

PROJET CHEF D'OEUVRE

CRÉATION D'UNE APPLICATION D'IA POUR LA PRÉDICTION DES ESPÈCES D'ARBRES

Awa MBENGUE SOW

Titre professionnel "Développeur en intelligence artificielle" de niveau 6 enregistré au RNCP sous le n°34757

DÉCEMBRE 2020

Table des matières

INTRODUCTION	3
1 Analyse du besoin client	4
1.1 Présentation du projet	4
1.2 Enjeux et objectifs du projet	5
1.3 Cahier des charges client	6
1.4 Etat de l'art	6
2 Gestion des données analytiques	8
2.1 Analyse exploratoire et description des données avec Pandas	9
2.2 Stockage dans mysql Workbench	12
2.3 Conception de la base de données analytique	13
2.4 Gestion des droits d'accès	14
3 Développement du programme d'IA	15
3.1 Préparation d'un environnement de développement	15
3.2 Choix du modèle d'algo	15
3.3 Workflow de ML	16
3.4 Analyse des résultats	18
4 Développement de l'application intégrant l'IA	19
4.1 Maquettage du front end de l'application avec UX DESIGN	19
4.2 Schéma fonctionnel et technos back-end de l'application	20
4.3 Gestion de la Base de données d'application My sqlAlchemy	21
4.4 Tests	22
4.5 Systèmes de suivi de l'application	23
4.6 Déploiement de l'Application et outils de mise en production	24
5 Gestion de projet	24
5.1 Organisation en mode Agile	25
5.2 Réalisation du planning projet	25
5.3 Comptes rendus et communications avec le client	26
5.4 Rétrospectives	29
CONCLUSION	30
ANNEXES	31

INTRODUCTION

L'intelligence artificielle (IA) est présente aujourd'hui dans le domaine de la santé, des économies d'énergie.

Depuis quelques années, les phénomènes de réchauffement climatique, d'incendies de forêts enregistrés en Sibérie, en Australie fin 2019 début 2020 et cet été au Portugal, en Amazonie, en Californie ont poussé les Data Scientists et les chercheurs en IA à s'investir davantage sur la proposition de solutions pour prévenir ces risques.

Dans la lutte contre la déforestation, l'Etat Français a créé la structure appelé **Inventaire forestier national (IFN)**.

Cette structure assure les missions suivantes :

- Inventorier les espèces d'arbres
- Prendre en charge le suivi et la gestion des forêts.

Sur le terrain, les agents de l'IFN mènent des enquêtes et travaux d'inventaires permanents sur les territoires forestiers français.

Depuis 2017, leurs **enquêtes sont à caractère obligatoire et reconnues d'intérêt général (article R. L.151.1 et I.151.1 du code forestier)**.

Afin d'optimiser leurs travaux sur le terrain, l'IFN souhaite intégrer les technologies d'Intelligence artificielle dans la réalisation de sa mission.

Dans ce cadre, nous sommes intervenus dans la réalisation d'une application métier qui va permettre de prédire le type de couverture forestière qui signifie les espèces d'arbres majoritairement présents sur une zone donnée.

Ce document a pour objectif de présenter les différentes étapes de la réalisation de cette application en partant de l'analyse du besoin, la mise en œuvre puis la présentation la méthode de gestion de projet utilisée.

1 Analyse du besoin client

1.1 Présentation du projet

En collaboration avec l'**ONF (Office National de Forêts)** en charge de l'exploitation du bois et du caoutchouc naturel, l'IFN prévoit une extension de ses zones d'intervention. Face à l'ampleur de la mission et la difficulté d'accéder à des zones, les responsables de ce service ont décidé de prendre des mesures.

Pour anticiper cette situation, l'IFN souhaite disposer d'une application qui lui permettra de savoir, sur une zone inconnue, quelles sont les espèces de bois présentes.

Cette mission est d'autant plus importante que les scientifiques s'accordent sur le fait que certaines essences d'arbres sont très inflammables et peuvent aggraver les feux de forêts. Les incendies dévastateurs comme ceux enregistrés cette année en Californie nous l'ont démontré. Ainsi la détection de la forte présence de certains arbres dans une forêt permettra de mettre en place des mesures de surveillance renforcée surtout en période de forte chaleur.

Notre intervention en tant que développeur Data IA a pour objectif d'exploiter les données statistiques provenant d'anciens travaux d'inventaires sur la base desquelles nous allons développer un modèle d'apprentissage supervisé qui sera intégré dans une application métier

1.2 Enjeux et objectifs du projet

L'enjeu est de créer une application intégrant un modèle de machine learning pouvant :

- **Détecter la présence de certaines classes d'arbres** dont celles vulnérables aux incendies afin d'avoir la possibilité de prendre des mesures de précaution adaptées comme la surveillance de certaines zones notamment en période de forte chaleur.
- **Assurer la continuité des missions d'inventaire** pendant les périodes d'intempéries où certains endroits deviennent difficiles d'accès.
- **Aider à mieux cibler les campagnes de reboisement** grâce à sa capacité à prédire les espèces d'arbres les plus adaptées à une zone donnée, en fonction des caractéristiques de l'écosystème connus.

L'application métier doit permettre aux équipes chargées de faire des inventaires forestiers de :

- **Disposer d'un outil de prédiction du type de couverture forestière rapide, efficace et de qualité.**

Pour cela, l'application doit intégrer un modèle de machine learning de classification des arbres présentes dans une zone donnée parmi les 7 types d'arbres sont le sapin (Spruce), le pin tordu (Lodgepole Pine), le pin Ponderosa (Ponderosa Pine), Cottonwood, le Tremble (Aspen), le Douglas Taxifolié, Krummholz.

Disposant d'un système d'information géographique qui leur permet de calculer les distances des zones à étudier avec les routes, les points d'eau..., nos clients possèdent un outil cartographique qui recense les types de sol dans chaque région forestière... Ils souhaitent que ces informations puissent être intégrées dans le formulaire de l'application.

