

# Mise en place et déploiement d'un modèle de machine

REALISE PAR

- Habib Aidara
- René Lothaire BAZIE
- Moussa Steve SANOGO

SOUS LA SUPERVISION DE

Dr. Ndiaye



Année universitaire : 2022/2023

# SOMMAIRE

SOMMAIRE .....	2
TABLE DES ILLUSTRATIONS .....	3
INTRODUCTION.....	4
I.  Environnement de travail utilisé .....	5
II. Pipeline de construction du modèle de machine Learning .....	6
III. Déploiement du modèle.....	15
TABLE DES MATIERES .....	27

## TABLE DES ILLUSTRATIONS

Figure 1: Présentation de l'environnement de travail PyCharm.....	5
Figure 2: Distribution de la cible 'churn'.....	7
Figure 3: Fonctionnalités de PyCaret.....	8
Figure 4: Phase de préparation des données.....	10
Figure 5: Comparaison des modèles de machine learning.....	10
Figure 6: Création et Tuning du meilleur modèle.....	11
Figure 7: Matrice de confusion.....	12
Figure 8: Graphe d'importance des variables.....	12
Figure 9: Grille d'évaluation du modèle offerte par la fonction evaluate_model.....	13
Figure 10: Finalisation du modèle.....	13
Figure 11 : Sauvegarde du modèle.....	14
Figure 12: Pipeline de construction de notre modèle de machine Learning.....	14
Figure 13: Terminal de Anaconda.....	16
Figure 14: Installation des packages requis.....	16
Figure 15 : Configuration de l'environnement virtuel dans PyCharm.....	17
Figure 16: Test du déploiement avec FastAPI.....	20
Figure 17: Online prediction with streamlit.....	21
Figure 18: Batch prediction with streamlit.....	22
Figure 19: Roadmap of Batch prediction with streamlit.....	22
Figure 20: Roadmap of datavisulization with streamlit.....	23
Figure 21: Types de graphes offerts dans l'API.....	23
Figure 22: Roadmap for the acquisition of information on a dataset.....	24
Figure 23: Déploiement avec Flask.....	26

# INTRODUCTION

L'Intelligence Artificielle (IA) est le domaine de l'informatique qui étudie comment faire faire à l'ordinateur des tâches pour lesquelles l'homme est aujourd'hui encore le meilleur. Grâce à l'IA les systèmes informatiques sont capables d'exécuter des tâches complexes qui nécessitent normalement une intelligence humaine, telles que la reconnaissance de la parole, la traduction de langues et la prise de décisions.

L'histoire de l'IA remonte aux années 1950, lorsque les scientifiques tels que Alan Turing, John McCarthy, Claude Shannon ont commencé à explorer les moyens de faire en sorte que les ordinateurs puissent imiter les capacités cognitives des êtres humains. Depuis lors, l'IA a connu des avancées considérables, notamment avec le développement de l'apprentissage automatique, qui permet aux ordinateurs de s'adapter et de s'améliorer en utilisant des données d'entraînement.

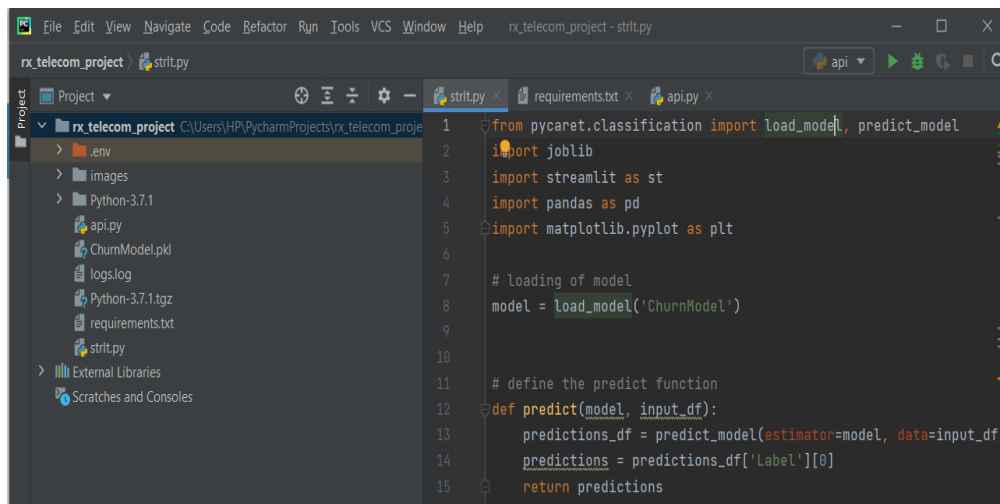
Elle a de nombreuses applications dans le domaine des télécommunications, notamment en matière de la reconnaissance de la voix, de la compréhension du langage naturel, la prévention des pannes et l'optimisation des performances du réseau. L'une de ces applications et non des moindres est la prédiction de churn. C'est-à-dire l'identification des clients qui sont les plus susceptibles de quitter un service de télécommunication. En effet l'acquisition de nouveaux clients étant plus onéreuse que la fidélisation de leurs clientèles, les opérateurs télécoms ont recours à cette méthode afin de préserver leur base d'abonnés.

A l'issue de notre cours « application de l'Intelligence Artificielle au réseaux télécoms » nous avons eu pour projet la construction et le déploiement d'un modèle de machine Learning sur la prédiction de churn par l'usage « *PyCaret* » et de « *FastAPI* ». Ce rapport présente de façon détaillée les différentes étapes et procédures qui ont permis la réalisation de ce travail.

# I. Environnement de travail utilisé

## 1. PyCharm

PyCharm est un environnement de développement intégré (IDE) Python dédié, fournissant une large gamme d'outils essentiels pour les développeurs, étroitement intégrés pour créer un environnement pratique pour le développement productif de Python, du Web et de la science des données. PyCharm est disponible en trois éditions : la version Community, Professional, et Edu. Les versions Community et Edu sont gratuites et open source, quant à la version Professional elle est payante. Pour notre projet nous avons utilisé la version Community, télécharger PyCharm [ici](#)



*Figure 1: Présentation de l'environnement de travail PyCharm*

## 2. Anaconda

Anaconda est un ensemble d'outils open source pour la science des données et l'analyse numérique, qui comprend une distribution de Python préinstallée avec de nombreuses bibliothèques scientifiques populaires. Il offre des applications telles que Jupyter Notebook, Spyder, RStudio, Glueviz et Orange, et permet aussi d'installer de nouvelles bibliothèques Python. Anaconda fournit une interface graphique utilisateur et un terminal de commande : Anaconda Navigator, Anaconda Prompt.

