

The Machine Learning Platform – Meidi KADRI

Perform your data analyst and machine learning without any line of code !

Le concept :

The **Machine Learning Platform** (MLP) est une application Flask permettant de faire de l'analyse de données ainsi que du Machine Learning (ML) via une interface graphique.

L'idée est de pouvoir effectuer toutes les étapes de ML sur une plateforme, **sans aucune ligne de code**.

Machine Learning Platform

How it works Pricing Sign-Up Sign-In

Perform your Data Analysis and Try some Machine Learning models
Without any line of code !

Machine Learning Platform

Dataset Upload Data Exploration Data Visualization Model Training Prediction & Export Settings Disconnect

Dataset Import

titanic.csv | 14 Columns | 1310 Rows

	pclass	survived	name	sex	age	sibsp	parch	ticket	fare	cabin	embarked	boat	body	home.dest
0	1.0	1.0	Allen, Miss. Elisabeth Walton	female	29.0000	0.0	0.0	24160	211.3375	B5	S	2	NaN	St Louis, MO
1	1.0	1.0	Allison, Master. Hudson Trevor	male	0.9167	1.0	2.0	113781	151.5500	C26	S	11	NaN	Montreal, PQ / Chesterville, ON
2	1.0	0.0	Allison, Miss. Helen Loraine	female	2.0000	1.0	2.0	113781	151.5500	C26	S	NaN	NaN	Montreal, PQ / Chesterville, ON
3	1.0	0.0	Allison, Mr. Hudson Joshua Creighton	male	30.0000	1.0	2.0	113781	151.5500	C26	S	NaN	135.0	Montreal, PQ / Chesterville, ON
4	1.0	0.0	Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	female	25.0000	1.0	2.0	113781	151.5500	C26	S	NaN	NaN	Montreal, PQ / Chesterville, ON
5	1.0	1.0	Anderson, Mr. Harry	male	48.0000	0.0	0.0	19952	26.5500	E12	S	3	NaN	New York, NY
			Andreas, Miss. Kornelia											

Summary

	pclass	survived	age	sibsp	parch	fare	body
count	1309.000000	1.009.000000	1046.000000	1309.000000	1309.000000	1308.000000	121.000000
mean	2.294862	0.381971	29.881135	0.498854	0.385027	33.295479	160.809917
std	0.837836	0.486055	14.413500	1.041658	0.885560	51.758668	97.696922
min	1.000000	0.000000	0.166786	0.000000	0.000000	0.000000	1.000000
25%	2.000000	0.000000	21.000000	0.000000	0.000000	7.895800	72.000000
50%	3.000000	0.000000	28.000000	0.000000	0.000000	14.454200	155.000000
75%	3.000000	1.000000	35.000000	1.000000	0.000000	31.275000	256.000000
max	3.000000	1.000000	80.000000	8.000000	9.000000	512.329200	328.000000

Dataframe Informations

	Info
0	<class 'pandas.core.frame.DataFrame'>
1	RangeIndex: 1310 entries, 0 to 1309
2	Data columns (total 14 columns):
3	pclass 1309 non-null float64
4	survived 1309 non-null float64

Missing values

	Missing Value Count
pclass	1
survived	1
name	1
sex	1
age	254
sibsp	1
parch	1

MLP est un projet imaginé en début d'année. Lors de la formation je me suis aperçu que le process pour faire du ML était sensiblement le même :

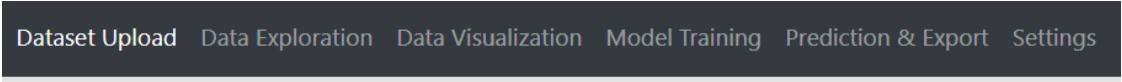
- Import et lecture d'un fichier.
- Exploration des données.
- Visualisation des datas pour mieux les comprendre.
- Sélections des features qui nous intéressent pour préparer le modèle.
- Entraînement du modèle avec un split en test_set et train_set et une standardisation si besoin.
- Évaluation du modèle et affinage des hyperparamètres.
- Prédiction.
- Export pour une utilisation future...

Bien sûr, ces étapes peuvent s'effectuer de plusieurs façons. Et il nous faut essayer différentes librairies et modules via de nombreuses lignes de codes sur Jupyter pour observer les données, les comprendre et les faire parler... Cela peut prendre énormément de temps. Pour parfois s'apercevoir finalement que l'on est sur une mauvaise piste !

C'est pourquoi j'ai eu l'idée de développer un outil qui me permettrait de gagner en productivité. Remplacer une ou plusieurs lignes de codes par un "simple" bouton sur une interface nous permettrait d'être plus rapide sans perdre en efficacité.