L'ensemble des fonctionnalités attendues sont rédigées dans le cahier des charges ci-dessous.

1.3 Cahier des charges

L'application doit répondre à ces critères fonctionnels ci-dessous :

- Simplicité de la page de garde avec un formulaire contenant des checkbox et des blocs
- Ce formulaire doit pouvoir contenir que des données numériques
- Affichage claire et rapide des résultats de prédiction
- Renseignement par l'utilisateur et en toute sécurité de ses identifiants et mot de passe
- Respect des droits prévus par le RGPD (Régime de protection des données personnelles) en ce qui concerne les données d'identification de l'utilisateur
- La sécurité des accès au logiciel et des données

1.4 Etat de l'art

Deux points seront abordés ici. Le 1er concerne les modèles d'IA utilisés dans le management de l'environnement et le 2e est relatif au traitement des données déséquilibrées.

A partir des données géographiques, météorologiques, photographiques...il est possible grâce au machine learning d'identifier la faune ou la flore comme la classification de fleur.

Avec le deep learning et la 'computer vision' il est devenu possible de détecter des zones de déforestation avec l'analyse des images satellites. Ces avancées sont au cœur des enjeux de suivi et de gestion durable de l'environnement. Les programmes comme AI for Earth lancée par Microsoft en

2017 offrent des solutions basées sur des modèles de deep learning pour faire de la reconnaissance des espèces grâce à l'imagerie satellite.

Nos recherches nous ont montré qu'il existe plusieurs travaux basés sur des données similaires aux nôtres tels que le logiciel développé par *Wifire Lab* (une start up américaine), permettant de modéliser les feux selon des conditions météorologiques et le type de végétation.

En France, il existe des projets qui s'inscrivent dans cette optique comme le programme '*Firecaster*' développé par les chercheurs du laboratoire Science pour l'environnement (SPE) de l'université de Corse. Leur outil d'aide à la décision entraîné sur des données de l'écosystème forestier, simule le comportement, les impacts et la vitesse de propagation des feux sur la végétation.

Le 2e point concerne le phénomène de données déséquilibrées. Réussir un modèle de classification multi classe nécessite d'avoir à disposition des données d'entraînement 'propres' et 'équilibrées' afin d'éviter tout risque d'"*overfitting*" (sur apprentissage) ou d' "*underfitting*" (sous apprentissage) notamment. En pratique, ce n'est pas toujours le cas car les jeux de données déséquilibrés sont courants dans les applications du monde réel telles que la *détection de fraude*, le *diagnostic des maladies rares* ou comme dans notre cas qui concerne la *classification des espèces d'arbres* (certaines espèces étant naturellement plus rares que d'autres). Pour entraîner un modèle de machine learning sur des données déséquilibrées l'état de l'art offre deux possibilités :

- chercher un algorithme adapté tel que le Boosting ou les SVM (Support Vector Machine)
- pratiquer le ré-échantillonnage des données c'est à dire, augmenter le nombre de données concernant les événements les plus rares ou sous échantillonner les classes sur représentées

Pour la deuxième possibilité relative au ré-échantillonnage, il existe un package python appelé **imbalanced-learn** qui propose les algorithmes suivants:

- NearMiss (basé sur les proches voisins et permet de sous-échantillonner la classe majoritaires),
- SMOTE (créant de individus proches des exemples minoritaires afin de sur représenter cette classe),
- SMOTEENN (donnant un résultat intermédiaire qui est une combinaison des deux stratégies : le sous-échantillonnage et le sur-échantillonnage).

D'autres méthodes telles que la pondération des coûts, le transfert learning (utilisé en deep learning, consiste à utiliser les informations apprises par un réseau de neurones pour résoudre des cas

similaires ou proches), ou les transformations aléatoires pour les images font partie des solutions existantes.

Ainsi, l'approche à privilégier va dépendre des données disponibles. Dans notre cas, nous avons testé les deux premières possibilités que sont l'utilisation de l'algorithme **Boosting** et le ré-échantillonnage des données. Nous verrons plus loin comment nous les avons implantés.

L'état de l'art montre que l'intégration de l'IA dans la gestion de l'environnement permet d'avoir des bénéfices en termes de rapidité, d'efficacité des actions à mener sur le terrain et d'anticipation sur les mesures à prendre. Pour arriver à de telles réalisations il faut également bien étudier les données pour pouvoir résoudre d'éventuels déséquilibres qui risquent de fausser le modèle d'apprentissage.

Les données constituent le fondement de notre projet. Sans elles, il est impossible d'avoir un modèle de machine learning.

Dans la phase suivante, l'organisation et la gestion des données seront abordées.

2 Gestion des données analytiques

Dans le cadre de la mise en place de l'application, nous avons pu avoir un jeu de données fourni par le client. Il contient les descriptions de plusieurs types de couvertures forestières ainsi que les caractéristiques topographiques, géographiques... du milieu dans lequel se trouvent 7 espèces d'arbres.

Les phases d'analyse des données, la conception, le système de gestion et le processus de stockage en bases de données seront détaillés dans cette partie.

2.1 Analyse exploratoire et description des données avec Pandas

Les données reçues sont sous format CSV (comma separated values), accompagné d'un document explicatif.

Pour le visualiser, l'explorer, comprendre son contenu et vérifier son adéquation avec le projet, nous avons utilisé les framework python: PANDAS , MATPLOTLIB et SEABORN (outils de réalisation de graphiques et visualisations).

Ce script ci-dessous nous a permis de lire les données dans le dossier 'data/' sur Jupiter notebook pour voir leur contenu et les différentes variables qui les composent.

```
import numpy as np, pandas as pd
df = pd.read_csv('data/covtype.csv')
df.info()
```

La commande .info() de Pandas nous donne les informations sur la structure, les noms des variables, le type de données, les valeurs manquantes, la taille des données en mémoire...

