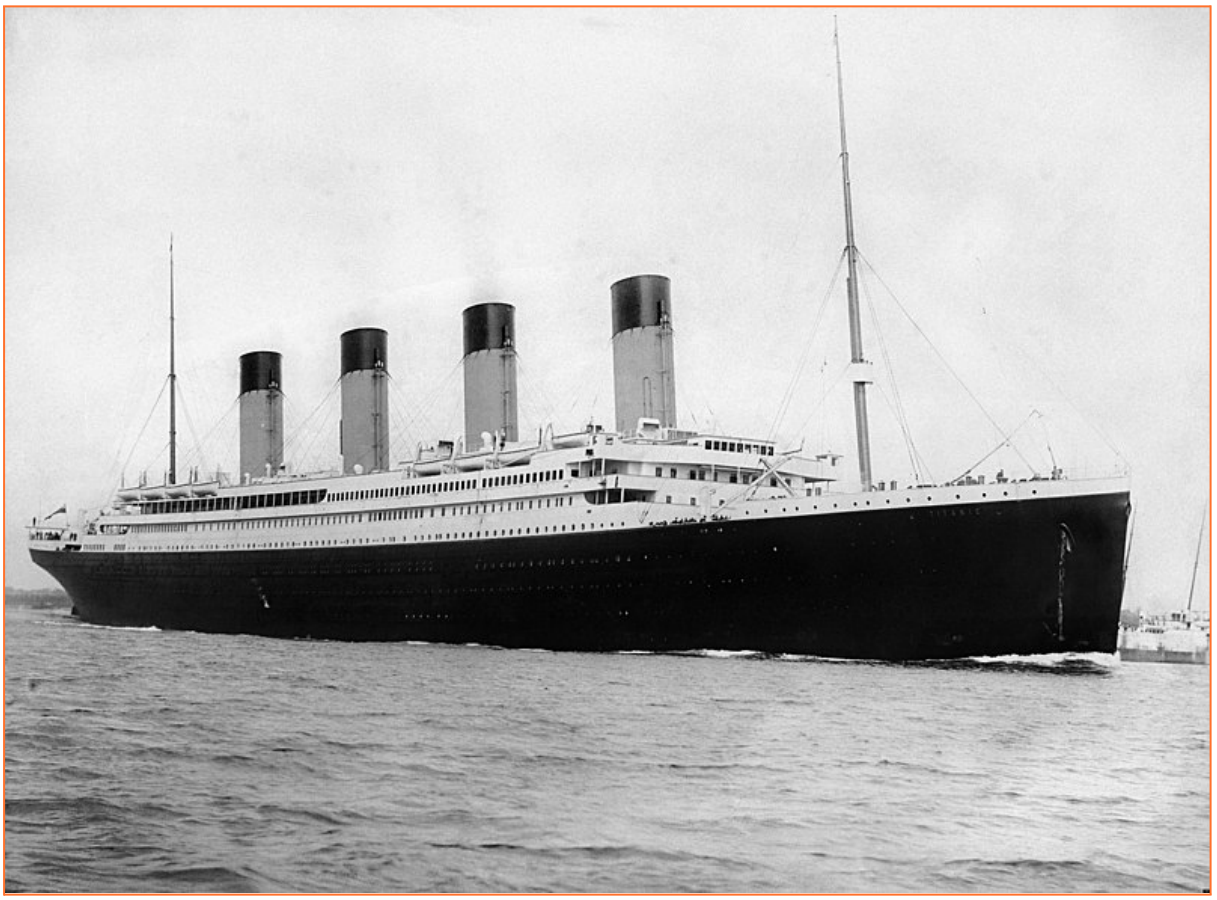

Cahier de recette

L2I1 : Machine Learning from disaster

—

Hongxiang Lin
Melissa Merabet
Mathieu Antonopoulos
Timothé Miel
20/02/2023

Université Paris Cité

Sommaire

1. Introduction

Le projet vise à construire un modèle prédictif par Machine Learning, en utilisant les données relatives aux passagers du *Titanic*, afin de prédire qui aurait le plus de chances de survivre au naufrage. Le projet se base sur un ensemble de données, '*train.csv*', contenant les informations sur les passagers tels que le nom, l'âge, le sex, la classe socio-économique, etc. On dispose également d'un fichier 'test.csv' permettant d'évaluer la précision du modèle.

2. Concepts de base

Dans ce projet, nous allons utiliser des ensembles de données similaires qui comprennent des informations sur les passagers comme le nom, l'âge, le sex, la classe socio-économique, la situation familiale, le numéro de cabine, le prix du ticket, le port d'embarquement, etc.

Un dataset est intitulé 'train.csv' et l'autre est intitulé 'test.csv'.

- 'train.csv' contiendra les détails d'un sous-ensemble de passagers à bord (891 pour être exact) et surtout, révélera s'ils ont survécu ou non.
- 'test.csv' permettra d'évaluer les performances du modèle.

En utilisant les modèles que nous avons trouvés dans les données train.csv, nous allons prédire si les 418 autres passagers à bord (trouvés dans test.csv) ont survécu.

Lors de l'analyse des données, nous allons utiliser divers outils mathématiques tels que matplotlib et seaborn afin d'analyser les résultats et mieux déterminer le modèle que nous utiliserons pour prévoir la survie des passagers à bord.

Enfin, nous utiliserons Streamlit pour construire une application web permettant de mieux visualiser les résultats de notre analyse de données.

