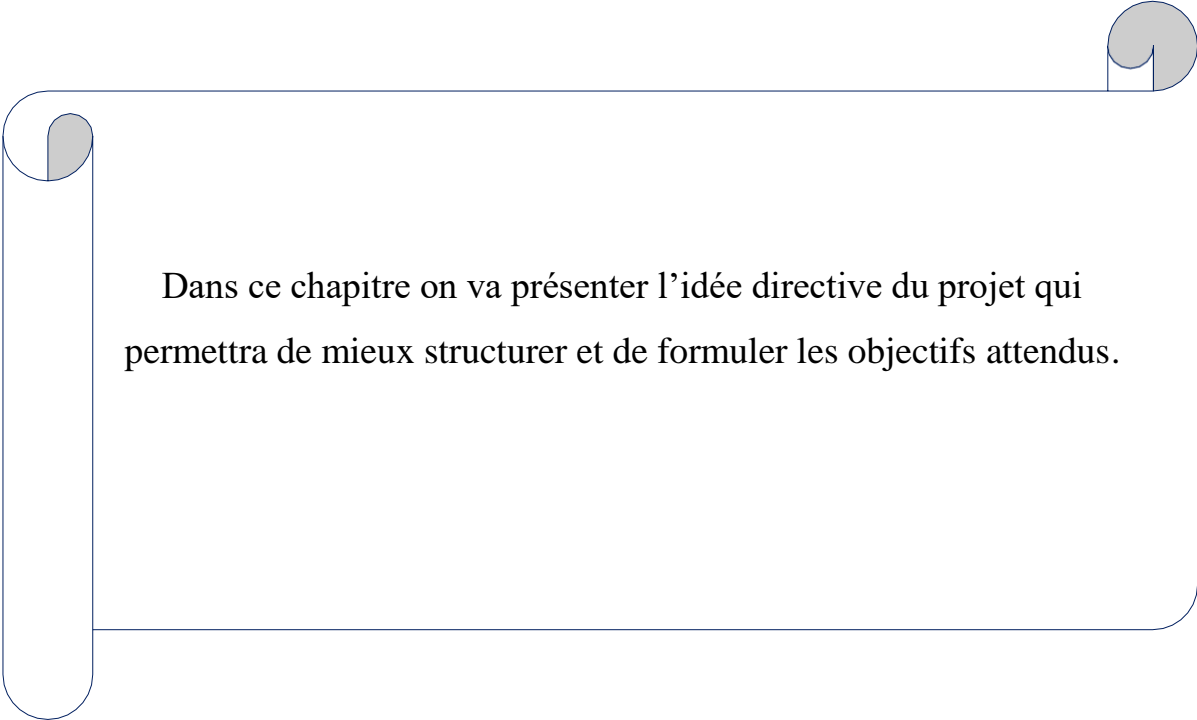
Dans **le premier chapitre**, on va décrire le contexte général de notre projet, par une présentation du projet, ses objectifs et son cahier de charges.

Le deuxième chapitre du rapport va être axé sur les langages de programmation utilisés et les Framework choisis.

Dans **le troisième chapitre**, nous aborderons les différentes étapes de l'intégration de la compétition.

Et finalement, nous avons mis en point notre projet dans **le quatrième chapitre**, en présentant les différentes interfaces de notre site web.

Chapitre 1 : Contexte général du projet



Dans ce chapitre on va présenter l'idée directive du projet qui permettra de mieux structurer et de formuler les objectifs attendus.

1.Introduction

Kaggle est le premier endroit où aller pour quiconque étudie l'apprentissage automatique. Cette plate-forme interactive en ligne fournit des centaines de bases de données et de didacticiels que vous pouvez utiliser pour lancer votre carrière en ML.

Mais ce pour quoi le site Web est le plus célèbre, ce sont ses compétitions. Il peut être difficile pour un nouveau venu de s'orienter dans l'interface et de comprendre par où commencer. Dans cet article, nous allons donc vous aider à démarrer votre premier concours Kaggle !

Les compétitions Kaggle sont des tâches d'apprentissage automatique réalisées par Kaggle ou d'autres sociétés comme Google ou l'OMS. Si vous concourez avec succès, vous pouvez gagner des prix en argent réel.


Les compétitions varient selon les types de problèmes et la complexité. Vous pouvez y participer même si vous êtes débutant. Cependant, les compétitions avancées sont beaucoup plus intéressantes, et une place de leader dans une compétition est un excellent ajout à votre CV d'ingénieur en apprentissage automatique.

Dans la compétition Titanic, vous devez utiliser l'apprentissage automatique pour créer un modèle qui prédit quels passagers ont survécu au naufrage du Titanic. On vous demande de construire un modèle prédictif qui répond à la question : "quels passagers étaient les plus susceptibles de survivre ?" en utilisant des données comme le nom, l'âge, le sexe, la classe socio-économique.

2.Objectifs

- L'intégration a la compétition Titanic
- Prédit quels passagers ont survécu au naufrage du Titanic
- La mise on forme d'une plateforme permet de visualiser le model de la compétition

Chapitre 2 : Mise en œuvre informatique



Après avoir donné une idée générale sur le projet, on abordera dans ce chapitre les outils utilisés pour la réalisation du projet.

Afin de mettre en place et de concevoir notre application, on a fait usage de différentes techniques et d'outils. De ce fait, cette partie aura pour mission de définir ces différentes méthodes

1. Langage de programmation et technologies :



Scheme, Smalltalk et Tcl.

Python (prononcé /pi.tɔ̃/) est un langage de programmation interprété, multiparadigme et multiplateformes. Il favorise la programmation impérative structurée, fonctionnelle et orientée objet. Il est doté d'un typage dynamique fort, d'une gestion automatique de la mémoire par ramasse-miettes et d'un système de gestion d'exceptions ; il est ainsi similaire à Perl, Ruby,

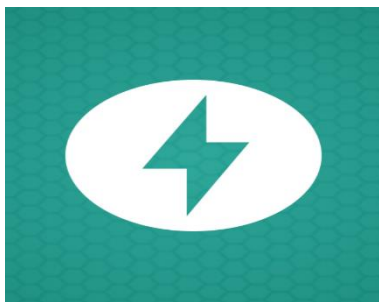


MLOps ou ML Ops est un ensemble de pratiques qui vise à déployer et maintenir des modèles de machine learning en production de manière fiable et efficace¹. Ce terme est composé de "machine learning" et de la pratique de développement continu de devops dans le domaine des logiciels. Les modèles d'apprentissage automatique sont testés et développés dans des systèmes expérimentaux isolés. Lorsqu'un algorithme est prêt à être lancé, le MLOps est pratiqué entre les scientifiques des données, les devops et les ingénieurs en apprentissage automatique pour faire passer l'algorithme aux systèmes de production



Docker est une plateforme permettant de lancer certaines applications dans des conteneurs logiciels.

Selon la firme de recherche sur l'industrie 451 Research, « Docker est un outil qui peut emballer une application et ses dépendances dans un conteneur isolé, qui pourra être exécuté sur n'importe quel serveur ». Il ne s'agit pas de virtualisation, mais de conteneurisation, une forme plus légère qui s'appuie sur certaines parties de la machine hôte pour son fonctionnement. Cette approche permet d'accroître la flexibilité et la portabilité d'exécution d'une application, laquelle va pouvoir tourner de façon fiable et prévisible sur une grande variété de machines hôtes, que ce soit sur la machine locale, un cloud privé ou public, une machine nue, etc.



FastAPI est un framework Web pour développer des API RESTful en Python. FastAPI est basé sur Pydantic et des indications de type pour valider, sérialiser et désérialiser les données, et générer automatiquement des documents OpenAPI.

Il prend entièrement en charge la programmation asynchrone et peut fonctionner avec les serveurs Gunicorn et ASGI pour la production tels que Uvicorn et Hypercorn. Pour améliorer la convivialité pour les développeurs, le support de l'éditeur a été envisagé dès les premiers jours du projet.

2. Les bibliothèques et les standards :

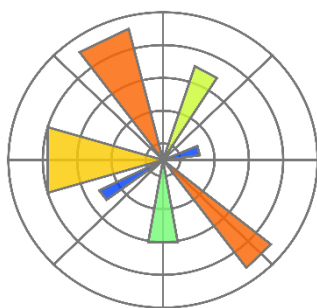


NumPy est une bibliothèque pour langage de programmation Python, destinée à manipuler des matrices ou tableaux multidimensionnels ainsi que des fonctions mathématiques opérant sur ces tableaux.

Plus précisément, cette bibliothèque logicielle libre et open source fournit de multiples fonctions permettant notamment de créer directement un tableau depuis un fichier ou au contraire de sauvegarder un tableau dans un fichier, et manipuler des vecteurs, matrices et polynômes.