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 581012 entries, 0 to 581011
Data columns (total 55 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   Elevation                            581012 non-null  int64
1   Aspect                              581012 non-null  int64
2   Slope                               581012 non-null  int64
```

Cet extrait ci-dessus nous montre la composition des données en détail. Elles contiennent **581012** lignes et **55** colonnes. Chaque ligne correspond à des observations faites à partir de parcelles de

forêt de 30 m sur 30 m qui sont classées parmi sept types de couvertures. Chaque colonne contient des variables relatives aux types d'arbres, l'éclairage des zones, la distance par rapport aux points de repère à proximité (routes, points d'eau, etc.), le type de sol et la topographie locale.

Toutes les variables sont des entiers de type int64. Elles peuvent être réparties en fonction de leur nature comme suit:

- 10 variables quantitatives, 5 qualitatives, 40 booléennes.

Ci-dessous, le tableau descriptif des variables.

Nom de variables	Mesure	Description / type de variable
Elevation	Mètre	Mesure l'altitude / quantitative
Aspect	Degrés	Aspect en degrés azimuth / quantitative
Slope	Degrés	Pente en degrés/quantitative
Horizontal_Distance_To_Hydrology	Mètre	Horizontale Distance proximité point d'eau/quantitative
Vertical_Distance_To_Hydrology	Mètre	Verticale Distance proximité point d'eau / quantitative
Horizontal_Distance_To_Roadways	Mètre	Horizontale Distance proximité route / quantitative
Hillshade_9am	0-255 pixels	Indice de luminosité à 9h, solstice d'été/ qualitative
Hillshade_Noon	0-255 pixels	Indice d'ombrage à midi, solstice d'été / qualitative
Hillshade_3pm	0-255 pixels	Indice des ombrages à 15h, solstice d'été / qualitative
Horizontal_Distance_To_Fire_Points	Mètre	Distance par rapport aux points l'allumage de feu de forêt les plus proches/ quantitative
Wilderness_Area (4 colonnes binaires)	1-4	Zone géographique / quantitative
Soil_Type (40 colonnes binaires)	0-1	Type de sol (booléens)
Cover_Type (7 Classes)	Classe (1-7)	Type de couvert forestier

La fonction *describe()* nous a permis d'avoir les statistiques descriptives de nos données à savoir, les valeurs minimum, maximum, moyenne, écart type, médiane... et permet de détecter notamment les valeurs aberrantes qui sont des valeurs extrêmes, anormalement différentes des autres valeurs de la même variable.

Pour les déceler, nous avons observé les valeurs supérieures au 3^{ième} quartile et celles inférieures au 1^{ier} quartile.

Sur l'exemple en image ci-dessous nous nous sommes intéressés aux valeurs surlignées en jaune qui sont éloignées des 1^{er} quartile (58 et 9). En annexes (1) se trouve le script de détection des Outliers que nous avons construit.

```
df.describe()
```

	Elevation	Aspect	Slope	H
count	581012.000000	581012.000000	581012.000000	
mean	2959.365301	155.656807	14.103704	
std	279.984734	111.913721	7.488242	
min	1859.000000	0.000000	0.000000	
25%	2809.000000	58.000000	9.000000	
50%	2996.000000	127.000000	13.000000	
75%	3163.000000	260.000000	18.000000	
max	3858.000000	360.000000	66.000000	

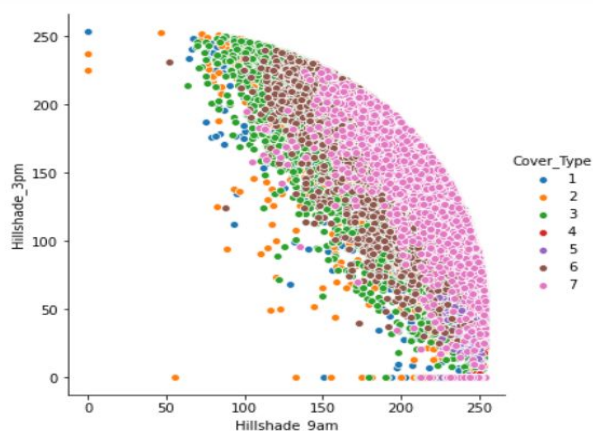
Le résultat nous montre que sur les 10 variables continues, 6 contiennent des valeurs extrêmes.

Il s'agit des variables suivantes:

- 'Slope', 'Vertical_distance_To_hydrology', 'horizontal_distance_To_hydrology', 'Hilshade_9am', 'Hilshade_Noon', 'horizontal_distance_To_Fire_Points'.

Une visualisation graphique avec l'outil SEABORN permet d'illustrer ce constat. Comme ce scatter plot ci-dessous où l'on aperçoit quelques points dispersés (exemple : en haut à gauche du graphe) qui s'éloignent du nuage de points..

```
# Pairplot pour la visualisation des outliers
for v,i,j in s_corr_list:
    sns.pairplot(data = df_train, hue='Cover_Type', size=5, x_vars=cols[i], y_vars=cols[j])
plt.show()
```



L'analyse des données met en évidence les problématiques qu'il a fallu prendre en charge :

- Une surreprésentation de certaines variables. Les variables ‘Soil_Type’ sont au nombre de 40 .

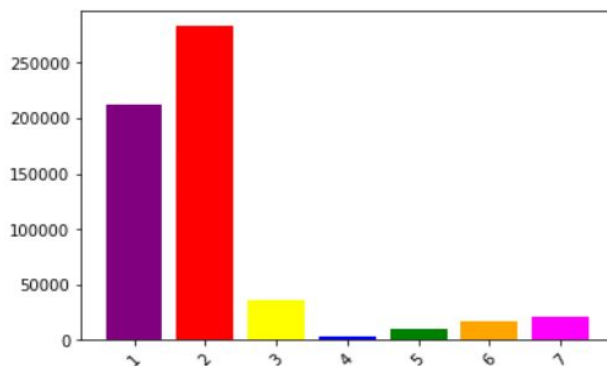
Face à cette problématique, nous avons procédé à un regroupement par similarité sera nécessairement effectué.

- Un déséquilibre des classes concernant les variables à prédire Cover_Types. Il existe une répartition inégale entre les espèces d’arbres à prédire. Certaines classes (classe 4) représentent moins de 1% des variétés. Cette visualisation ci-dessous permet d’illustrer ces cas.