Nous avons utilisé le prompt d'anaconda afin de créer un environnement virtuel nécessaire au déploiement de notre modèle de classification. Il nous également permis de créer un notebook pour la construction de notre modèle.

## II. Pipeline de construction du modèle de machine Learning

### 1. Présentation du jeu de données

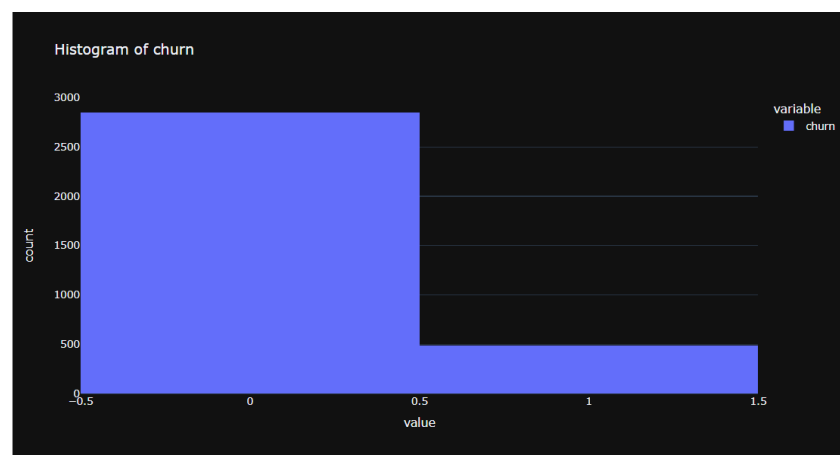
Le dataset soumis à notre étude provient de la plateforme web Kaggle. Il s'agit d'un fichier CSV contenant **3333 lignes** et **21 colonnes** dont la colonne cible est '**churn**'. Chaque ligne représente un client, chaque colonne contient les attributs du client. La colonne '**churn**' contient la valeur des étiquettes de classe, '**True**' signifie que le client va se désabonner, sinon '**False**'

Les colonnes de notre jeu de données sont les suivant :

Columns Names	Description
<b>state</b>	
<b>account length</b>	Depuis combien de temps le client est abonné l'entreprise
<b>area code</b>	Le préfixe du numéro de téléphone
<b>phone number</b>	Le numéro de téléphone
<b>international plan</b>	Le plan de numérotation international (country code +national number)
<b>voice mail plan</b>	Plan de messagerie vocale
<b>Number vmag messages</b>	Nombre de messages vocaux
<b>total day minutes</b>	Durée total des minutes d'appel journalier
<b>total day calls</b>	Nombre total des appels journaliers
<b>total day charge</b>	Total des frais d'appel journalier
<b>total eve minutes</b>	Durée totale d'appel du soir
<b>total eve calls</b>	Nombre total d'appel du soir
<b>total eve charge</b>	Total des frais d'appel du soir
<b>total night minutes</b>	Durée totale d'appel de nuit

<b>total night calls</b>	Nombre total des appels de nuit
<b>total night charge</b>	Total des frais d'appel de nuit.
<b>total intl minutes</b>	Durée Totale des appels internationaux
<b>total Intl calls</b>	Nombre total des appels internationaux
<b>total Intl charge</b>	Total des frais des appels internationaux
<b>customer service calls</b>	Appel du service clientèle
<b>churn</b>	Attrition

## 2. Analyse exploratoire



*Figure 2: Distribution de la cible 'churn'*

Comme on peut le constater notre jeu de données contient plus de lignes **churn : 'False'** que **churn**, nous devons équilibrer les données avant de faire des prédictions. Sinon le modèle aura tendance à prédire la classe majoritaire « **churn = '0'** » avec une précision élevée, mais aura des difficultés à prédire la classe minoritaire avec précision. Pour l'équilibrage des données nous utiliserons la technique **SMOTE (Synthetic Minority Over-sampling Technique)** une technique de suréchantillonnage des minorités synthétiques.

### 3. Présentation de PyCaret



*Figure 3: Fonctionnalités de PyCaret*

PyCaret est une librairie d'apprentissage automatique open source basée sur Python. Il est simple, facile à utiliser et fournit un code simplifié qui peut diviser par cent votre nombre de lignes de code.

Il offre une variété de fonctionnalités pour faciliter le développement de modèles de machine Learning, telles que :

- ✚ **Préparation des données** : en fournissant des outils pour nettoyer et préparer les données, tels que la suppression des valeurs manquantes et la normalisation des données.
- ✚ **Sélection de modèles** : il peut automatiquement essayer plusieurs algorithmes de machine Learning pour trouver celui qui convient le mieux à vos données.
- ✚ **Évaluation des modèles** : PyCaret peut automatiquement évaluer la performance des modèles en utilisant des métriques telles que l'erreur quadratique moyenne et la précision.
- ✚ **Visualisation des résultats** : PyCaret fournit des graphiques et des tableaux pour visualiser les résultats des modèles, ce qui aide à comprendre comment les modèles fonctionnent et à déterminer les améliorations nécessaires.



En utilisant PyCaret, les développeurs peuvent économiser du temps et de l'effort en automatisant certaines tâches fastidieuses et en fournissant une interface facile à utiliser pour les autres tâches. PyCaret peut aider les développeurs à créer des modèles de machine Learning plus rapidement et avec moins d'erreurs, ce qui peut augmenter la qualité des modèles et la vitesse du développement.

#### 4. Préparation des données

Grâce à la fonction « *setup()* » de PyCaret nous allons effectuer les étapes de prétraitement des données et de préparation du modèle pour l'apprentissage.

Cette fonction « *setup()* » prend plusieurs arguments, notamment:

- + **'data'**: l'ensemble de données à utiliser pour la classification ;
- + **'target'**: la variable cible à prédire (dans ce cas "churn") ;
- + **'fix\_imbalance'** : sa valeur à **True** permet l'équilibrage des deux classes(0, 1) par usage de la méthode SMOTE ;
- + **'session\_id'**: un nombre entier utilisé pour initialiser l'environnement PyCaret ;
- + **'categorical\_features'**: une liste des noms de colonnes dans l'ensemble de données qui doivent être traitées comme des variables catégorielles ( "state", "area\_code", "international\_plan", "voice\_mail\_plan" et "customer\_service\_calls").