Notre projet sera développé avec les outils suivants :

- Jupyter Notebook (avec Python dans l'environnement d'Anaconda 3)
- Serveur SVN (via le logiciel client Tortoise SVN.)
- Numpy
- Pandas (data management) : https://pandas.pydata.org/

- Seaborn (https://seaborn.pydata.org/examples/index.html
- Matplotlib (https://matplotlib.org/stable/gallery/index)
- Les différents modèles de Machine Learning : Linear Models, etc.
- Streamlit (https://streamlit.io/)
- Plotly (Plotly: Low-Code Data App Development)
- Python (version 3.9)
- Scikit-learn pour le Machine Learning. (https://scikit-learn.org/stable/index.html)

3. Description de la fourniture

Le projet sera rendu sous 2 formes (interface web optionnelle):

- Premièrement, dans un fichier trunk :
  - Un répertoire contenant les documents 'test.csv' et 'train.csv' fournis par Kaggle.
  - Un fichier 'pre_processing.py' : Ce document exécute une fonction python permettant d'appliquer la fonction de prétraitement aux sets données 'test.csv' et 'train.csv'.
  - Un fichier 'run_model.py' : Ce document exécute une fonction python permettant de faire tourner le modèle sur les sets de données prétraité. Ainsi, ce code permettra de faire tourner le modèle sur les données d'entraînement afin de tester son efficacité. De plus, un fichier de soumission sera créé. Ce fichier 'soumission.csv' donnera les prédictions du modèle sur les données test. Le format du fichier est celui demandé par Kaggle et permet donc d'obtenir un score.
  - Un fichier 'Conception_Model.ipynb' : Notebook Jupyter commentant toutes les étapes de pré-processing, entraînement et soumission. De plus ce Notebook retrace une partie des analyses et recherches réalisées au cours du projet afin d'assurer une meilleure interprétation des résultats.
  - Un fichier 'Conception_Model.pdf': version pdf de 'Conception_Model.ipyb'

- Un fichier texte ‘ReadMe.txt’ expliquant à l'utilisateur le contenu de chaque fichier et fournissant des indications nécessaires à l'exécution du modèle.
- Un fichier texte ‘Requirements.txt’ contenant les bibliothèques et environnements nécessaires à l'exécution des programmes pyhton.
- Un répertoire ‘Resultats’ où seront stockés les résultats après exécution.
- Un fichier ‘guide.pdf’, cette documentation vise à fournir des instructions détaillées sur l'exécution des fichiers de prétraitement (pre_processing.py) et d'exécution du modèle (run_model.py) sur Windows.
- Un répertoire ‘Web_App’ contenant le code de l’interface Webb.

- Application web streamlit : Application web accessible en ligne pour tous gratuitement depuis n’importe quelle plateforme (mobile, pc), permettant d'entrer les données relatives à un utilisateur et renvoyant la prédiction du modèle. Cette interface permet une utilisation ludique du modèle : l’utilisateur peut voir s' il aurait survécu ou non au naufrage du Titanic selon notre modèle. De plus, en testant le modèle avec différentes données, l’utilisateur peut identifier certains facteurs de survie et ainsi comprendre les bases du modèle.

4. Conformité

I. Conformité du modèle

Le modèle doit être en mesure de prédire le survit d’un passager à partir de ses données. L’objectif est d’obtenir un taux de précision de 0,8. La fonction de pré-processing doit rendre lisible par le modèle les sets de données fournis par Kaggle. Le fichier de soumission doit être conforme aux règles du concours Kaggle.

II. Conformité du Notebook Jupyter

Le Notebook Jupyter doit retracer toutes les étapes de conception du modèle de façon lisible et accessible. On doit y trouver des extraits de codes python commentés ainsi que des illustrations. Ainsi ce document facilite la compréhension du modèle et de sa conception.

III. Conformité de l'application web

L'application doit être accessible en ligne gratuitement depuis un navigateur web. Elle doit également être disponible sur mobile. Celle-ci doit permettre de rentrer des données de passagers (Sex, âge, classe etc..) et renvoyer la prédiction du modèle. L'interface doit être simple et intuitive.

5. Conformité de la documentation

La documentation doit permettre à tout utilisateur de comprendre le contenu de chaque fichier et de tester le modèle avec les programmes python à disposition.

Le guide d'exécution doit permettre à l'utilisateur d'installer les librairies et programmes nécessaires. De plus ce document doit spécifier les prérequis hardware et software à l'exécution des fichiers.

Le guide d'installation conda doit permettre à l'utilisateur l'installation de 'conda' afin de pouvoir utiliser l'interface de notebook Jupyter pour pouvoir exécuter les codes python du notebook. Ce guide doit également préciser les prérequis à l'utilisation de l'environnement conda.

6. Procédures de tests

I. Test du modèle sur les données d'entraînement

Le modèle doit renvoyer un score supérieur à 0.8 (objectif fixé au début du projet) sur les données d'entraînement. Le score sur les données d'entraînement doit apparaître dans le répertoire ‘Resultats’.

II. Test du modèle sur les données Test

Pour vérifier que le modèle est fonctionnel, le score obtenu sur les données test doit être proche de celui obtenu sur les données d'entraînement. Les deux scores doivent être cohérents pour que le modèle soit efficace.

III. Test de la transférabilité du modèle

Le modèle doit être transférable sur de nouvelles données. Nous allons donc tester le modèle sur quatres passagers fictifs correspondant aux quatres membres du groupe. Le modèle devra renvoyer une prédiction cohérente. Cette prédiction sera disponible dans le répertoire ‘Resultats’.

IV. Test de l’application Web

L’application doit permettre de renvoyer une prédiction de survie à partir d'informations données par l’utilisateur. On va donc tester l'interface sur une série de données afin de vérifier la cohérence des résultats et la qualité de l’interface. test sur

.