```

In [6]: # Visualisation de la répartition des classes
plt.bar(range(1,8),df.groupby('Cover_Type')['Cover_Type'].count(),
        color=['purple','red', 'yellow','blue', 'green','orange','fuchsia'])
plt.xticks(rotation = 45)
plt.show()

```



- Une inadaptation des variables qualitatives comme (Wilderness_Area) qui ne répondent pas au besoin de simplicité du formulaire définis dans le cahier des charges du client. Nous envisageons donc de les exclure de notre modèle.

Cette première exploration a permis de comprendre les données et de connaître les mesures à prendre pour le nettoyage des données.

Elles ne contiennent pas de valeurs manquantes ni de doublons. L’identification des variables explicatives et du label (Cover_Type) faite dans cette étape a permis de valider la faisabilité du modèle d’entraînement et donc du projet.

Nous pouvons ainsi les stocker dans une base de données pour une facilité d’accès et d’optimisation d’espace.

2.2 Stockage dans MySQL Workbench

Le choix s’est porté sur les bases de données relationnelles avec MySQL Workbench.

Cette solution est simple, gratuite et adaptée pour le stockage de nos données structurées en local et en toute sécurité. Elle offre la possibilité de personnaliser la configuration du serveur et les paramètres de gestion des utilisateurs. Il fournit également des outils de sauvegarde, de modélisation de données (reverse engineer), de requêtage SQL.

Sa mise en place a suivi les étapes suivantes :

- Installation du logiciel
- Paramétrage des droits d'accès, de modification et de rectification
- Création de la base de données nommée 'PROJET_FINAL'
- Création de la table appelée COVTYPE pour contenir les données tout en respectant les types de chaque colonne. (Annexe 2 : Scripts de création de la base de données, de la table et d'insertion des données).

Les numérotations de lignes constituent la clé Primaire qui permet d'identifier chaque exemple. Dans la création de la table, chaque type de variable est respecté.(Elles sont toutes des entiers de type INT.)

Le script d'insertion des données est exécuté directement dans l'interface de mysql workbench.

2.3 Conception de la base de données analytique

La stratégie de nettoyage élaborée vise à obtenir le bon format de données qui servira à entraîner notre modèle de classification et qui permettra d'avoir un interface utilisateur simplifié avec un nombre raisonnable d'informations à renseigner par l'utilisateur de l'application.

Pour cela, nous avons procédé à une réduction du nombre de variables.

Dans cette phase de feature engineering nous nous sommes intéressés aux variables catégoriques qui sont au nombre de 44. Nous décidons de supprimer les 4 champs Wilderness_Area relatifs aux caractéristiques de ces zones géographiques.

```
#suppression des colonnes non utilisées dans le modèle
df=df.drop(['Wilderness_Area1', 'Wilderness_Area2', 'Wilderness_Area3', 'Wilderness_Area4'], axis=1)
```

- Suppression des variables 'Wilderness_Area'

Suite à la lecture du document explicatif qui accompagne les données, nous savons que les variables 'Wilderness_Area..' renvoient à des zones géographiques servant à localiser les zones.

Nous avons constaté que les autres variables topographiques, climatiques permettent également d'identifier des clusters et d'avoir des informations sur les zones concernées.

Nous avons pu conclure que la suppression de ces variables n'a pas d'impact significatif sur la prédiction du modèle.

- Réduction des variables 'Soil_Type'

Les 40 variables décrivant les types de sol sont mis en évidence dans le document explicatif des données.

Nous avons pu comprendre ces nombreux champs et leurs caractéristiques. Ils présentent des similarités et peuvent être regroupés.

Nous allons ainsi les réunir jusqu'à obtenir 5 catégories (au lieu des 40). Nous avons constitué les groupes suivants: sol de type argileux, rocheux, aquolis_crya, 'substratum', 'limber' .

```
df['soil_type_stony']    = df['Soil_Type1'] + df['Soil_Type2'] + ... + df['Soil_Type40']
df['soil_type_rocky']    = df['Soil_Type3'] + df['Soil_Type4'] + ... + df['Soil_Type35']
df['soil_type_substratum'] = df['Soil_Type21'] + df['Soil_Type23']
df['soil_type_limber']    = df['Soil_Type8'] + df['Soil_Type3']
df['soil_type_aquolis_crya'] = df['Soil_Type14'] + df['Soil_Type16'] + ... + df['Soil_Type20']
```

- Rééquilibrage des classes de 'Cover_Type'

En ce qui concerne les 7 classes d'arbres à prédire, l'exploration des données nous a montré une inégale répartition des classes. Le nombre d'observations des classes 4 et 7 est très faible par rapport aux autres notamment pour les classes 2 et 1 surreprésentées.

Effectuer une classification sur de telles données risque de biaiser le modèle. Pour cette raison, nous avons prévu dès cette étape du projet d'effectuer un traitement pour résoudre ce problème appelé aussi 'unsampled data' en utilisant la méthode SMOTE préalablement définie.

2.4 Gestion des droits d'accès

La mise à disposition des données implique un paramétrage des droits d'accès afin de préserver les données et d'éviter qu'elles ne soient modifiées par un membre de l'équipe de développement ou une tierce personne.

Nous avons de ce fait programmé notre base de données analytique de telle sorte que seuls les droits de lecture soient autorisés.

3 Développement du programme d'IA

Notre programme d'IA est basé sur du machine learning. L'objectif est de concevoir des modèles prédictifs faisant partie intégrante de notre application.

3.1 Préparation d'un environnement de développement

Après avoir stocké les données, nous avons créé un environnement de développement sous LINUX pour développer notre application et éviter toute dépendance ou conflits de versions entre les différents packages python.

Les interfaces Jupyter notebook, jupyter lab et VS CODE ont servi pour le codage de notre application, l'exploration, la visualisation, le nettoyage des données et l'entraînement de notre modèle d'IA.

3.2 Choix du modèle d'algorithme

L'entraînement d'un modèle de classification peut se faire avec différents outils de la Data science tels que Scikit-Learn, Tensorflow, PyTorch, SparkML, l'auto-ML avec Azure.

Notre choix pour entraîner l'algorithme s'est porté sur le package de référence pour faire du machine learning avec python : **Scikit-Learn**.

Ce package a l'avantage de rassembler de grandes quantités d'algorithmes, de modules de prétraitements des données tels que `sklearn.preprocessing`.

Il est également plus adapté à la structure du jeu de données, du type de variables et de notre problématique Business.

Après plusieurs tests de pertinence sur différents modèles, nous avons retenu deux modèles à présenter:

- Extratrees Classifier
- RandomForestClassifier.

3.3 Workflow de ML

Le processus d'entraînement a suivi les étapes suivantes.

Choix du label et sélection des variables explicatives

Notre label à prédire est la variable **'Cover Type'**. Il compte 7 classes. Chacune renseigne une espèce d'arbre présente dans une zone forestière. Les variables explicatives sont désormais au nombre de 15 après réduction du nombre de variables catégoriques (par groupage manuel, car la réduction de dimension avec des PCA (cette méthode d'analyse en composantes principales n'est pas du tout adaptée vu qu'il s'agit ici de variables booléennes).

Les 15 variables qui passeront dans le modèle sont les suivantes :