Plus spécifiquement les tâches effectuées par cette fonction sur notre jeu de données sont :

- + la séparation des données en ensembles d'entraînement et de test ;
- + l'encodage des variables catégorielles ;
- + la gestion des valeurs manquantes ;
- + la normalisation des données ;
- + la création d'un pipeline de traitement des données.

```
from pycaret.classification import *

s = setup(data= df, target='churn', session_id=123, fix_imbalance=True, categorical_features=
          ["state", "area_code", "international_plan", "voice_mail_plan", "customer_service_calls"] )
```

*Figure 4: Phase de préparation des données*

## 5. Formation et sélection de modèles

Maintenant que la préparation des données est terminée, commençons le processus de formation à l'aide de la fonctionnalité : « *compare\_model* ». Cette fonction entraîne tous les algorithmes disponibles dans la bibliothèque de modèles et évalue plusieurs mesures de performance à l'aide de la validation croisée.

```
# compare all models
best_model = compare_models()
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
<b>xgboost</b>	Extreme Gradient Boosting	0.9421	0.9055	0.7103	0.8825	0.7849	0.7520	0.7592	1.2240
<b>lightgbm</b>	Light Gradient Boosting Machine	0.9421	0.9129	0.6988	0.8913	0.7805	0.7480	0.7570	0.5770
<b>gbc</b>	Gradient Boosting Classifier	0.9237	0.9020	0.6931	0.7753	0.7290	0.6850	0.6882	1.0720
<b>rf</b>	Random Forest Classifier	0.9061	0.8954	0.4607	0.8567	0.5860	0.5399	0.5786	0.4100
<b>dt</b>	Decision Tree Classifier	0.8975	0.8202	0.7100	0.6432	0.6720	0.6117	0.6146	0.0770
<b>et</b>	Extra Trees Classifier	0.8877	0.8825	0.3953	0.7400	0.5107	0.4542	0.4850	0.3910
<b>ada</b>	Ada Boost Classifier	0.8602	0.8336	0.4868	0.5429	0.5113	0.4303	0.4323	0.3560
<b>dummy</b>	Dummy Classifier	0.8504	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0330
<b>lr</b>	Logistic Regression	0.8229	0.8412	0.7621	0.4483	0.5631	0.4616	0.4884	2.6690
<b>lda</b>	Linear Discriminant Analysis	0.8208	0.8368	0.7391	0.4423	0.5523	0.4493	0.4733	0.1130
<b>ridge</b>	Ridge Classifier	0.8195	0.0000	0.7361	0.4397	0.5494	0.4456	0.4697	0.0440
<b>knn</b>	K Neighbors Classifier	0.6614	0.6350	0.5067	0.2243	0.3102	0.1295	0.1480	0.1100
<b>nb</b>	Naive Bayes	0.6472	0.6251	0.5182	0.2205	0.3059	0.1226	0.1425	0.0400
<b>svm</b>	SVM - Linear Kernel	0.6280	0.0000	0.4229	0.2191	0.1895	0.0783	0.1034	0.1280
<b>qda</b>	Quadratic Discriminant Analysis	0.1886	0.4878	0.9168	0.1464	0.2523	-0.0074	-0.0417	0.0560

*Figure 5: Comparaison des modèles de machine learning*

Comme on peut le constater le modèle ayant plus de précisions à l'issue de la phase de formations des modèles est **xgboost**, avec une précision de **0.9490 %**. Nous l'utiliserons dans la suite de notre travail.

## 6. Création et ajustement du modèle

Après la phase de formation du modèle avec la fonction « *compare\_model* », on procède maintenant à la création et l'ajustement du modèle de classification binaire ayant obtenu la plus grande précision lors de phases précédentes.

Pour ce faire, dans un premier temps grace à la fonction « *create\_model* » de PyCaret on crée un modèle **xgboost** en passant en paramètre le mot clé « *xgboost* ».

Après cela on procède à l'ajustement du modèle (**Tuning**) pour affiner ses hyperparamètres. L'objectif est de trouver les valeurs optimales des hyperparamètres pour améliorer les performances de classification du modèle.

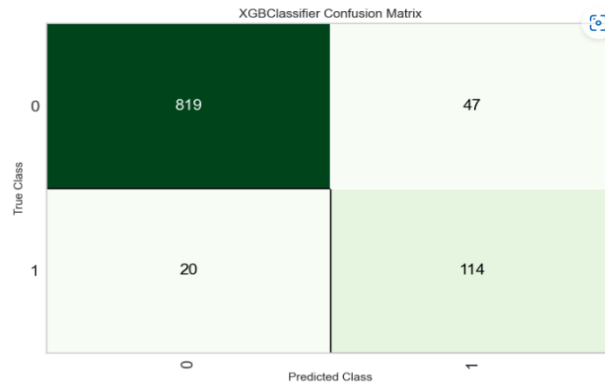
L'affinement du modèle est réalisé grace à la fonction « *tune\_model()* » de *PyCaret*. Elle utilise une méthode de recherche de grille pour explorer différentes combinaisons d'hyperparamètres et sélectionner la meilleure combinaison en termes de métrique de performance spécifiée par défaut, la précision est choisie.