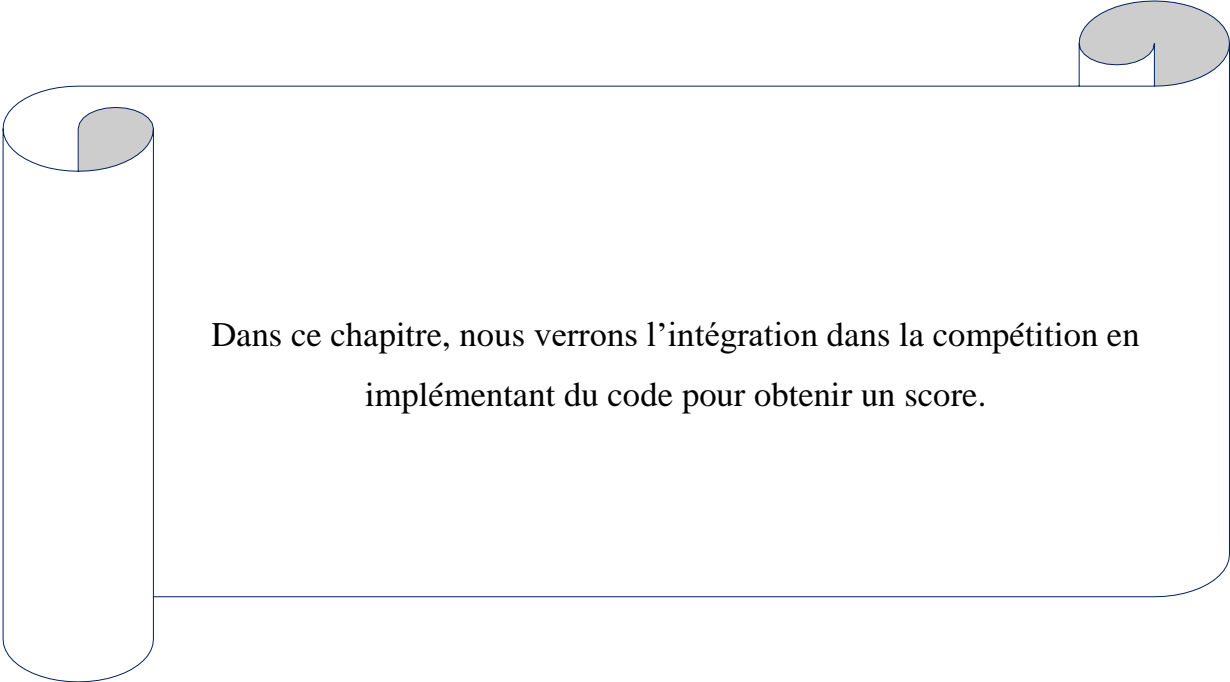


Pandas est une bibliothèque écrite pour le langage de programmation Python permettant la manipulation et l'analyse des données. Elle propose en particulier des structures de données et des opérations de manipulation de tableaux numériques et de séries temporelles.



Matplotlib est une bibliothèque du langage de programmation Python destinée à tracer et visualiser des données sous forme de graphiques. Elle peut être combinée avec les bibliothèques python de calcul scientifique NumPy et SciPy⁶. Elle fournit également une API orientée objet, permettant d'intégrer des graphiques dans des applications, utilisant des outils d'interface graphique polyvalents tels que Tkinter, wxPython, Qt ou GTK.

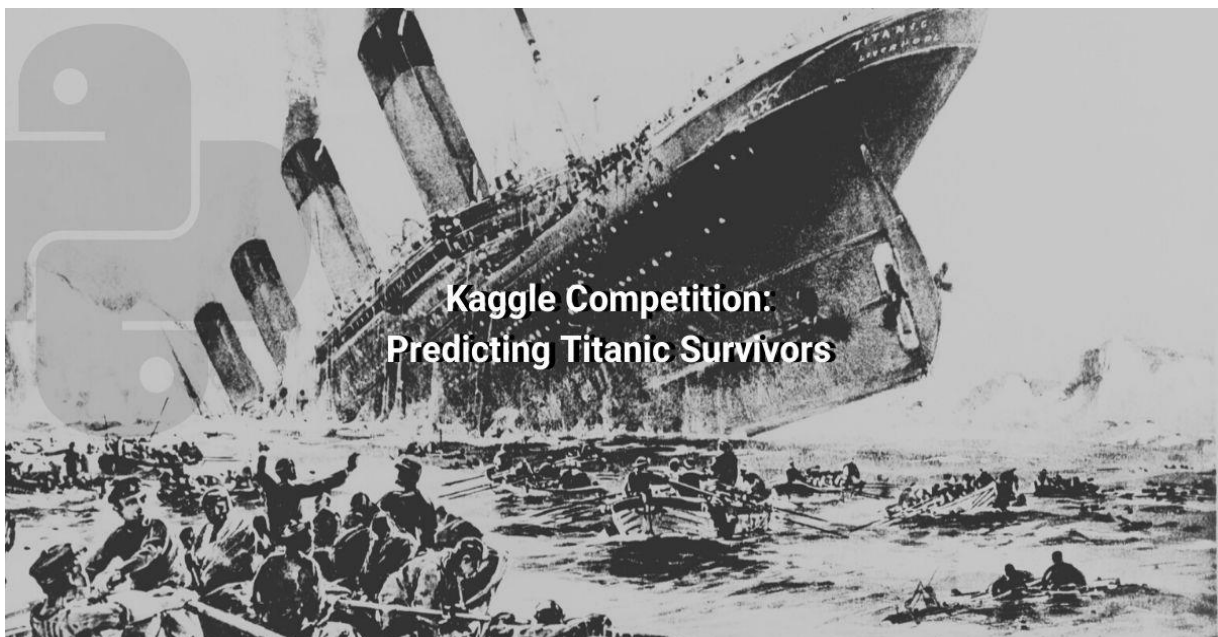
Chapitre 3 : L'intégration de compétition



Dans ce chapitre, nous verrons l'intégration dans la compétition en implémentant du code pour obtenir un score.

La Compétition :

Une tragédie comme le naufrage du RMS Titanic en 1912, quatre jours après le début du voyage inaugural du plus grand navire du monde, peut être analysée sous plusieurs angles : la signification historique, les conséquences géopolitiques ou, dans le cadre du concours Kaggle, peut être utilisé comme un scénario qui peut aider à expliquer la puissance de Machine Learning (ML).



La longue compétition Titanic sur Kaggle vous demande de prédire quels passagers ont survécu au naufrage du Titanic. Parce qu'il n'y avait pas assez de canots de sauvetage pour tout le monde, 1502 des 2224 personnes à bord sont mortes, mais que vous ayez survécu ou que vous mouriez n'était pas aléatoire. Le défi consiste donc à construire un modèle prédictif basé sur les données fournies par Kaggle pour répondre à la question : "quels types de traits caractérisent les survivants ?"

.

Pour Prédire la survie sur le Titanic, nous suivrons les étapes suivantes :

1. Collecte des données

Charger le train et le test l'ensemble de données à l'aide de Pandas

Firdaous&Insafe

NotebookDataLogsComments (0)Settings

In [3]:

```
import pandas as pd

train = pd.read_csv('/kaggle/input/titanic/train.csv')
test = pd.read_csv('/kaggle/input/titanic/test.csv')
```

In [4]:

```
train.head(100)
```

Out[4]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
...
95	96	0	3	Shorney, Mr. Charles Joseph	male	NaN	0	0	374910	8.0500	NaN	S
96	97	0	1	Goldschmidt, Mr. George B	male	71.0	0	0	PC 17754	34.6542	A5	C
97	98	1	1	Greenfield, Mr. William Bertram	male	23.0	0	1	PC 17759	63.3583	D10 D12	C
98	99	1	2	Doling, Mrs. John T (Ada Julia Bone)	female	34.0	0	1	231919	23.0000	NaN	S
99	100	0	2	Kantor, Mr. Sinai	male	34.0	1	0	244367	26.0000	NaN	S

2. L'analyse exploratoire des données

- Dictionnaire de données
- Survived: 0 = No, 1 = Yes
- pclass: Ticket class 1 = 1st, 2 = 2nd, 3 = 3rd
- sibsp: # of siblings / spouses aboard the Titanic
- parch: # of parents / children aboard the Titanic
- ticket: Ticket number
- cabin: Cabin number
- embarked: Port of Embarkation C = Cherbourg, Q = Queenstown, S = Southampton

➤ Total des lignes et des colonnes

Nous pouvons voir qu'il y a 891 lignes et 12 colonnes dans notre jeu de données d'entraînement.