```
Index(['Elevation', 'Aspect', 'Slope', 'Horizontal_Distance_To_Hydrology',  
      'Vertical_Distance_To_Hydrology', 'Horizontal_Distance_To_Roadways',  
      'Hillshade_9am', 'Hillshade_Noon', 'Hillshade_3pm',  
      'Horizontal_Distance_To_Fire_Points', 'Cover_Type', 'soil_type_stony',  
      'soil_type_rocky', 'soil_type_substratum', 'soil_type_limber',  
      'soil_type_aquolis_crya'],  
      dtype='object')
```

Train/test/validation

Le découpage des données en plusieurs parties, constitue une phase importante avant l'entraînement d'un modèle.

Nous avons séparé nos données en plusieurs parties :

- Une partie réservée à l'entraînement : elle concerne le jeu de données réel que nous utilisons pour entraîner le modèle. Le modèle voit et apprend de ces données.
- Une autre partie réservée au test du modèle : le jeu de test constitue un échantillon de données utilisé pour fournir une évaluation non biaisée du modèle
- Une dernière partie pour la validation : elle est utilisée pour valider le modèle et pour mettre à jour les hyper-paramètres.

Choix du modèle

Pour rappel, nous disposons de données déséquilibrées car la répartition des classes d'arbres à prédire est inégale.

Pour cette raison nous avons orienté notre choix d'algorithmes sur des modèles de classification multi classes susceptibles de gérer ces situations. Nous avons donc testé plusieurs algorithmes avec et sans gestion de déséquilibre des classes pour les comparer.

Les algos ExtraTreesClassifier et SGDClassifier du package Sklearn ont été retenus avec des taux de réussite respectifs de 86% et 79%.

Pour obtenir ces résultats nous avons inclus l'outil GridSearchCV. Cette approche permet :

- D'avoir un modèle plus robuste
- De renseigner les valeurs des hyper-paramètres que l'on veut tester grâce à la création d'un dictionnaire comme ci-dessous:

```
# setting parameters
param = [{'n_estimators':[20,30,100], 'max_depth':[5,10,15], 'max_features':[0.1,0.2,0.3]}]
```

- De choisir un critère d'évaluation du modèle, la méthode de validation
- De tester plusieurs combinaisons à travers la Cross Validation. Elle consiste à découper aléatoirement le jeu d'entraînement en k sous-ensembles. Ci-dessous une illustration de la mesure de performance utilisé ('accuracy') et la validation croisée à 4 groupes avec (cv=4)

```
etc_model= GridSearchCV(ExtraTreesClassifier(),param_grid= param, scoring = 'accuracy',
cv=4)
```

Optimisation

La fonction **best_params()** nous permet de bien cibler les valeurs paramètres à modifier pour avoir un meilleur résultat. L'optimisation de code consiste à choisir les meilleurs hyper-paramètres comme la pénalité 'L1_ratio'.

Validation

Pour évaluer la performance du modèle, nous avons ajouté la matrice de confusion. Son principe consiste à mesurer le nombre de fois qu'une classe A a été rangée dans la classe B par exemple. Pour la comprendre, il faut savoir que chaque ligne de la matrice représente une classe réelle tandis que chaque colonne représente une classe prédite.


```

: ETC.best_score_
: 0.7934463478331999

: y_pred = ETC.predict(X_test)