```
# model creation  
xgboost_model = create_model('xgboost')
```

```
# model tuning  
tuned_xgboost_model = tune_model(xgboost_model)
```

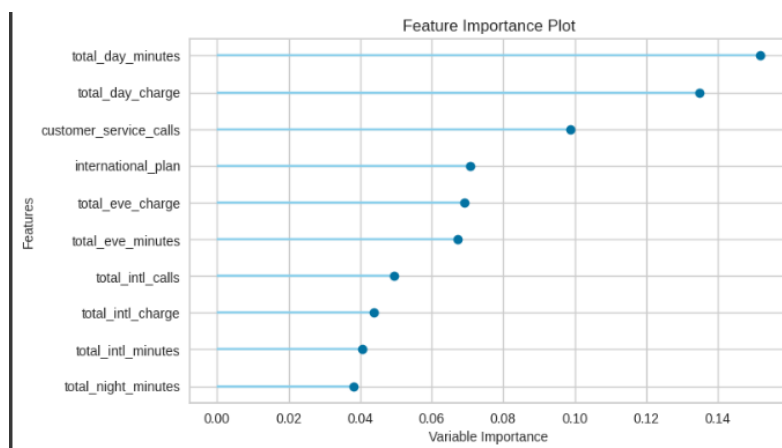
*Figure 6: Création et Tuning du meilleur modèle*

Ci-dessous on retrouve la matrice de confusion notre modèle. En rappel, la matrice de confusion est un tableau qui permet de visualiser les performances d'un modèle de classification en comparant les prédictions de ce modèle avec les vraies valeurs de la variable cible.



*Figure 7: Matrice de confusion*

Le graphique d'importance des variables montre les variables les plus importantes en ordre décroissant de leur impact sur la prédiction du modèle. Cela peut aider les à comprendre quelles variables ont le plus d'influence sur les prédictions du modèle et à identifier les caractéristiques importantes de l'ensemble de données.



*Figure 8: Graphe d'importance des variables*

## 7. Evaluation du modèle

Avant d'aller plus loin, évaluons les performances du modèle. Nous allons exploiter la fonction ***evaluate\_model()*** de PyCaret pour évaluer et mettre en évidence les aspects importants du modèle retenu. La fonction ***evaluate\_model*** affiche les mesures de performance dans la console, comme le montre l'image ci-dessous :

Pipeline Plot	Hyperparameters	AUC	Confusion Matrix	Threshold	Precision Recall	Prediction Error	Class Report
Feature Selection	Learning Curve	Manifold Learning	Calibration Curve	Validation Curve	Dimensions	Feature Importance	Feature Importance...
Decision Boundary	Lift Chart	Gain Chart	Decision Tree	KS Statistic Plot			

*Figure 9: Grille d'évaluation du modèle offerte par la fonction ***evaluate\_model****

## 8. Finalisation du modèle

La finalisation du modèle implique souvent de réentraîner le modèle sur l'ensemble complet des données, y compris les données de test, afin d'améliorer les performances et de maximiser la généralisation du modèle. La finalisation peut également impliquer l'application de transformations finales sur les données, telles que la normalisation ou la transformation de la cible, pour s'assurer que le modèle est compatible avec les données de prédiction.

La fonction ***finalize\_model*** de PyCaret permet de simplifier le processus de finalisation du modèle en effectuant automatiquement les étapes finales et en produisant un modèle finalisé qui peut être utilisé pour faire des prédictions sur de nouvelles données avec une grande facilité.

```
final_tuned_xgboost_model = finalize_model(xgboost_model)
```

*Figure 10: Finalisation du modèle*

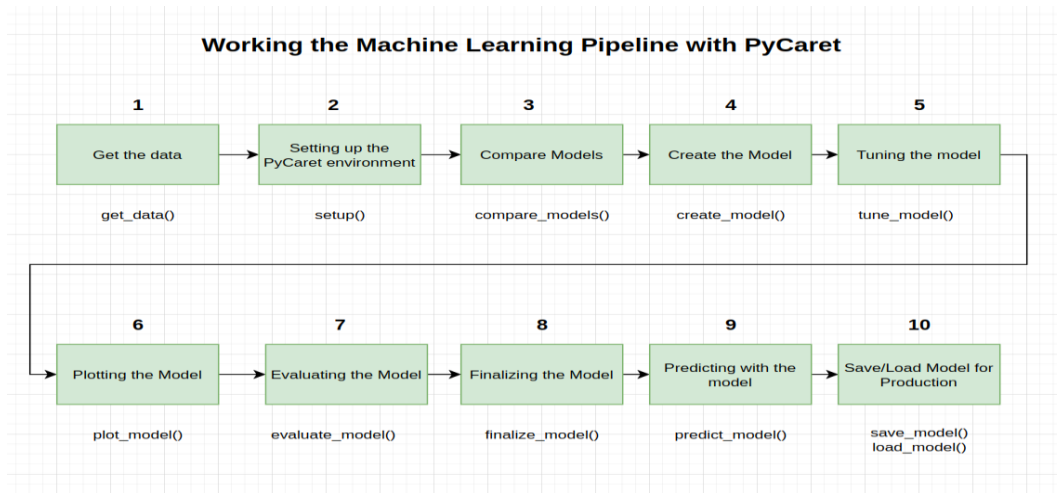
## 9. Sauvegarde du modèle

La fonction *save\_model* de PyCaret permet de sauvegarder le modèle finalisé sous forme de fichier binaire qui peut être chargé ultérieurement pour effectuer des prédictions sur de nouvelles données.

En sauvegardant le modèle, nous pouvons le partager avec d'autres personnes ou l'utiliser ultérieurement pour faire des prédictions sur de nouvelles données, sans avoir à réentraîner le modèle à chaque fois. Cela permet de gagner du temps et d'assurer une certaine cohérence dans les résultats de prédiction.