Le workflow et Fonctionnalités :

Dans cette version que je qualifie de Beta, le workflow est orienté par le menu de l'application. Invitant ainsi l'utilisateur à passer par chaque page qui représente une étape du process de ML.



Dataset Upload Data Exploration Data Visualization Model Training Prediction & Export Settings

Dataset Upload :

Ici, l'utilisateur peut récupérer un dataset présent sur son disque dur ou bien utiliser l'un des 3 datasets de la librairie scikit learn importées directement dans l'app.

Une fois téléchargé, le dataset est affiché dans une table ainsi que diverses informations utiles comme le nom, nombre de lignes et de colonnes. Trois tableaux viennent apporter d'autres informations utiles sur le jeu de données comme un describe, les infos sur les types de datas de chaque colonne et un résumé du nombre de valeurs manquantes par colonne.

Dataset Exploration :

Explorer un jeu de données est d'une importance capitale. De nombreuses options s'offrent à l'analyste. Malheureusement, il ne m'aura pas été possible de toutes les proposer dans cette version.

Cependant, comme dans chacune des étapes de l'application, mon objectif était d'inclure un minimum de fonctionnalités afin de faire la démonstration du concept de MLP.

Ainsi sont disponibles les fonctionnalités suivantes :

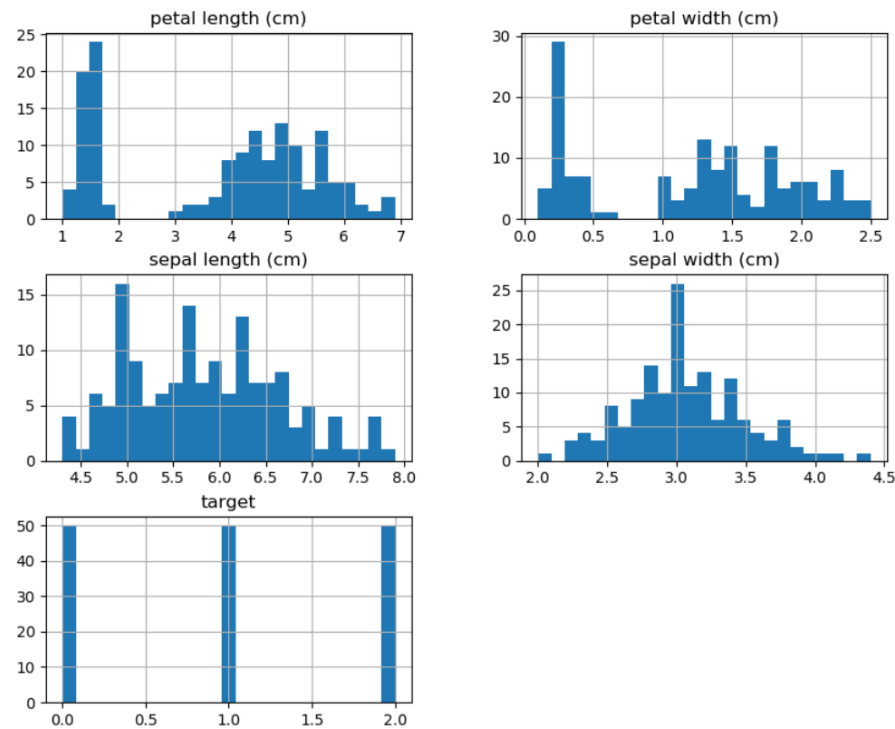
- **Drop NaN** : Supprimer les lignes comportant des valeurs manquantes (NaN) sur une ou plusieurs colonnes.
- **Fill NaN Values** : Remplacer les NaN d'une colonne soit par la moyenne, la médiane ou une valeur définie par l'utilisateur.
- **Drop duplicates** : La possibilité de supprimer les lignes en double dans le dataframe en filtrant sur une colonne sélectionnée par le user.
- **Drop Column** : Suppression d'une ou plusieurs colonnes sélectionnées par le user.

Data Visualization

[Previous step](#)[Save Graph as Image](#)[Save Page as PDF](#)[Next step](#)

Dataset : iris_sklearn_dataset

Histogram Graph.



© 2020 Copyright: Meidi KADRI

Comme une image vaut mieux que 1000 mots, les graphiques sont largement utilisés pour observer et trouver ou confirmer des relations entre plusieurs données. C'est ici que nous pourrions afficher les graphes suivants :

- **Heatmap** : Afin d'identifier les valeurs manquantes ou nulles.
- **Histogramme** : La fonction Hist nous permet d'observer les distributions de tout le dataset ou d'une colonne en particulier.
- **Mosaic** : La fonction "mosaic" nous aide à visualiser la répartition de deux ou plusieurs variables qualitatives. Par exemple une variable explicative et la target.
- **Plot** : Dans cette version un simple scatter plot de seaborn est disponible.
- **Correlation Matrix** : Une matrice de corrélation afin d'observer les corrélation entre les varibales.