: confusion_matrix(y_test, y_pred2)
: array([[15188,  5884,    4,    0,    0,    1,  102],
        [ 2959, 25061,  121,    0,    2,   36,   16],
        [    0,   292, 3208,    7,    0,   78,    0],
        [    0,    0,   75,  209,    0,   10,    0],
        [   39,  797,   25,    0,   77,    0,    0],
        [    1,  417,  402,    7,    1,  992,    0],
        [  557,   10,    0,    0,    0,    0, 1524]], dtype=int64)

```

3.4 Analyse des résultats

Algos ML	Modèles Sans gestion des déséquilibres de classes				Modèles Avec gestion des déséquilibres de classes			
	Accuracy	CV	Max_iter	Penalty	Accuracy	CV	Max_iter	Penalty
ExtraTreesClassifier / GridsearchCV	79 %	2	1000	L1_ratio	86%	2	1000	L1_ratio
SGDClassifier / GridsearchCV	59 %	5	1000	Elastic-net	79%	5	1000	Elastic-net

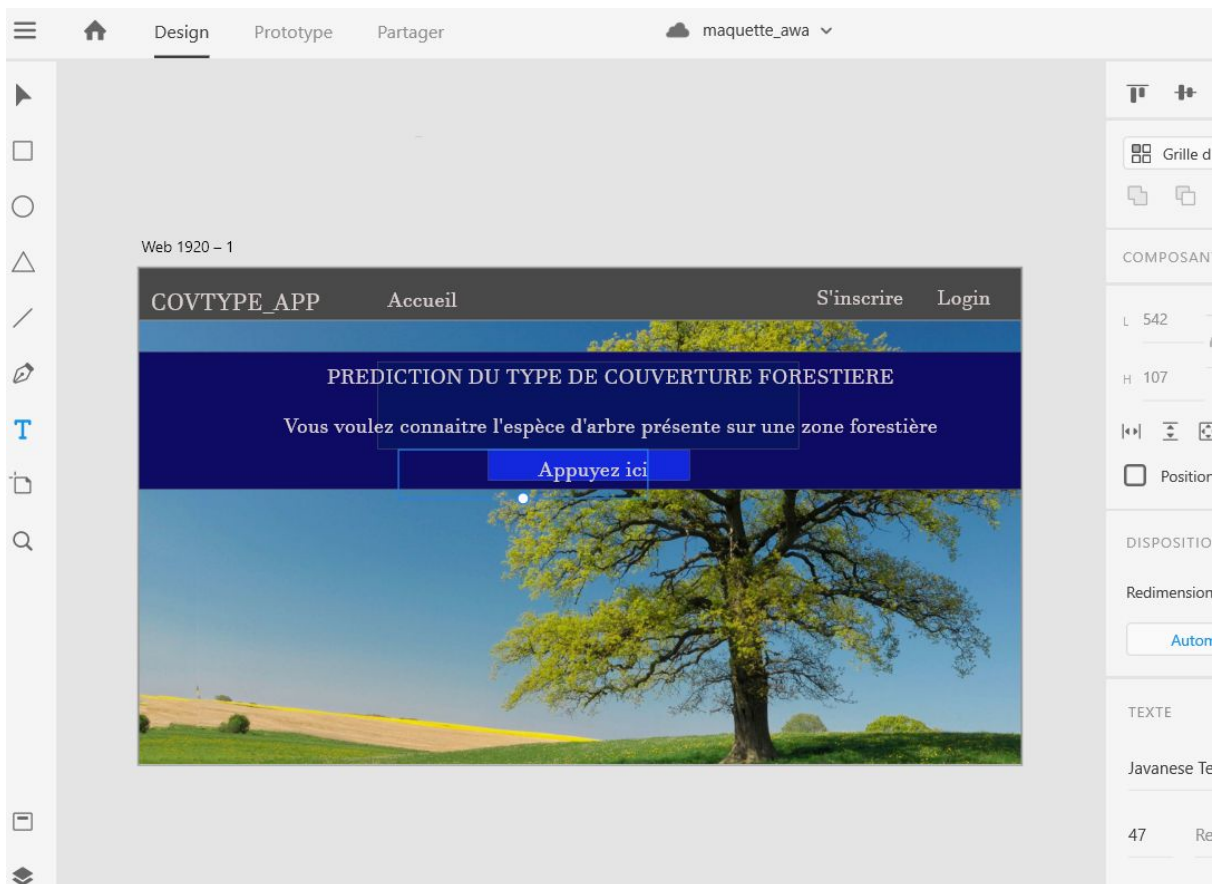
Le modèle retenu pour être intégré à l'application est le Extratrees Classifier les résultats sont meilleurs et la réalisation d'une matrice de confusion a permis de confirmer cette hypothèse. Pour chaque classe, le modèle réussit à bien prédire la classe de cover_type.

4 Développement de l'application intégrant l'IA

4.1 Maquettage du front end de l'application avec UX DESIGN

La maquette de l'interface de l'application est réalisée avec l'outil UX Design, un outil de conception d'interface d'application.

Nous avons retenu le modèle suivant comme étant la page d'accueil de l'application et l'avons proposé au client qui l'a validé par la suite.



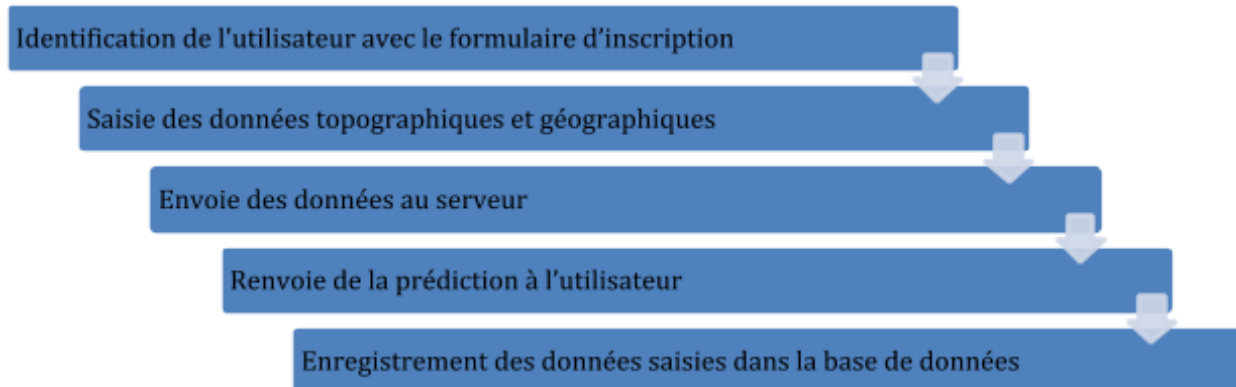
Ce prototype répond aux exigences du cahier de charges :

- Une barre de navigation claire et simple
- La visibilité des onglets principaux permettant l'identification de l'utilisateur
- Bouton d'accès au formulaire de prédiction.

Les autres pages de la maquette vous sont présentées en annexes 3.

4.1.1 Schéma fonctionnel et technos back end

Le fonctionnement de l'application doit respecter ce schéma ci-dessous allant de l'identification à l'enregistrement des données saisies dans la base de données d'application.



Les technologies et langages de programmation utilisés pour la réalisation du backend de l'application sont les suivants.

- Flask

Pour développer notre application nous avons utilisé le framework FLASK. Nous l'avons choisi pour les raisons suivantes:

- il est facile à utiliser et adapté au langage python
- il permet de concevoir une application web solide de manière professionnelle
- il s'adapte facilement à l'intégration des algorithmes de machine learning
- puissant, complet Flask comprend plusieurs modules de développement web
- il couvre toutes les fonctionnalités nécessaires à la réalisation du front-end d'une application
- la sécurité est garantie
- il assure une mise en place d'un suivi complet

- HTML

Le langage HTML a été nécessaire à la modélisation de la structure de notre application. Pour avoir un meilleur graphisme du formulaire et un beau design nous avons téléchargé BOOTSTRAP 5 . Ce dernier a permis de rendre notre application responsive c'est-à-dire avec un visuel bien optimisé sur ordinateur, tablette ou smartphone. Il est une référence dans le monde du développement car utilisé par une large communauté de développeurs web dans leurs projets. Ses points positifs sont:

- sa légèreté car pouvant s'intégrer facilement dans une application
- sa grande communauté d'utilisateurs fait sa force
- il est customisable et modulable en fonction du besoin et du type d'interface souhaités

- CSS

Le code CSS a été ajouté dans notre développement pour ajouter les décorations et couleurs nécessaires au visuel de l'interface de l'application et en rapport avec les attentes de notre client.

4.2 Gestion de la Base de données d'application MysqlAlchemy

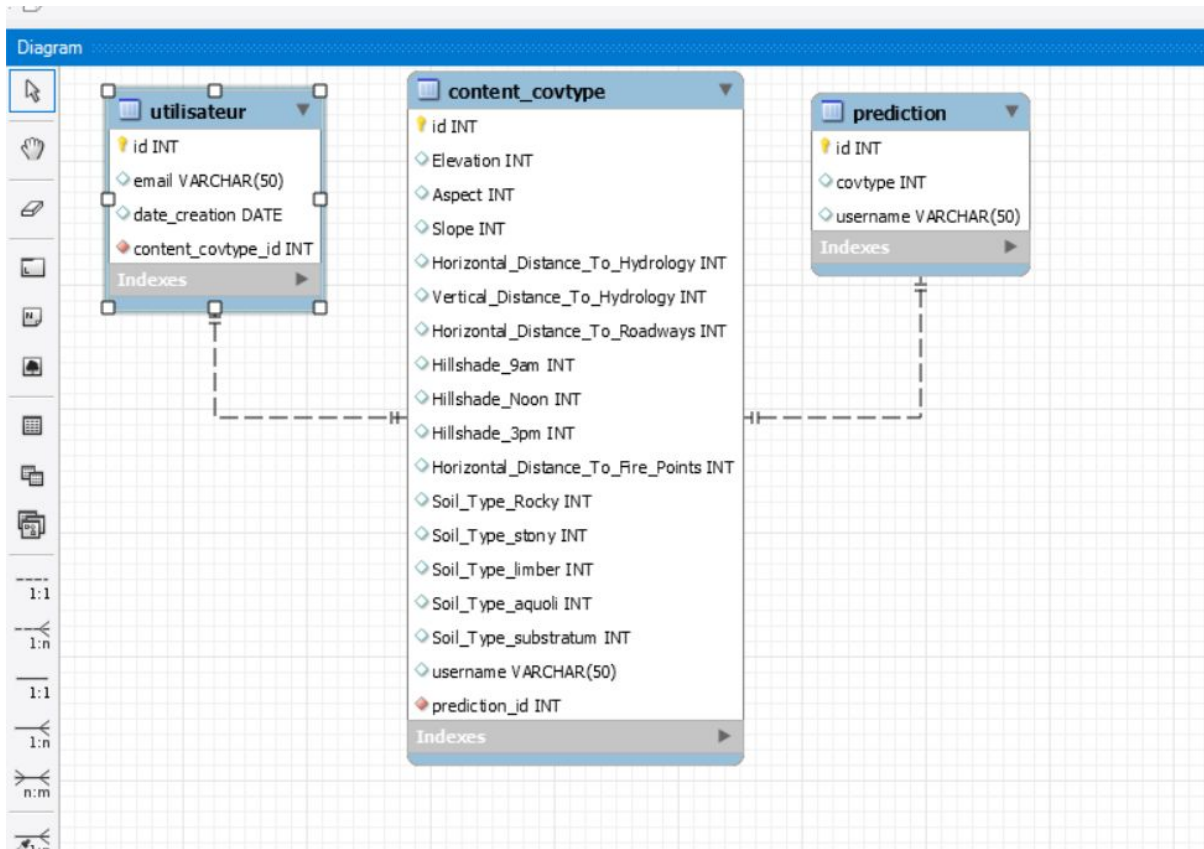
- Après avoir réalisé la structure et le développement des principales fonctionnalités de notre application, nous nous sommes intéressés à la configuration de notre base de données d'application. Pour créer une interaction entre la base de données et l'application nous avons installé dans notre environnement de développement Flask-SQLAlchemy. C'est une extension du framework Flask, facile d'utilisation. Elle fournit un ORM (Object Relational Mapping) permettant d'interagir avec la base de données d'application.
 - Les ORM
- Ils permettent de faire communiquer la base de données et l'application. Grâce à cette technique, les requêtes écrites python seront traduites en SQL puis les résultats de la requête seront renvoyés sous la forme d'objets Python avec lesquels il est possible d'interagir. Certaines modifications comme l'ajout ou la suppression d'items peuvent être apportées quel que soit le type de système de gestion de bases de données (relationnelle ou Nosql).
- Pour gérer ces interactions et mettre en œuvre les ORM, nous avons procédé en 3 étapes:
- création d'un fichier 'config.py' qui contient les configurations et paramètres essentiels que sont le code secret (SECRET KEY) et le paramètre SQLALCHEMY_DATABASE_URI qui est la chaîne de connexion dont nous avons besoin pour nous connecter à notre base de données.