```
# save model to disk
save_model(final_best, 'ChurnModel')
```

*Figure 11 : Sauvegarde du modèle*



*Figure 12: Pipeline de construction de notre modèle de machine Learning*

### III. Déploiement du modèle

#### 1. Nécessité d'un environnement virtuel

Un environnement virtuel est une installation de Python qui est isolée du système global installé sur votre ordinateur. Il permet de créer un environnement de développement séparé pour chaque projet Python, avec ses propres dépendances et packages installés sans interférer avec les autres projets Python installés sur le même système.

Nous avons créé un environnement virtuel pour ce projet Python car il utilise des dépendances spécifiques qui ne sont pas compatibles avec ceux sur le système global. En effet le projet nécessite principalement les packages aux versions suivantes :

- *Scikit-learn==0.23.2*
- *pycaret==2.3.10*
- *streamlit==1.18.1*
- *xgboost==1.7.4*
- *fastapi*
- *python==3.8.10*

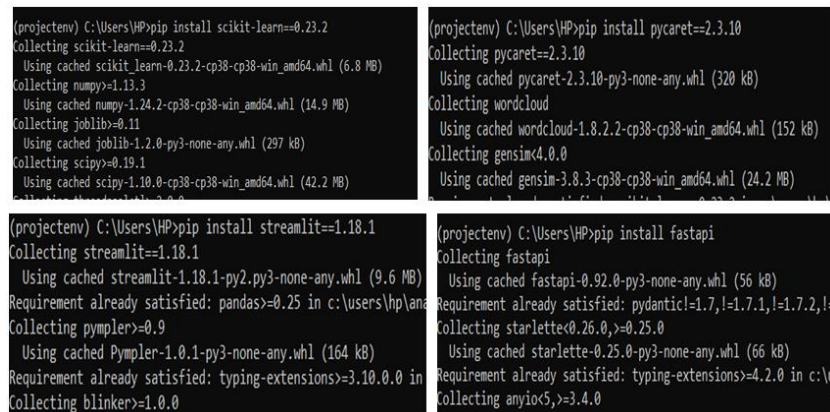
#### 2. Configuration de l'environnement

Avec le prompt de Anaconda nous pouvons créer et installer les différents packages requis pour notre projet :



*Figure 13: Terminal de Anaconda*

- « *conda create -name projectenv python=3.8* » : création d'un environnement virtuel du nom de *projectenv* avec *python 3.8* pour interpreteur
- « *conda activate projectenv* » : activation de l'environnement virtuel.



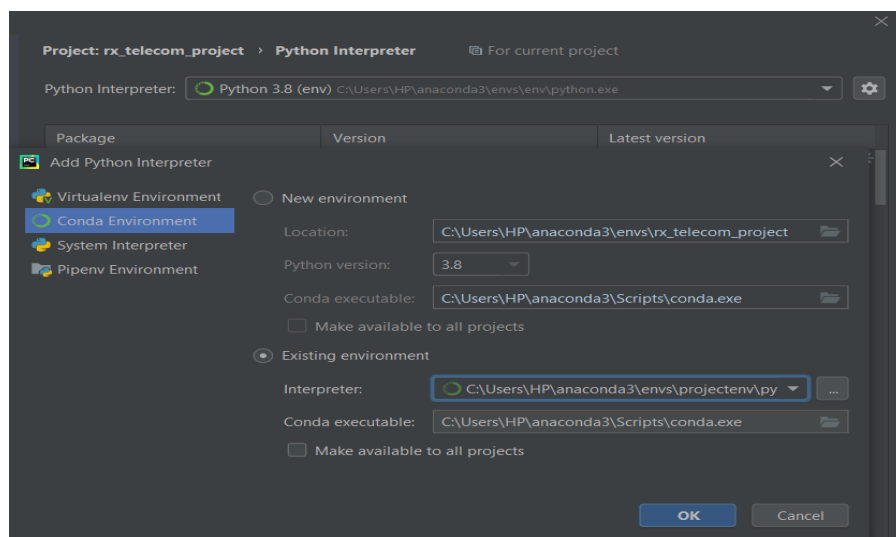
*Figure 14: Installation des packages requis*



## Configuration de l'environnement virtuel dans PyCharm

Cette configuration vise à informer PyCharm de la présence d'un environnement virtuel qu'il faut utiliser pour interpréter notre code Python. Pour cela suivez les instructions suivantes :

1. Ouvrir PyCharm et puis votre projet ;
2. Dans le menu "**File**" et sélectionnez "**Settings** » ;
3. Au niveau de la fenêtre "Settings", sélectionnez "Project" dans la colonne de gauche, puis sélectionnez "**Project Interpreter**" ;
4. Cliquez sur l'icône paramètre en haut à droite de la fenêtre, puis sélectionnez "**Add**".
5. Sélectionnez "**Conda Environment**" dans la colonne de gauche, puis "**Existing environment**". Entrez le chemin d'accès complet à votre environnement virtuel conda (dans notre cas ce chemin est le suivant : "C:\Users\HP\anaconda3\envs\env\python.exe") dans le champ "Interpreter".
6. Cliquez sur "OK" pour ajouter l'interpréteur de l'environnement virtuel à votre projet.



*Figure 15 : Configuration de l'environnement virtuel dans PyCharm*

### 3. Déploiement avec FastAPI



FastAPI est un Framework web haute performance, open source, permettant de créer des APIs avec Python à partir de la version 3.6.

Ses principales caractéristiques sont les suivantes :

- ✚ Rapide : très haute performance, à égalité avec NodeJS et Go (grâce à Starlette et Pydantic).
- ✚ Rapide à coder : augmente la vitesse de développement des fonctionnalités d'environ 200 % à 300 %.
- ✚ Facile : conçu pour être facile à utiliser et à apprendre. Moins de temps à lire des documents.


Il utilise Uvicorn, un serveur ASGI (Asynchronous Server Gateway Interface) rapide et léger pour Python 3.6+ basé sur le moteur de serveur de synchronisation asynchrone "asyncio".

Il est utilisé pour déployer des applications web écrites en utilisant les Framework web Python tels que FastAPI, Django, Flask, etc. Uvicorn peut être utilisé pour fournir une meilleure performance et une latence plus faible pour les applications web en comparant avec les serveurs web traditionnels tels que WSGI (Web Server Gateway Interface).

Dans le répertoire « *rx\_telecom-project* » fournit en pièce-jointe vous trouvez deux fichiers utilisés pour le déploiement avec **FastAPI** :


- ✚ « *api.py* » : c'est un programme Python qui utilise les bibliothèques *pycaret*, *fastapi*, *pandas* et *uvicorn* pour créer une l'API de prédiction de *churn*.

Dans un premier temps le programme importe les bibliothèques nécessaires. Ensuite le modèle de machine Learning entraîné « **ChurnModel.pkl** » est chargé à l'aide de la fonction **load\_model()** de **pycaret.classification** pour les prédictions futures. Après cela, une fonction « **predict** » permet le renvoi d'un entier indiquant si le client est susceptible de résilier son abonnement « predict=1 » ou non « predict=0 ».

 « **ChurnModel.pkl** » : il s'agit du modèle de classification que nous avons sauvegardé, il est utilisé dans le fichier « api.py » pour la prédiction.

Pour exécuter ce fichier dans le terminal de PyCharm il faut :

- Saisir la commande « **uvicorn api:app --reload** ».
- Aller à l'adresse « <http://127.0.0.1:8000/docs> » dans votre navigateur.

**FastAPI** 0.1.0   
/openapi.json

default

POST /predict Predict

Vous verrez alors apparaître la méthode **predict** de notre api, cliquez sur le bouton dérouler, puis sur « **Try it out** ». Vous serez par la suite invité à entrer les différentes caractéristiques de votre client dans des champs. Après les avoir bien remplis cliquer sur le bouton « **Execute** ». C'est fait vous aurez en retour une prédiction vous indiquant si votre client a de potentiels chances de quitter votre réseau.

default

POST /predict Predict

Parameters

Name	Description
state <small>required</small> (query)	KS
account_length <small>required</small> (query)	128
area_code <small>required</small> (query)	415
international_plan <small>required</small> (query)	no
voice_mail_plan <small>required</small> (query)	yes
number_vmail_messages <small>required</small> (query)	25
total_day_minutes <small>required</small> (query)	265.1

Cancel

200

Response body