Firdaous&Insafe

Notebook

Data

Logs

Comments (0)

Settings

1

Edit

Out[5]:

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q
1	893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S
2	894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q
3	895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S
4	896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	S
...
95	987	3	Tenglin, Mr. Gunnar Isidor	male	25.0	0	0	350033	7.7958	NaN	S
96	988	1	Cavendish, Mrs. Tyrell William (Julia Florence...	female	76.0	1	0	19877	78.8500	C46	S
97	989	3	Makinen, Mr. Kalle Edvard	male	29.0	0	0	STON/O 2. 3101268	7.9250	NaN	S
98	990	3	Braf, Miss. Elin Ester Maria	female	20.0	0	0	347471	7.8542	NaN	S
99	991	3	Nancarrow, Mr. William Henry	male	33.0	0	0	A/5. 3338	8.0500	NaN	S

100 rows × 11 columns

In [6]:

train.shape

Out[6]:

(891, 12)

- Import python lib Pour la visualisation
- Bar Chart pour Categorical Features

Firdaous&Insafe

Notebook

Data

Logs

Comments (0)

Settings

Embarked

0

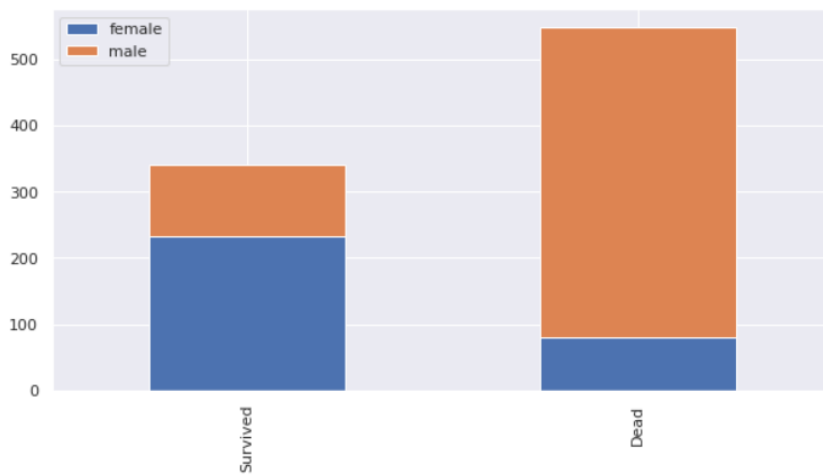
dtype: int64

In [12]:

```
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
sns.set() # setting seaborn default for plots
```

In [14]:

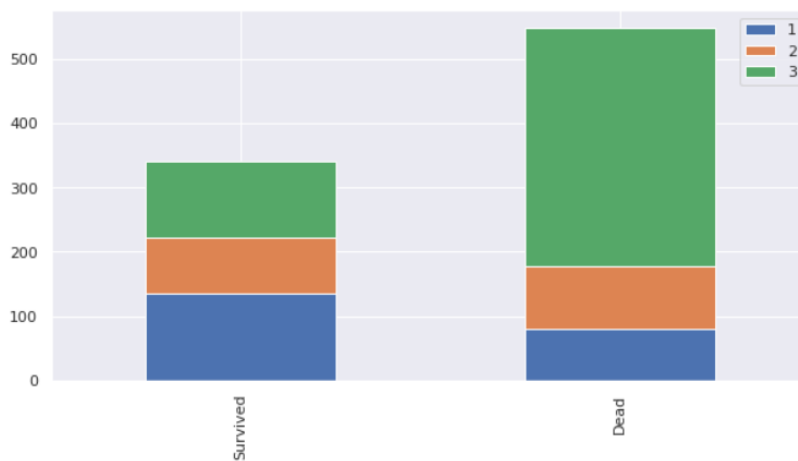
```
bar_chart('Sex')
```



Le graphique confirme que les femmes ont plus de chances de survivre que les hommes

In [15]:

```
bar_chart('Pclass')
```



Le graphique confirme que la 1ère classe a plus probablement survécu que les autres classes

Le graphique confirme que la 3e classe est plus probablement morte que les autres classes

3. Ingénierie des fonctionnalités

L'ingénierie des fonctionnalités est le processus d'utilisation de la connaissance du domaine des données pour créer des fonctionnalités (vecteurs de fonctionnalités) qui font fonctionner les algorithmes d'apprentissage automatique. Vecteur de caractéristiques est un vecteur à n dimensions de caractéristiques numériques qui représentent un objet.

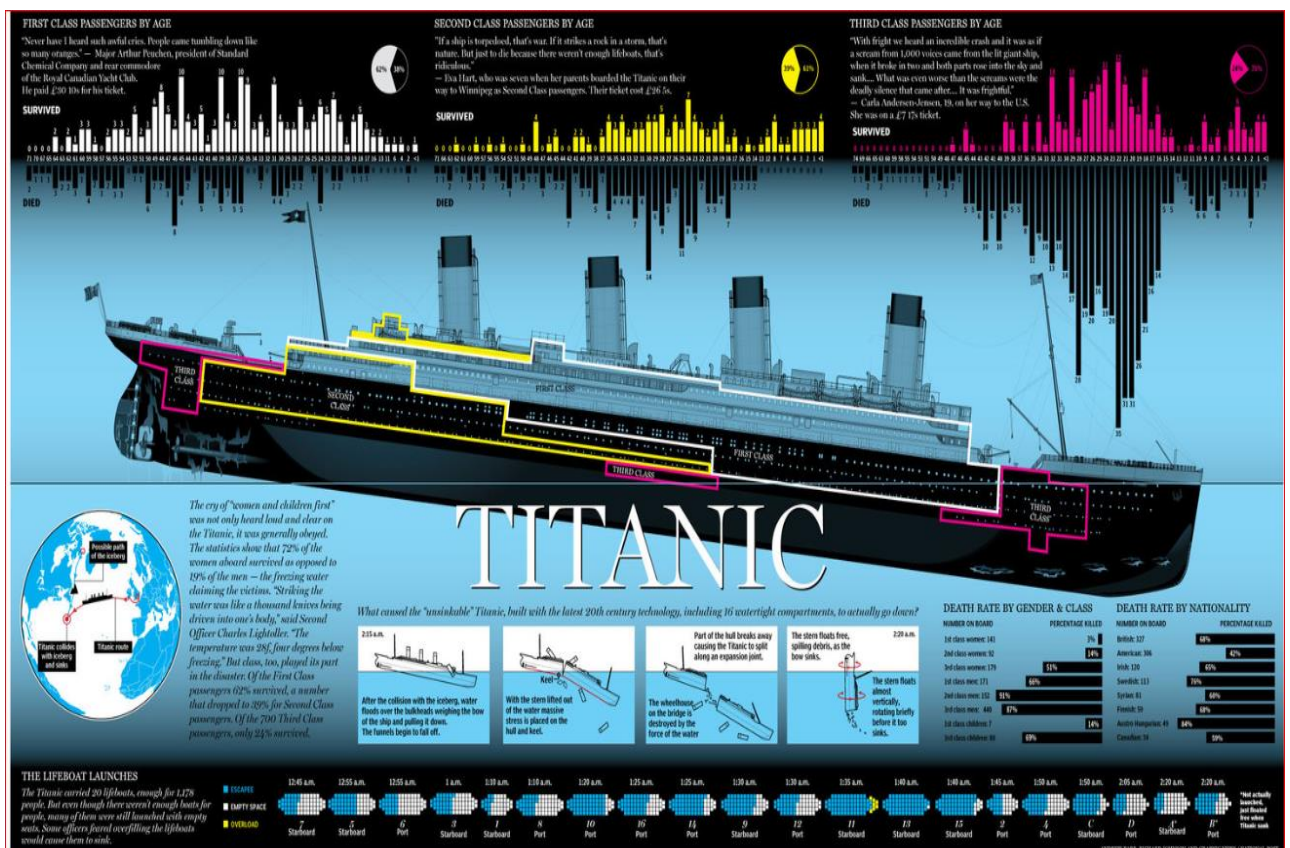
De nombreux algorithmes en Machine Learning nécessitent une représentation numérique des objets,

Puisque de telles représentations facilitent le traitement et l'analyse statistique

a) *Comment Titanic a coulé ?*

Il a coulé de la proue du navire où se trouvaient les chambres de troisième classe