Model Training :

C'est ici que nous allons pouvoir faire tourner nos modèles.

Grâce à l'analyse des variables précédentes, nous pouvons désormais sélectionner une cible [Target Y] ainsi que les variables pour entraîner le ou les modèles.

Une partie preprocessing sera également disponible pour gérer les dummies et les standardisations.

Nous pouvons définir la répartition de notre split avec la fonction **test_size**, puis il nous reste plus qu'à choisir les modèles que nous souhaitons essayer. Dans cette version sont disponibles :

- **Régression** : Logistique et Linéaire.
- **Classification** : KNN, KMeans, Random Forest Classifier, Decision Tree et SGDClassifier.

Un tableau affiche les scores des modèles entraînés pour que l'utilisateur puisse choisir le plus performant à exporter.

Model	Run time	Accuracy Score
Linear Regression	00:00:00	0.382
Logistic Regression	00:00:00	0.803
Random Forest Classifier	00:00:00	0.962
Decision Tree Classifier	00:00:00	0.975

Prédiction & Export :

Dans cette page, le user peut observer la pertinence des prédictions du modèle choisit.

Sera disponible une **Matrice de confusion** ainsi qu'un tableau affichant les prédictions et des outils comme la **Courbe RoC** pour optimiser les hyperparamètres du modèle.

Sauvegarde des fichiers :

Sur chaque page, l'utilisateur peut enregistrer divers fichiers dans une base de données afin de garder une trace de son travail.

- **Tableaux** : En format csv.
- **Graphiques** : En format png.
- **Modèles** : En format pikle.
- **Pages** : En format pdf.

User :

Pour accéder à la plateforme, l'utilisateur doit s'enregistrer. Pour cela j'ai utilisé l'API d'authentification de Google **Firestore**. Cette librairie est assez puissante et permet de gérer les modifications/ pertes de mots de passe, les connexions aux bases de données et la sécurité.

Le `user_id` permet ensuite de créer des bases de données directement liées à un utilisateur précis. **Firestore database** est une base de données NoSql réactive et plutôt robuste. Mais surtout facile à utiliser.

C'était un choix pertinent pour mon projet car Firestore offre aussi un service pour stocker les images, csv, pdf et modèles grâce à **Firestore storage**.

Technologies utilisées :

Flask:

Il s'agit d'une application Flask. Le code contient donc une partie en Python et une partie HTML. A cela s'ajoute un fichier CSS et Javascript/ Ajax pour ajouter du design et du dynamisme au projet. Pour ajouter du style, j'ai également utilisé Bootstrap.

Firestore :

Comme indiqué plus haut, pour gérer la base de données, le stockage et les authentifications j'ai utilisé la librairie Python **Pyrebase** qui permet d'utiliser l'API de Google Firestore.

- **Authentication** : Permet de gérer les connexions au site, de créer des identifiants utilisateur réutilisables pour les bases de données. Un système de gestion des pertes ou modifications de mots de passe est également inclus.
- **Database**: Base de données NoSQL.
- **Storage** : Base de données de stockage pour les graphiques, les pages et les tableaux du projet.

Scikit Learn :

Impossible d'éviter la librairie `sklearn` quand on traite de ML. J'ai donc utilisé l'import de trois datasets (Iris, Diabetes, Breast Cancer) ainsi que les modèles et les métriques pour l'évaluation des performances.

Retour d'expérience :

Ce projet a été très enrichissant pour moi. Il représente une **Proof Of Skills** assez intéressante qui démontre ma capacité à imaginer des outils de productivité, et à trouver des solutions pour les concevoir. Je suis assez satisfait du résultat même si je reste mobiliser pour faire évoluer l'application. Cette version est bien trop basique pour le moment.

En effet, je pense avoir sous évalué la charge de travail et le délai nécessaire pour mener à bien la mission, et intégrer la totalité des fonctionnalités que j'avais en tête.

J'ai également perdu du temps sur certaines problématiques comme la gestion de l'asynchrone avec Ajax ou XML. Je cherchais une dynamique aussi efficace que RShiny sans y parvenir. Je pense que cela viendra avec les futures mises à jour mais mon entêtement aura gaspillé plusieurs heures sur des fonctionnalités plus pertinentes. C'est un élément que je retiens pour mes futurs projets, bien délimiter le timing et surtout, garder les extras pour la fin si le temps reste disponible. Il me faut mieux prioriser les tâches.

Il s'agit ici d'un projet de démonstration du concept de **Machine Learning Platform**. Il reste encore beaucoup de travail pour que cette application soit à la hauteur de ce que je souhaite. Je continuerai à y consacrer du temps pour l'optimiser autant que possible.