```
{
  "prediction": 0
}
```

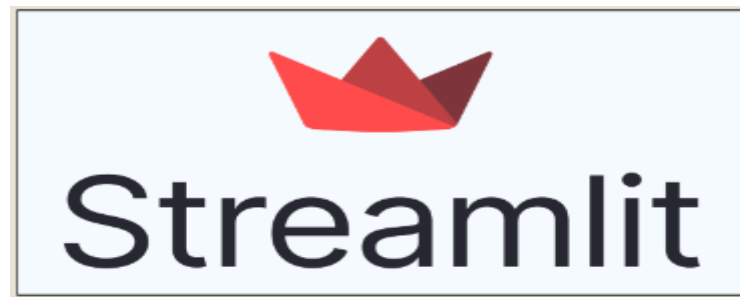
Download

Response headers

```
content-length: 16
content-type: application/json
date: Sat, 18 Feb 2023 22:28:15 GMT
server: uvicorn
```

Figure 16: Test du déploiement avec FastAPI

## 4. Déploiement avec Streamlit



Streamlit est un Framework open-source Python spécialement conçu pour les ingénieurs en machine Learning et les Data Scientist. Ce Framework permet de créer des applications web qui pourront intégrer aisément des modèles de machine Learning et des outils de visualisation de données.

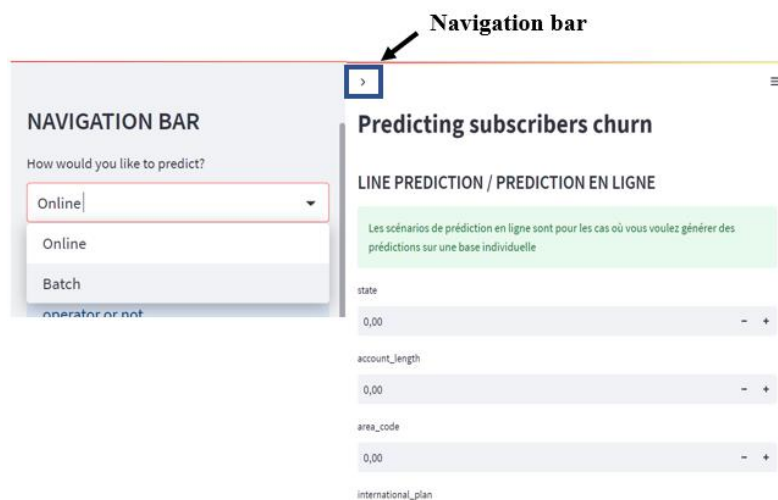
Contrairement aux autres Framework python pour créer des applications autour de la données, Streamlit permet de créer de belles applications web sans écrire du code HTML. Ce Framework permet aussi d'avoir des applications performantes grâce à la mise en cache via une annotation.

Un autre point positif est que Streamlit peut se connecter à plusieurs plateformes logicielles. Elle est compatible avec la majorité des Framework de datavisualisation (Matplotlib, Plotly, Seaborn,) et de Machine Learning (Pandas, Pytorch...). Cela fait

gagner en productivité et réduit, également, de façon significative les coûts de développement logiciel.

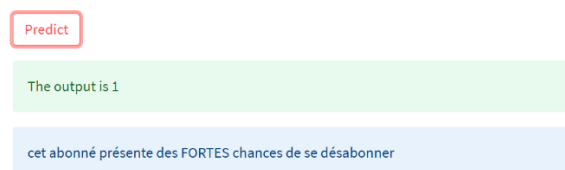
Le programme Python « *strlt.py* » contenu dans le dossier notre projet « *rx\_telecom-project* » contient un ensemble de fonctions permettant le déploiement de notre modèle avec Streamlit et offre les fonctionnalités suivantes :

### ❖ *La prédiction online*



*Figure 17: Online prediction with streamlit*

Elle d’ permet effectuer une prédiction portant sur une seule observation. Comme on peut le voir sur la figure ci-dessous, pour effectuer une prédiction online il faut choisir l’option « **online** » au niveau de la barre de navigation se trouvant à la page d’accueil de notre projet. Après cela vous serez invité à entrer les différentes caractéristiques de votre client pour la prédiction. Enfin appuyez sur le bouton prédit comme le montre la figure ci-dessous pour obtenir le resultat de la prédiction.



## ❖ La prédiction en Batch

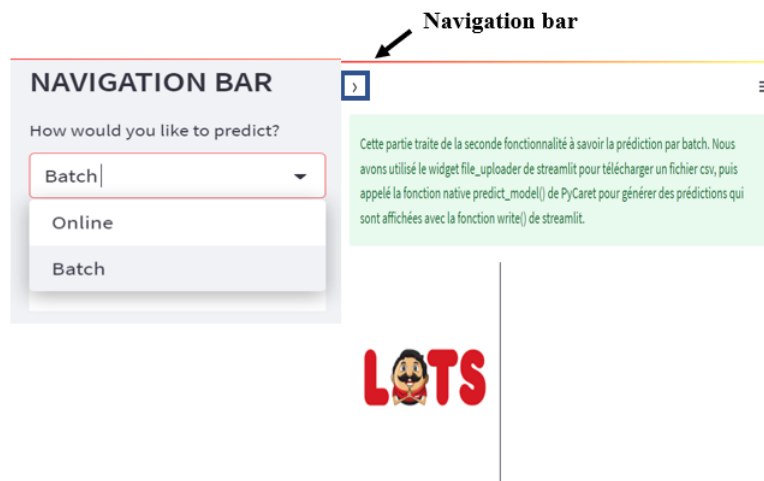


Figure 18: Batch prediction with streamlit

Là, il s'agit d'effectuer une prédiction sur un ensemble de clients contenu dans un fichier csv. Après avoir choisi ce type de prédiction au niveau de la barre de navigation il faut par la suite charger le fichier csv grâce à l'option « **deployment of model** » se trouvant sous le titre « Datascience » dans la même barre de navigation. Enfin il faut choisir le modèle pré-entraîner qui sera utilisé pour la prédiction puis appuyer sur le bouton « **predict** » comme nous le montre l'image ci-dessous :