On conclure que, Pclass est une caractéristique clé pour le classifieur



b) *name*

➤ Title map

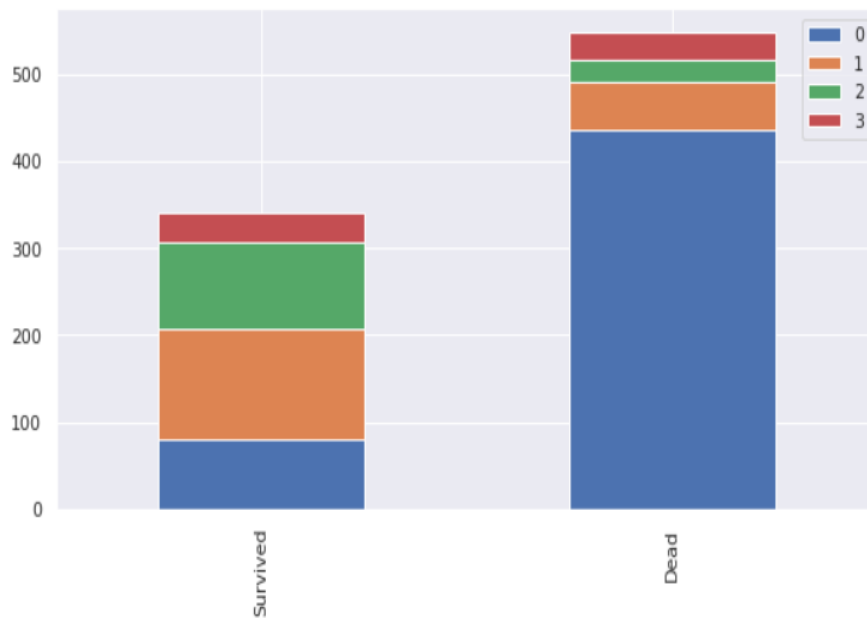
Mr	0
Miss	1
Mrs	2
Others	3

Firdaous&Insafe

Notebook Data Logs Comments (0) Settings

In [28]:

```
bar_chart('Title')
```



c) *Sexe*

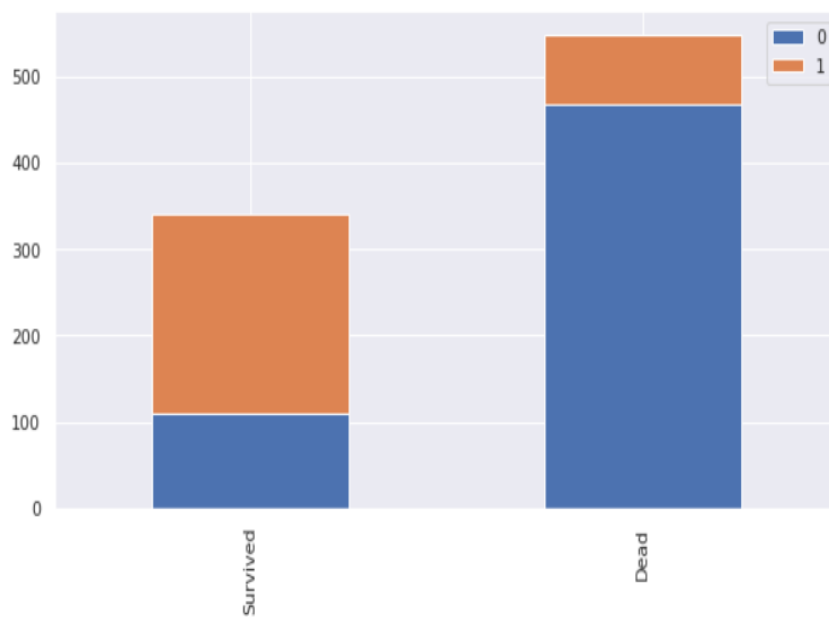
Male	0
Female	1

Firdaous&Insafe

Notebook Data Logs Comments (0) Settings

In [33]:

```
bar_chart('Sex')
```



d) Age

Utilisons l'âge médian du titre pour l'âge manquant

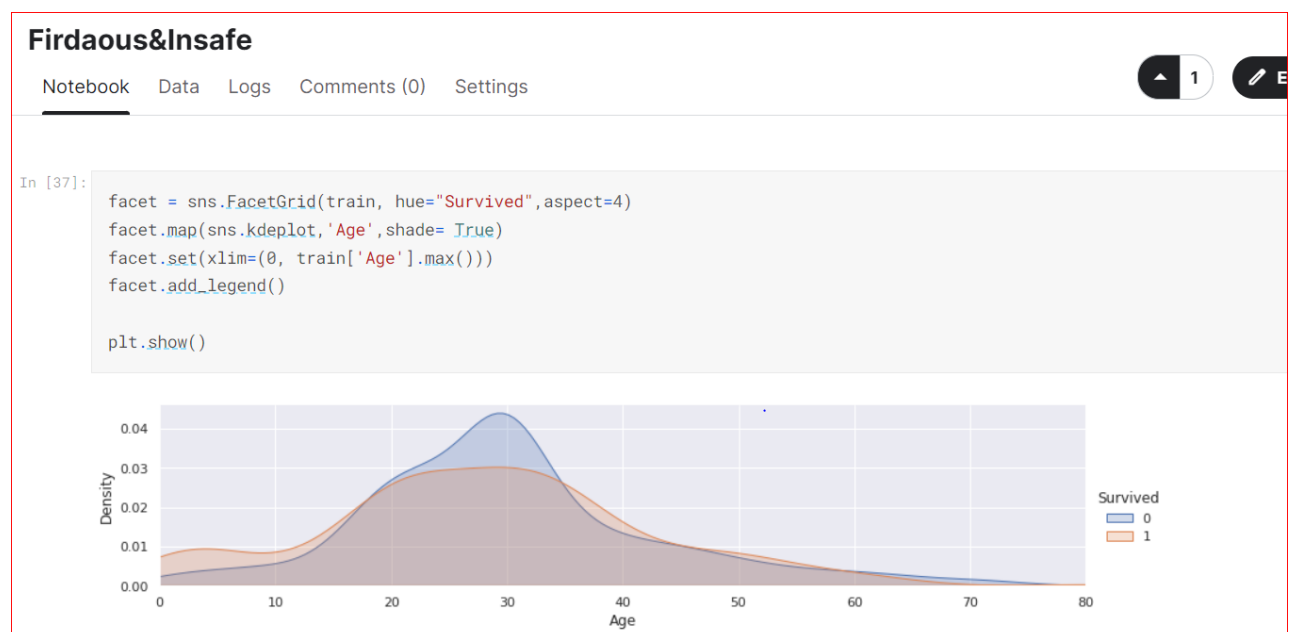
Firdaous&Insafe

Notebook Data Logs Comments (0) Settings

```
In [35]: # fill missing age with median age for each title (Mr, Mrs, Miss, Others)
train["Age"].fillna(train.groupby("Title")["Age"].transform("median"), inplace=True)
test["Age"].fillna(test.groupby("Title")["Age"].transform("median"), inplace=True)
```

```
In [36]: train.head(30)
train.groupby("Title")["Age"].transform("median")
```

```
Out[36]:
0      30.0
1      35.0
2      21.0
3      35.0
4      30.0
...
886     9.0
887    21.0
888    21.0
889    30.0
890    30.0
Name: Age, Length: 891, dtype: float64
```



4. La modélisation

Pour le choix de modèle nous avons tester plusieurs modèles comme Cross Validation, KNN, Arbre de décision, Radom Forest, SVM et nous avons prendre le score le plus grand de ces modèles pour l'utiliser dans la partie test

6.2.3 Ramdom Forest

```
In [81]: clf = RandomForestClassifier(n_estimators=13)
scoring = 'accuracy'
score = cross_val_score(clf, train_data, target, cv=k_fold, n_jobs=1, scoring=scoring)
print(score)
```

```
[ 0.77777778  0.80898876  0.82022472  0.76404494  0.86516854  0.82022472
 0.79775281  0.80898876  0.76404494  0.83146067]
```

```
In [82]: # Random Forest Score
round(np.mean(score)*100, 2)
```

```
Out[82]: 80.59
```

6.2.4 Naive Bayes

```
In [83]: clf = GaussianNB()
scoring = 'accuracy'
score = cross_val_score(clf, train_data, target, cv=k_fold, n_jobs=1, scoring=scoring)
print(score)
```

```
[ 0.85555556  0.73033708  0.75280899  0.75280899  0.70786517  0.80898876
 0.76404494  0.80898876  0.86516854  0.83146067]
```

```
In [84]: # Naive Bayes Score
round(np.mean(score)*100, 2)
```

```
Out[84]: 78.78
```

6.2.5 SVM

```
In [85]: clf = SVC()
scoring = 'accuracy'
score = cross_val_score(clf, train_data, target, cv=k_fold, n_jobs=1, scoring=scoring)
print(score)
```

```
[ 0.83333333  0.80898876  0.83146067  0.82022472  0.84269663  0.82022472
 0.84269663  0.85393258  0.83146067  0.86516854]
```

```
In [86]: round(np.mean(score)*100,2)
```

```
Out[86]: 83.5
```

Alors on peut voir que le score le plus grand c'est de modèle SVM

5. Testing

Nous avons utilisé le modèle SVM

```
In [79]:
clf = SVC()
clf.fit(train_data, target)

test_data = test.drop("PassengerId", axis=1).copy()
prediction = clf.predict(test_data)
```

```
In [80]:
submission = pd.DataFrame({
    "PassengerId": test["PassengerId"],
    "Survived": prediction
})

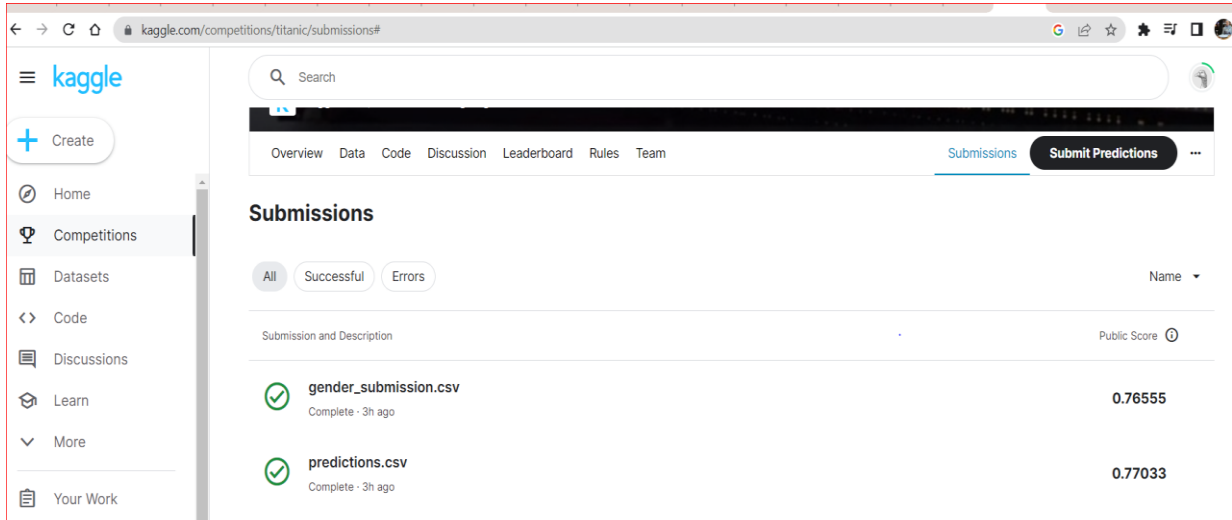
submission.to_csv('submission.csv', index=False)
```

```
In [81]:
submission = pd.read_csv('submission.csv')
submission.head(20)
```