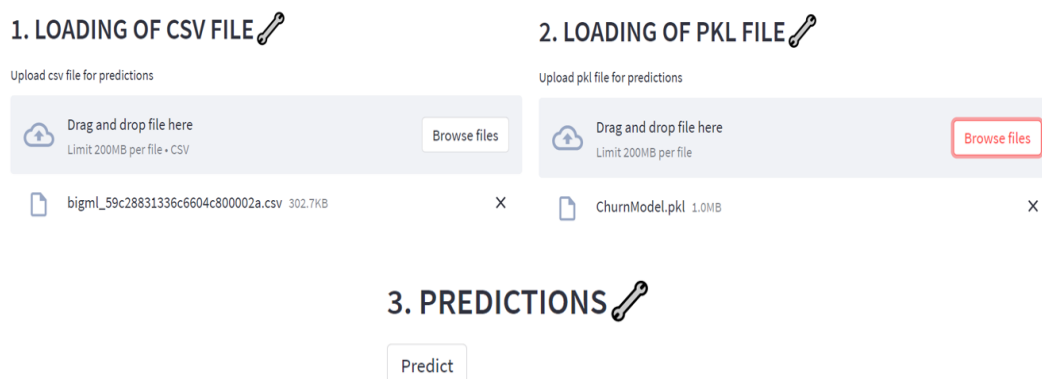


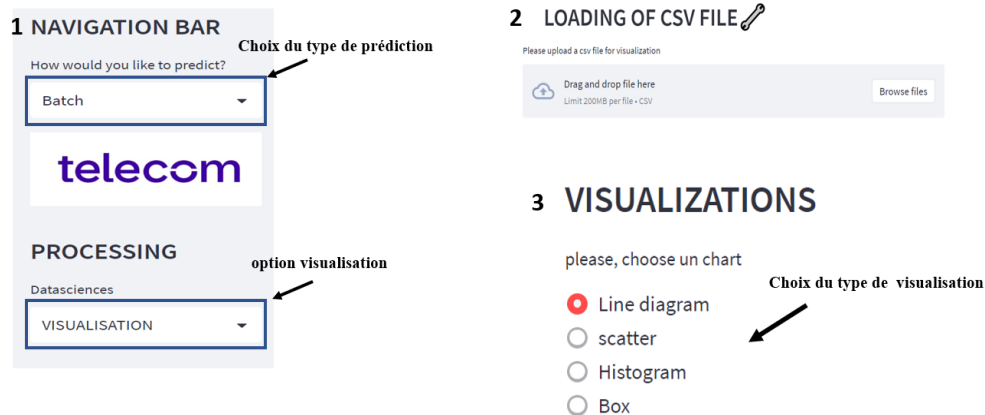
Figure 19: Roadmap of Batch prediction with streamlit

## ❖ *La visualisation de données*

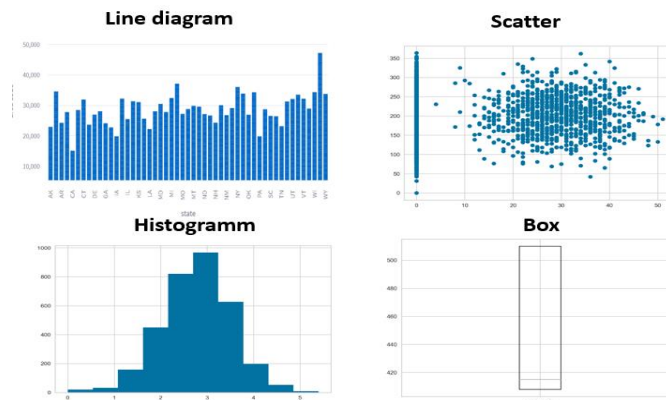
Cette fonctionnalité permet d'observer la représentation graphique des attributs de votre ensemble de données. Parmi types de graphe mis à votre disposition sont :

- **Line diagram**
- **Scatter**
- **Histogram**
- **Box**

Pour effectuer une visualisation sur vos données vous devez au préalable choisir « **Batch** » comme type de prédiction. Puis sélectionner « **visualisation** » au niveau du menu déroulant se trouvant sous le titre « **Datascience** » dans la barre de navigation. Enfin charger votre fichier csv et choisissez le type de graphe que vous désirez.



*Figure 20: Roadmap of datavisulization with streamlit*



*Figure 21: Types de graphes offerts dans l'API*

## ❖ L'obtention d'informations sur un jeu de données

Elle permet de retourner un ensemble d'informations utiles sur un dataset tel que :

- La taille du dataset (nombre de lignes et colonnes) ;
- Des informations sur les colonnes présentes ainsi que leurs types
- Des informations statistiques telles que la moyenne, l'écart-type etc... ;
- Des informations sur les colonnes possédant des valeurs manquantes.

Pour obtenir des informations sur vos données vous devez au préalable choisir « **Batch** » comme type de prédiction. Puis sélectionner « **INFORMATIONS ON DATASET** » au niveau du menu déroulant se trouvant sous le titre « **Datascience** » dans la barre de navigation. Enfin charger votre fichier csv et obtenez le type d'information que vous désirez.

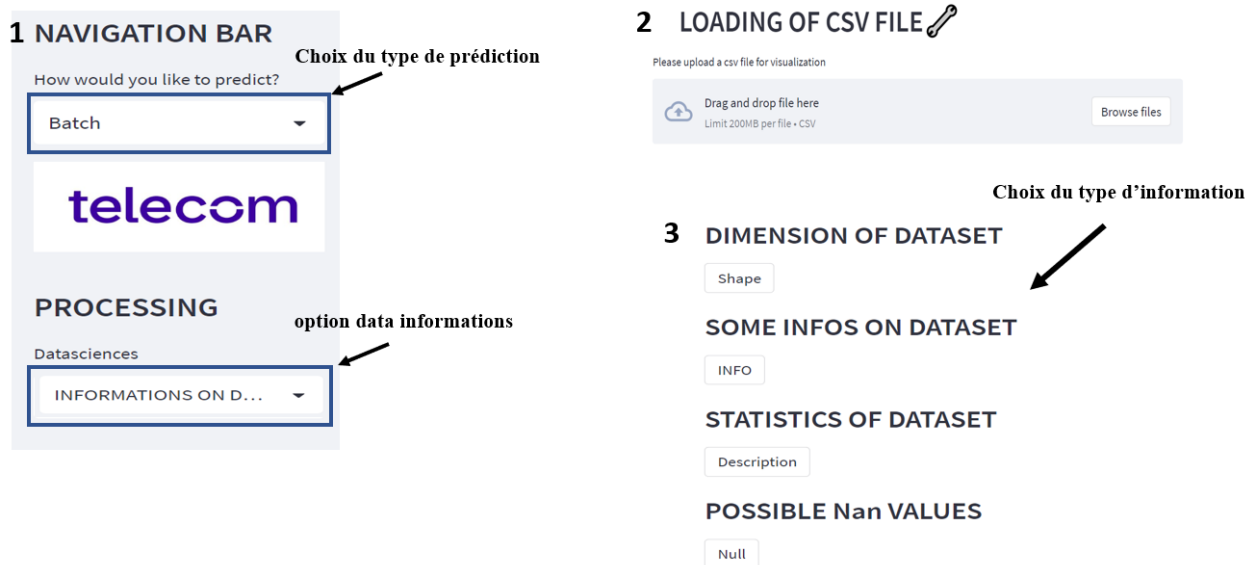


Figure 22: Roadmap for the acquisition of information on a dataset

Pour exécuter le fichier `strlt.py` il faut saisir la commande suivante dans le prompt du projet dans PyCharm : « *streamlit run strlt.py* ».



## 5. Déploiement avec Flask



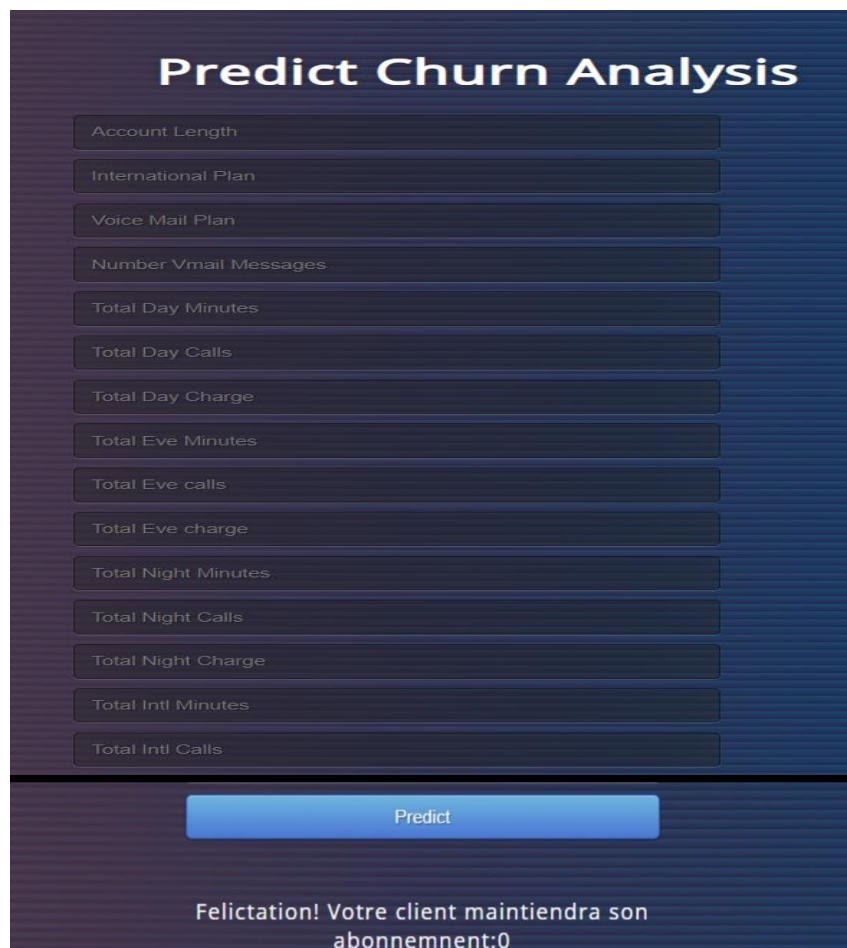
Flask est un micro Framework Web Python qui facilite la création d'une application Web complète. Un micro Framework d'un ensemble de modules qui permettent de développer plus rapidement en fournissant des fonctionnalités courantes. Lorsque vous concevez une application web, vous avez toujours besoin de gérer les requêtes http d'un serveur web, d'afficher des pages web dynamiques, de gérer les cookies. Un Framework web vous fournit ces fonctionnalités pour commencer un projet sur des bases solides.

Flask est populaire car il est à la fois très puissant et très léger. Par exemple, il permet de créer une application web minimale en 7 lignes.

Le déploiement et le fonctionnement de notre API avec Flask sont similaires à celui faits avec FastAPI. Pour l'utiliser vous devez aussi procéder par création d'un environnement virtuel avec Anaconda puis à l'installation des packages nécessaires :

- *Scikit-learn==1.0.2*
- *python==3.8.10*
- *Flask==2.2.3*
- *numpy==1.24.2*
- *pandas==1.5.3*

Pour exécuter notre API déployé avec Flask veuillez exécuter le fichier : « *app.py* » dans PyCharm. Cliquez par la suite sur le lien « <http://127.0.0.1:5000> » qui s’affiche sur votre terminal. Enfin remplissez les différents champs et cliquez sur le bouton « *Predict* »



The image shows a web application titled "Predict Churn Analysis". It features a dark blue background with a grid of 15 input fields for user data. The fields are labeled as follows:

- Account Length
- International Plan
- Voice Mail Plan
- Number Vmail Messages
- Total Day Minutes
- Total Day Calls
- Total Day Charge
- Total Eve Minutes
- Total Eve calls
- Total Eve charge
- Total Night Minutes
- Total Night Calls
- Total Night Charge
- Total Intl Minutes
- Total Intl Calls

Below the input fields is a large blue button labeled "Predict". At the bottom of the interface, a message reads: "Felicitation! Votre client maintiendra son abonnement:0".

*Figure 23: Déploiement avec Flask*

# TABLE DES MATIERES

SOMMAIRE .....	2
TABLE DES ILLUSTRATIONS .....	3
INTRODUCTION .....	4
I. Environnement de travail utilisé .....	5
1. PyCharm .....	5
2. Annaconda .....	5
II. Pipeline de construction du modèle de machine Learning .....	6
1. Présentation du jeu de données .....	6
2. Analyse exploratoire .....	7
3. Présentation de PyCaret .....	8
4. Préparation des données .....	9
5. Formation et sélection de modèles .....	10
6. Création et ajustement du modèle .....	11
7. Evaluation du modèle .....	13
8. Finalisation du modèle .....	13
9. Sauvegarde du modèle .....	14
III. Déploiement du modèle .....	15
1. Nécessité d'un environnement virtuel .....	15
2. Configuration de l'environnement .....	15
• Création et activation de l'environnement .....	16
• Installation des packages requis dans l'environnement virtuel .....	16
• Configuration de l'environnement virtuel dans PyCharm .....	17
3. Déploiement avec FastAPI .....	18
4. Déploiement avec Streamlit .....	20
❖ La prédiction online .....	21
❖ La prédiction en Batch .....	22
❖ La visualisation de données .....	23
❖ L'obtention d'informations sur un jeu de données .....	24
5. Déploiement avec Flask .....	25
TABLE DES MATIERES .....	27



FIN