Out[81]:

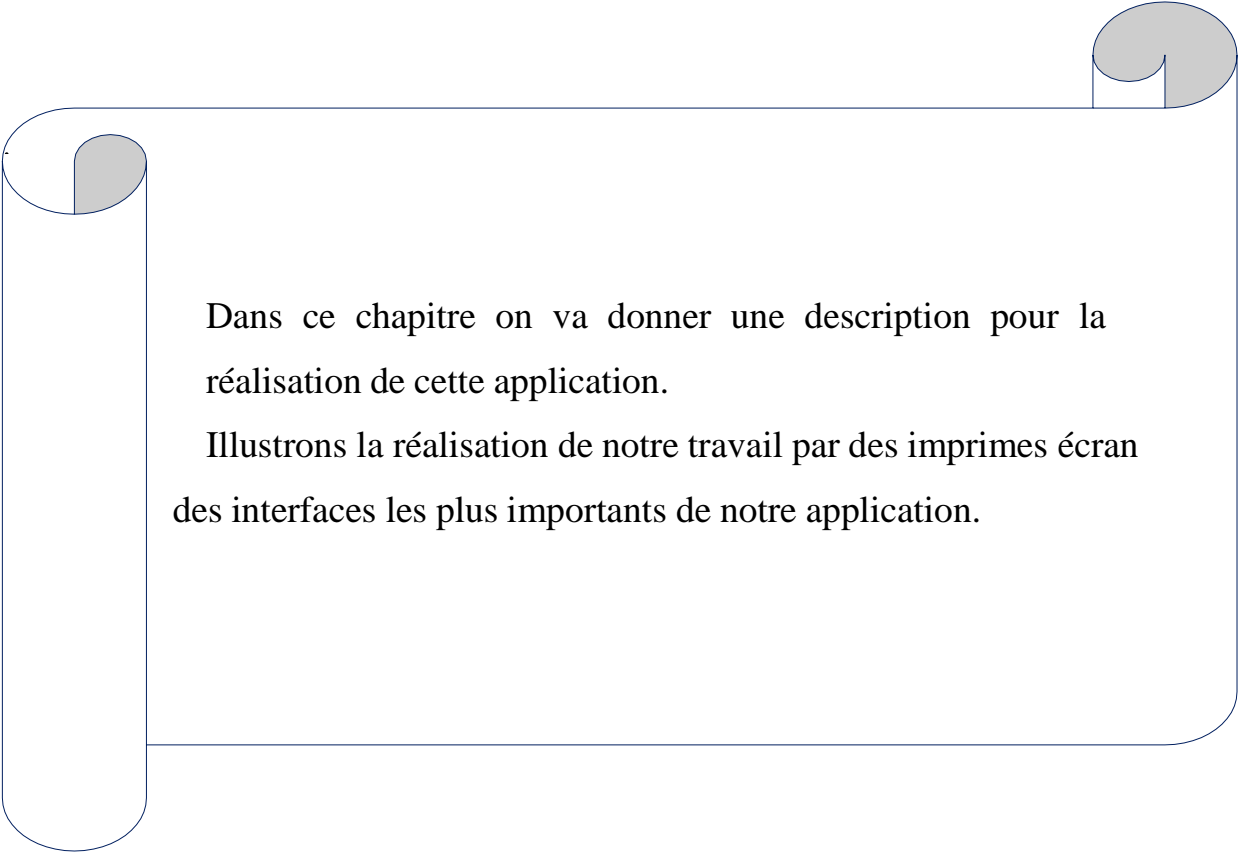
	PassengerId	Survived
0	892	0
1	893	0
2	894	0
3	895	0
4	896	0
5	897	0
6	898	0
7	899	0
8	900	0
9	901	0
10	902	0
11	903	0
12	904	1
13	905	0
14	906	1
15	907	0
16	908	0
17	909	0
18	910	0
19	911	0

Après la réalisation du code et avoir un fichier CSV de prédiction nous pouvons avoir notre score



Submissions		Name ▾
Submission and Description		Public Score ⓘ
✓	gender_submission.csv Complete · 3h ago	0.76555
✓	predictions.csv Complete · 3h ago	0.77033

Chapitre 4 : Réalisation du Projet

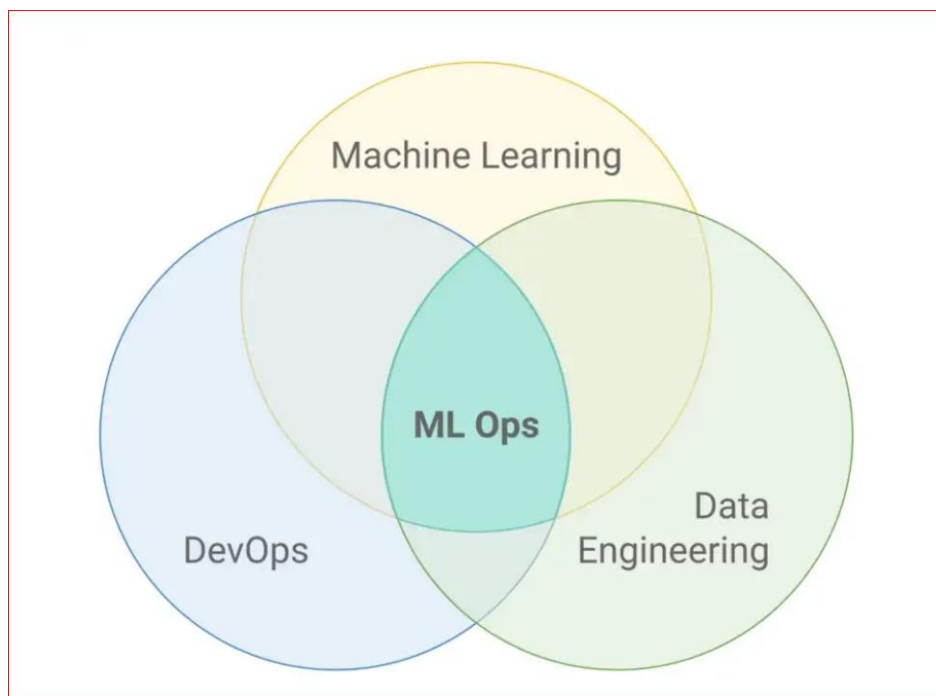
A decorative graphic consisting of a large, light blue rectangular frame with rounded corners. The frame has a vertical scroll-like element on the left side and a horizontal scroll-like element at the top right, both with grey shading to give a 3D effect.

Dans ce chapitre on va donner une description pour la réalisation de cette application.

Illustrons la réalisation de notre travail par des imprimes écran des interfaces les plus importants de notre application.

1. Le principe de base

la fiabilité est au cœur des MLOps. Cela signifie que tout le code que vous exécutez localement dans votre environnement Jupyter ou Python doit produire le même résultat dans l'environnement de production. Ce n'est pas difficile à résoudre quand il y a des graines aléatoires. Vous pouvez utiliser la même graine dans tout votre code, et tout ira bien. Mais les choses peuvent devenir un peu plus difficiles en ce qui concerne la configuration de l'environnement et les dépendances.



La reproductibilité est le mot. Comment pouvez-vous vous assurer que votre code peut être reproduit sur une autre machine ?

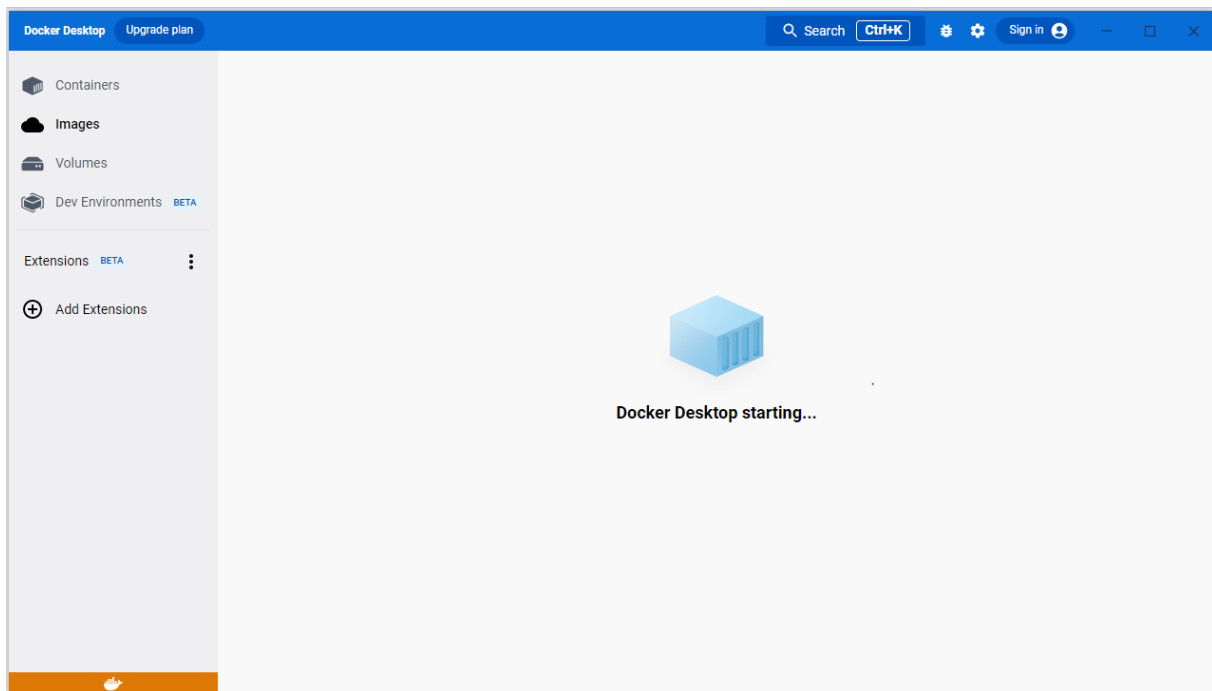
2. Exigences du projet

Pour répliquer ce projet dans votre environnement local, vous devez avoir Python installé (j'utilise python 3.10.1) et le moteur Docker.

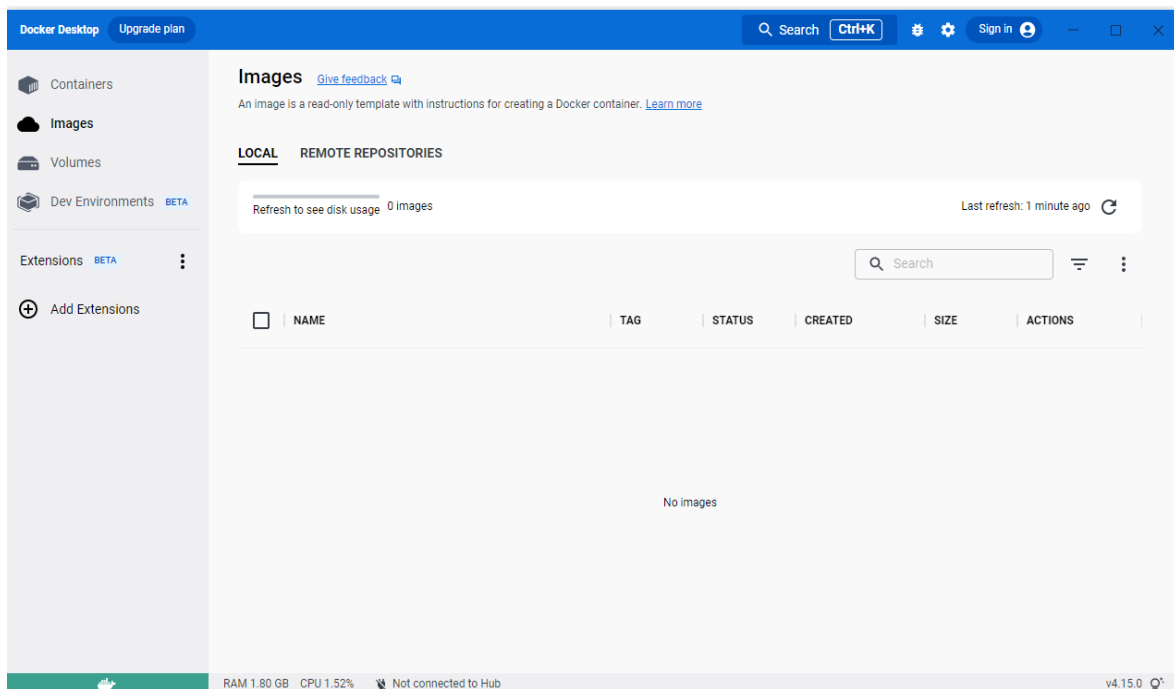
```
C:\Users\Acer>python --version
Python 3.10.1

C:\Users\Acer>
```

a) Docker installation



Avant la creation de l'image :



Après avoir installé le Docker, créez un nouveau dossier pour notre projet et créez un nouvel environnement virtuel avec les éléments suivants :

```
C:\Windows\System32\cmd.exe
Microsoft Windows [version 10.0.19045.2364]
(c) Microsoft Corporation. Tous droits réservés.

D:\DTéléchargement\basic-mlops-main\basic-mlops-main>python3 -m venv .env
D:\DTéléchargement\basic-mlops-main\basic-mlops-main>.env\Scripts\activate.bat
```

3. Développement du modèle

Comme l'accent est mis ici sur l'utilisation de Docker et non sur la modélisation, nous utiliserons un simple classificateur Gradient Boosting avec les fonctionnalités numériques dont nous disposons. Si vous clonez le dépôt, installez les dépendances avec la commande suivante :

```
C:\Windows\System32\cmd.exe
(.env) D:\DTéléchargement\basic-mlops-main\basic-mlops-main>pip install -r requirements.txt
Requirement already satisfied: fastapi==0.88.0 in d:\dtéléchargement\basic-mlops-main\basic-mlops-main\env\lib\site-packages (from -r requirements.txt (line 1)) (0.88.0)
Requirement already satisfied: pandas==1.5.2 in d:\dtéléchargement\basic-mlops-main\basic-mlops-main\env\lib\site-packages (from -r requirements.txt (line 2)) (1.5.2)
Requirement already satisfied: pydantic==1.10.2 in d:\dtéléchargement\basic-mlops-main\basic-mlops-main\env\lib\site-packages (from -r requirements.txt (line 3)) (1.10.2)
Requirement already satisfied: scikit-learn==1.1.3 in d:\dtéléchargement\basic-mlops-main\basic-mlops-main\env\lib\site-packages (from -r requirements.txt (line 4)) (1.1.3)
Requirement already satisfied: uvicorn==0.20.0 in d:\dtéléchargement\basic-mlops-main\basic-mlops-main\env\lib\site-packages (from -r requirements.txt (line 5)) (0.20.0)
Requirement already satisfied: starlette==0.22.0 in d:\dtéléchargement\basic-mlops-main\basic-mlops-main\env\lib\site-packages (from fastapi==0.88.0->-r requirements.txt (line 1)) (0.22.0)
Requirement already satisfied: numpy==1.21.0 in d:\dtéléchargement\basic-mlops-main\basic-mlops-main\env\lib\site-packages (from pandas==1.5.2->-r requirements.txt (line 2)) (1.24.1)
Requirement already satisfied: pytz==2020.1 in d:\dtéléchargement\basic-mlops-main\basic-mlops-main\env\lib\site-packages (from pandas==1.5.2->-r requirements.txt (line 2)) (2022.7)
Requirement already satisfied: python-dateutil==2.8.1 in d:\dtéléchargement\basic-mlops-main\basic-mlops-main\env\lib\site-packages (from pandas==1.5.2->-r requirements.txt (line 2)) (2.8.2)
Requirement already satisfied: typing-extensions==4.1.0 in d:\dtéléchargement\basic-mlops-main\basic-mlops-main\env\lib\site-packages (from pydantic==1.10.2->-r requirements.txt (line 3)) (4.4.0)
Requirement already satisfied: scipy==1.3.2 in d:\dtéléchargement\basic-mlops-main\basic-mlops-main\env\lib\site-packages (from scikit-learn==1.1.3->-r requirements.txt (line 4)) (1.10.0)
Requirement already satisfied: joblib==1.0.0 in d:\dtéléchargement\basic-mlops-main\basic-mlops-main\env\lib\site-packages (from scikit-learn==1.1.3->-r requirements.txt (line 4)) (1.2.0)
Requirement already satisfied: threadpoolctl==2.0.0 in d:\dtéléchargement\basic-mlops-main\basic-mlops-main\env\lib\site-packages (from scikit-learn==1.1.3->-r requirements.txt (line 4)) (3.1.0)
Requirement already satisfied: h11==0.8 in d:\dtéléchargement\basic-mlops-main\basic-mlops-main\env\lib\site-packages (from uvicorn==0.20.0->-r requirements.txt (line 5)) (0.14.0)
```

Et exécutez le script d'entraînement avec cette commande :

```
C:\Windows\System32\cmd.exe
(.env) D:\DTéléchargement\basic-mlops-main\basic-mlops-main>python -m train_titanic
[2023-01-09T12:07:43Maroc (heure d'été)] INFO in __main__: Starting dataset download.
[2023-01-09T12:07:45Maroc (heure d'été)] INFO in __main__: Dataset download finished.
[2023-01-09T12:07:45Maroc (heure d'été)] INFO in __main__: Starting dataset preprocessing.
[2023-01-09T12:07:45Maroc (heure d'été)] INFO in __main__: Dataset preprocessing finished.
[2023-01-09T12:07:45Maroc (heure d'été)] INFO in __main__: Training classifier model.
[2023-01-09T12:07:45Maroc (heure d'été)] INFO in __main__: Finished model training.
[2023-01-09T12:07:45Maroc (heure d'été)] INFO in __main__: Starting model evaluation.
[2023-01-09T12:07:45Maroc (heure d'été)] INFO in __main__: The model f1 score is: 1.0.
[2023-01-09T12:07:45Maroc (heure d'été)] INFO in __main__: Saving artifact train_df.
[2023-01-09T12:07:45Maroc (heure d'été)] ERROR in __main__: Unable to find a usable engine; tried using: 'pyarrow', 'fastparquet'.
A suitable version of pyarrow or fastparquet is required for parquet support.
Trying to import the above resulted in these errors:
- Missing optional dependency 'pyarrow'. pyarrow is required for parquet support. Use pip or conda to install pyarrow.
- Missing optional dependency 'fastparquet'. fastparquet is required for parquet support. Use pip or conda to install fastparquet.
Traceback (most recent call last):
  File "D:\DTéléchargement\basic-mlops-main\basic-mlops-main\train_titanic\_main_.py", line 92, in save_artifact
    artifact.to_parquet(f"./artifacts/{filename}.parquet")
  File "D:\DTéléchargement\basic-mlops-main\basic-mlops-main\env\lib\site-packages\pandas\util\_decorators.py", line 211, in wrapper
    return func(*args, **kwargs)
  File "D:\DTéléchargement\basic-mlops-main\basic-mlops-main\env\lib\site-packages\pandas\core\frame.py", line 2975, in to_parquet
    return to_parquet(
  File "D:\DTéléchargement\basic-mlops-main\basic-mlops-main\env\lib\site-packages\pandas\io\parquet.py", line 426, in to_parquet
    impl = get_engine(engine)
```


Nous avons maintenant un artefact de modèle à utiliser dans notre API.

```
C:\Windows\System32\cmd.exe
raise ImportError(
ImportError: Unable to find a usable engine; tried using: 'pyarrow', 'fastparquet'.
A suitable version of pyarrow or fastparquet is required for parquet support.
Trying to import the above resulted in these errors:
- Missing optional dependency 'pyarrow'. pyarrow is required for parquet support. Use pip or conda to install pyarrow.
- Missing optional dependency 'fastparquet'. fastparquet is required for parquet support. Use pip or conda to install f
astparquet.
[2023-01-09T12:07:45Maroc (heure d'été)] INFO in __main__: Saving artifact titanic_classifier.
[2023-01-09T12:07:45Maroc (heure d'été)] INFO in __main__: Artifact titanic_classifier successfully saved.
[2023-01-09T12:07:45Maroc (heure d'été)] INFO in __main__: Saving artifact titanic_features_scaler.
[2023-01-09T12:07:45Maroc (heure d'été)] INFO in __main__: Artifact titanic_features_scaler successfully saved.
```

4. Créer une API REST avec FastAPI

Maintenant que nous avons une classe qui gère la prédiction et que notre code python est installable par pip, nous pouvons créer une application FastAPI qui encapsulera notre modèle dans un fichier app.py. Il s'agit de l'API REST, ce qui signifie que notre modèle peut désormais recevoir des requêtes HTTP avec les fonctionnalités et y répondre avec la prédiction.

Le fichier app.py crée un modèle de passager à l'aide de Pydantic. Ceci est nécessaire pour que FastAPI valide l'entrée envoyée par la requête HTTP. Ensuite, le script crée une instance FastAPI avec une route de prédiction qui recevra une requête GET.

```
(.env) D:\DTéléchargement\basic-mlops-main\basic-mlops-main>python app.py
[2023-01-09T12:30:39Maroc (heure d'été)] INFO in titanic_model: Retrieving artifact: titanic_classifier.
[2023-01-09T12:30:40Maroc (heure d'été)] INFO in titanic_model: Artifact titanic_classifier successfully loaded.
[2023-01-09T12:30:40Maroc (heure d'été)] INFO in titanic_model: Retrieving artifact: titanic_features_scaler.
[2023-01-09T12:30:40Maroc (heure d'été)] INFO in titanic_model: Artifact titanic_features_scaler successfully loaded.
+ [32mINFO+ [0m: Started server process [+ [36m13684+ [0m]
+ [32mINFO+ [0m: Waiting for application startup.
+ [32mINFO+ [0m: Application startup complete.
+ [32mINFO+ [0m: Uvicorn running on + [1mhttp://0.0.0.0:8080+ [0m (Press CTRL+C to quit)
```

Résultat :

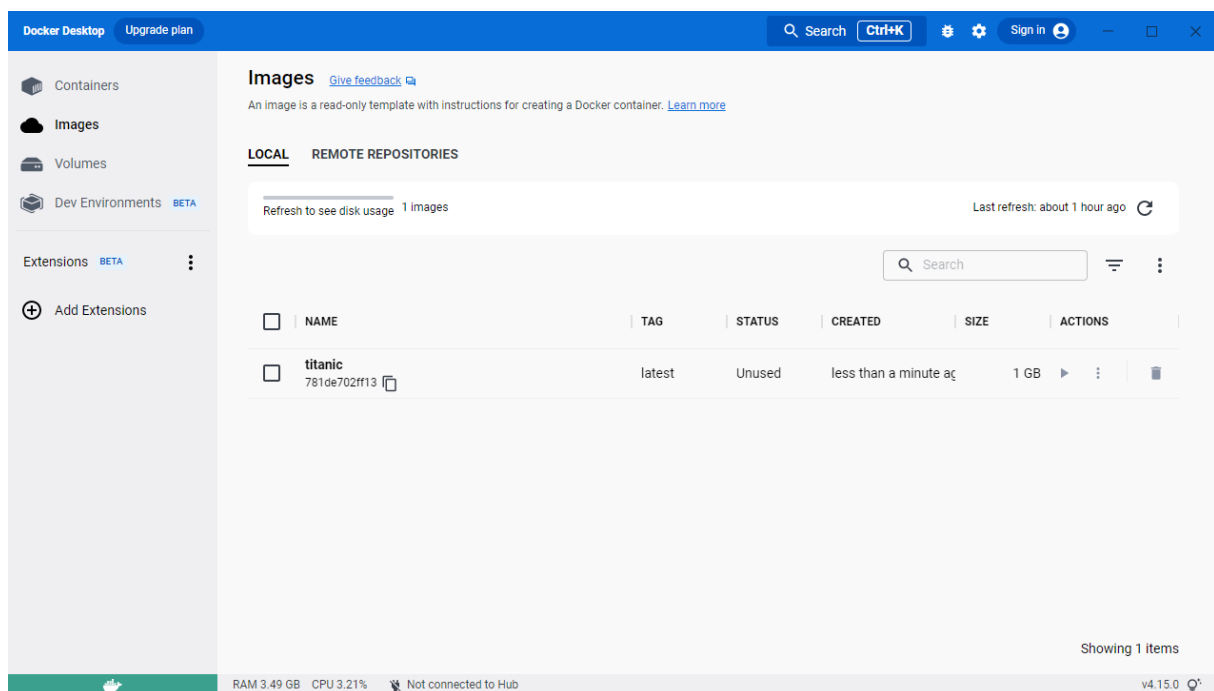
```
← → ↻ ⓘ localhost:8080
How to set up Pyth... Employabilité : défi... Développer l'emplo... Wix Website Editor... Integrating GraphQ...
{"message": "This is the Titanic API. Access /docs to check the swagger."}
```

5. Création de l'image Docker

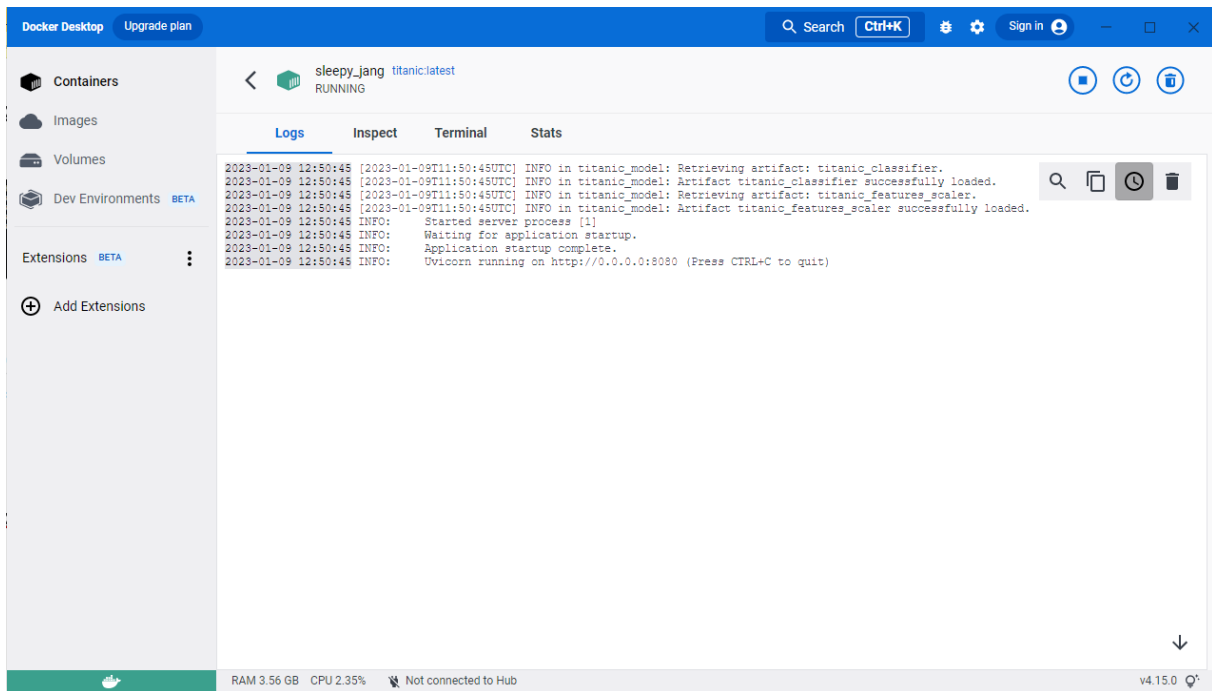
Pour l'acte final, nous allons écrire du code pour le moteur Docker afin de construire notre application et de la compiler en tant qu'image Docker. Comme c'est peut-être la première fois que vous regardez le code "Docker"

Pour tester si tout fonctionne, vous pouvez créer l'image en exécutant `docker build -t titanic`. Cette commande indique à Docker de créer une image. Le `-t titanic` marque cette nouvelle image, et le `.`. Le paramètre indique à Docker de le construire dans le dossier actuel.

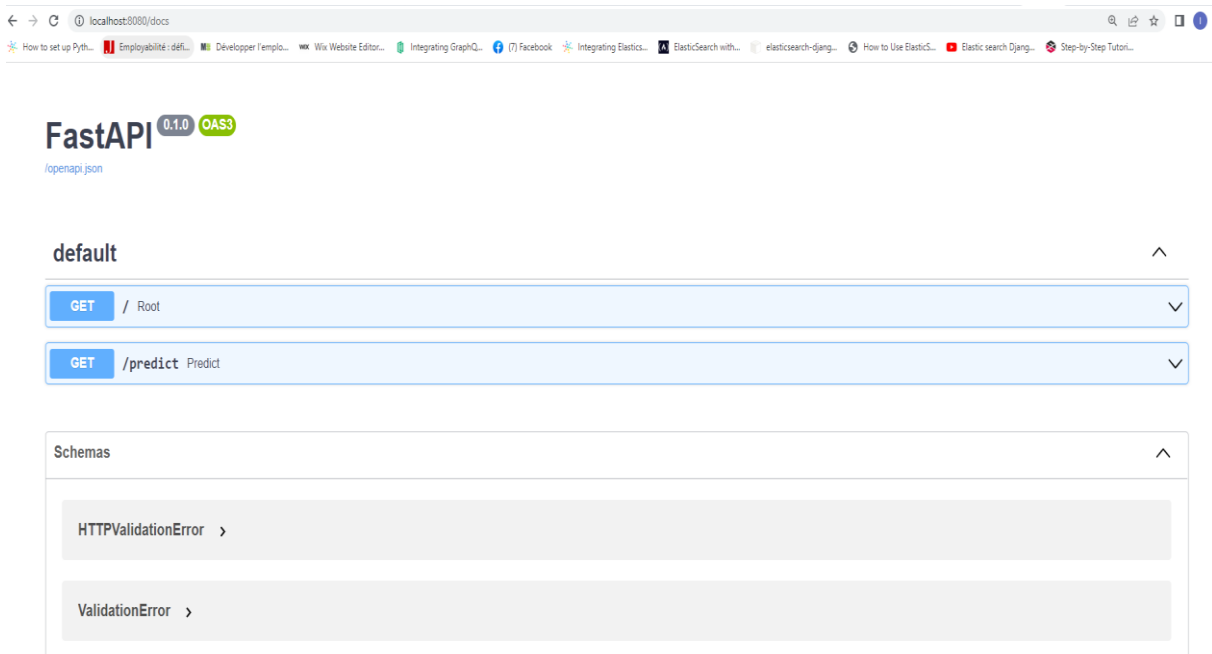
```
(.env) D:\DTéléchargement\basic-mlops-main\basic-mlops-main>docker build -t titanic .
[+] Building 381.0s (14/14) FINISHED
=> [internal] load build definition from Dockerfile                                0.0s
=> => transferring dockerfile: 328B                                              0.0s
=> [internal] load .dockerignore                                                  0.0s
=> => transferring context: 2B                                                    0.0s
=> [internal] load metadata for docker.io/library/python:3.8-slim-buster        3.6s
=> [internal] load build context                                                  43.2s
=> => transferring context: 335.30MB                                             43.2s
=> [1/9] FROM docker.io/library/python:3.8-slim-buster@sha256:4cda66a01b5571bd4f3d634b301f72e580a94b2c1ce87ead0 95.7s
=> => resolve docker.io/library/python:3.8-slim-buster@sha256:4cda66a01b5571bd4f3d634b301f72e580a94b2c1ce87ead05 0.0s
=> => sha256:4cda66a01b5571bd4f3d634b301f72e580a94b2c1ce87ead057040a6dea4a416 988B / 988B      0.0s
=> => sha256:e659e823e3fb173b0ff6a7e905042c18cc4eb117650f3ba74f2b69239cc8d4cc 1.37kB / 1.37kB 0.0s
=> => sha256:6ba145ad2ad6bd0be9290f841bdb5726ccfaf54a72bb3a25ea104ec031ee4655 7.53kB / 7.53kB 0.0s
=> => sha256:b52ebda398ed2c4602ea06056f78d45a59474ee4e2a020967251ba082424e7e2 27.14MB / 27.14MB 93.1s
=> => sha256:5ea3b77facf9f24e5cda370be6af00cfc85988f3140bee862e2529dd50e6eec1 2.78MB / 2.78MB 14.9s
=> => sha256:3cf88772a5396c14aae45be5131a2e1ccba7a5850c1041de201b52b745e34cce 11.29MB / 11.29MB 87.5s
=> => sha256:58e7c3cd0452f64ffd5e78c534d35a06366820b8e12588c506f515d266ace0dd 235B / 235B 15.3s
=> => sha256:8b23ff4b1264a9a2736e18169645d7853993db56e422cccaf22bd0bf0215b7c 3.18MB / 3.18MB 37.6s
=> => extracting sha256:b52ebda398ed2c4602ea06056f78d45a59474ee4e2a020967251ba082424e7e2 0.9s
=> => extracting sha256:5ea3b77facf9f24e5cda370be6af00cfc85988f3140bee862e2529dd50e6eec1 0.2s
=> => extracting sha256:3cf88772a5396c14aae45be5131a2e1ccba7a5850c1041de201b52b745e34cce 0.6s
=> => extracting sha256:58e7c3cd0452f64ffd5e78c534d35a06366820b8e12588c506f515d266ace0dd 0.0s
=> => extracting sha256:8b23ff4b1264a9a2736e18169645d7853993db56e422cccaf22bd0bf0215b7c 0.6s
=> [2/9] RUN mkdir -p /home/artifacts/                                           0.4s
```



L'image a été créée avec succès



Si l'image a été créée avec succès, vous pouvez l'exécuter en exécutant `docker run --rm -it -p 8080:8080`. Vous pouvez maintenant accéder à `localhost:8080/docs` dans votre navigateur pour voir le swagger généré par FastAPI.



On peut saisir les données de n'importe quel passager

La phase de test :

Ici nous avons effectuer un test avec un passager non survivé :

```
(.env) D:\DTéléchargement\basic-mlops-main\basic-mlops-main>curl -X GET "http://localhost:8080/predict?pclass=3&sex=male&age=22&fare=7.25"
false
(.env) D:\DTéléchargement\basic-mlops-main\basic-mlops-main>
```

Le resultat dans la plateforme FastAPI

Responses		
Code	Description	Links
200	Successful Response	No links
Media type: application/json		
Controls Accept header:		
Example Value Schema		
"string"		
422	Validation Error	No links
Media type: application/json		
Example Value Schema		
{ "detail": [{ "loc": ["string", 0], "msg": "string", "type": "string" }] }		

Il nous donne une reposnse par true ou bien false alors que le passager soit survivre ou non.

Ici au contraire nous avons tester avec les données d'un passager survivé :

```
(.env) D:\DTéléchargement\basic-mlops-main\basic-mlops-main>curl -X GET "http://localhost:8080/predict?pclass=3&sex=female&age=0&fare=7.75"
true
(.env) D:\DTéléchargement\basic-mlops-main\basic-mlops-main>
```

Voilà le résultat prédit sur la plateforme

Responses

Curl

```
curl -X 'GET' \
  'http://localhost:8080/predict?pclass=3&sex=female&age=0&fare=7.75' \
  -H 'accept: application/json'
```

Request URL

```
http://localhost:8080/predict?pclass=3&sex=female&age=0&fare=7.75
```

Server response

Code	Details
200	<div><div>Response body</div><pre>true</pre></div> <div><div>Response headers</div><pre>content-length: 4 content-type: application/json date: Mon, 09 Jan 2023 12:58:29 GMT server: uvicorn</pre></div>

Conclusion

Dans ce projet, nous avons implémenté une seule page web pour gérer l'entrée et la sortie d'un modèle machine Learning.

Ce travail nous a permis de découvrir et de nous familiariser avec différentes méthodologies et technologies : MLOPS, Docker, Python et autres.

Ce projet a été d'une grande importance. Il nous a permis de découvrir le monde de la Machine Learning, il nous permet de découvrir de nouvelles technologies et il nous permet également de savoir comment gérer notre temps et partager des idées avec les autres collègues.

Et pour finir nous pouvons dire que ce travail nous a permis de prendre conscience de plusieurs points importants dans la réalisation d'un projet, tout d'abord sur le plan organisationnel, nous avons appris comment répartir les tâches tout en gardant l'homogénéité du projet, sans nier le respect du planning pour aboutir à notre but dans le délai énoncé. Et sur le plan humain, nous avons su l'importance de travailler en harmonie en groupe, d'adopter d'autres perspectives, et finalement s'ouvrir sur d'autres domaines et horizons.