```
from flask_sqlalchemy import SQLAlchemy
app.config['SECRET_KEY'] = 'motdep@sse'
app.config['SQLALCHEMY_DATABASE_URI'] = 'mysql://root:adama@localhost:3306/projet_final'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
```

- dans notre fichier __init__.py nous avons renseigné la présence et l'emplacement de la base de données
- models.py contient les requêtes de création des tables qui accueilleront les données de l'application ainsi que leur structure. Elles sont au nombre de 3 et sont directement créées à partir de Flask grâce aux ORM. La table 'utilisateur' pour enregistrer les identifiants des utilisateurs (avec les champs id, username, email). La table 'content_covtype' (15 variables au

total) qui va contenir les données de l'écosystème forestier renseignées et la table 'prediction' qui enregistre les résultats des prédictions.

Le schéma relationnel suivant réalisé avec l'outil '*reverse engineer*' de mysql Workbench permet de visualiser les relations entre les tables de notre base de données d'application.



- Sécurité de l'application

La sécurité de l'application est un élément important que nous ayons pris en compte dans le développement du back end de notre application pour faire face aux problèmes courants de sécurité liés à l'envoi des données à un serveur. Pour prévenir certaines failles de sécurité, les problèmes d'ingestion SQL et les attaques contre les serveurs, nous avons programmé les formulaires HTML en limitant la taille et le type de données susceptibles d'être saisies puis installé l'extension FLASKFORM qui permet de sécuriser le formulaire puis intégré dans notre application comme dans cet exemple ci-dessous:

```

from flask_wtf import FlaskForm
class RegistrationForm(FlaskForm):
    username= StringField(label='Username', validators=[DataRequired(), Length(min=3,
max=20)])
    email = StringField(label='Email', validators=[DataRequired()])
    password=PasswordField(label='Password', validators=[DataRequired(), Length(min=6,

```

```

max=16))
    submit = SubmitField(label='Sign UP')

class LoginForm(FlaskForm):
    email = StringField(label='Email', validators=[DataRequired()])
    password=PasswordField(label='Password', validators=[DataRequired(), Length(min=6,
max=16)])
    submit =SubmitField(label='Login')

```

4.3 Tests

- Tests unitaires

Nous avons fait des tests unitaires pour vérifier que certaines portions du code ont bien le comportement attendu. Par exemple :

=> Si l'utilisateur arrive à accéder à la page et à s'authentifier

=> Si le résultat de la prédiction s'affiche

=> Si les données sont bien envoyées dans la base de données d'application

- Tests fonctionnels

Tests fonctionnels réalisés nous ont permis de vérifier l'enchaînement de différentes fonctionnalités et voir qu'elles sont conformes aux besoins exprimés par les métiers. Ces tests visent à s'assurer que le logiciel renvoie les informations attendues par l'utilisateur pendant qu'il réalise une action. Concrètement, les actions que nous avons testées sont : de voir si l'utilisateur qui arrive sur la page de connexion, qui entre son identifiant et son mot de passe et qui clique sur le bouton de connexion, arrive bien sur la page d'accueil du formulaire de prédiction. Ci-dessous le tableau récapitulatif des tests effectués.

N°	Actions	Attendu	Résultats
1	Aller sur le site http... et cliquer sur le bouton connecter	<i>Ouverture de la page</i>	OK
2	Renseigner un email valide non enregistrée	Message d'erreur <i>"Désolé, vous devez vous identifier"</i>	OK
3	Renseigner un email sans @	Message d'erreur <i>"Veuillez entrer un email valide"</i>	OK
4	Renseigner des types de données différents d'entiers	Message d'erreur "Type de données non conforme"	OK
5	Entrer un email et un identifiant correct	Accès au formulaire	OK

Tester manuellement tous ces éléments peut prendre du temps c'est pourquoi nous avons mis en place d'un système pour réaliser les tests fonctionnels automatiquement avec l'installation de

Pytest. Pour l'implémenter nous avons créé un dossier 'tests' à la racine du projet et dans lequel il y a 2 fichiers principaux: test_app.py et config.py qui reprend les configurations de l'application.

Les tests de non-régression sont des catégories de test pour vérifier que la version en cours est bonne compte tenu d'éventuels changements de versions.

4.4 Systèmes de suivi de l'application

Le suivi d'une application passe par le monitoring automatique.

Les outils de monitoring comme Flask-Mail permettent de configurer un système d'alerte dans notre application Flask. Une fois installé, la programmation du script ci-dessous permettra à l'administrateur du site de recevoir des messages d'alerte automatiques lorsque des faits anormaux (tels que l'envoi d'un résultat de prédiction vide) sont constatés. Pour inclure ce processus dans notre application, il a fallu ajouter ces lignes de codes dans le fichier config.py de notre application.

```
app.config['MAIL_SERVER']='smtp.gmail.com'
app.config['MAIL_PORT'] = 465
app.config['MAIL_USERNAME'] = 'sowawa@gmail.com'
app.config['MAIL_PASSWORD'] = '*****'
app.config['MAIL_USE_TLS'] = False
app.config['MAIL_USE_SSL'] = True
```

Puis dans le fichier routes.py nous avons spécifié l'adresse email de l'administrateur qui recevra les messages d'alerte.

```
@app.route("/")
def index():
    msg = Message('Hello', sender = 'sowawa@gmail.com', recipients = ['ada@gmail.com'])
    msg.body = "Détection anomalies dans l'application!"
    mail.send(msg)
    return "Sent"
```

Pour la partie surveillance des performances de l'application, nous avons également utilisé AIRBRAKE et installé son extension flask nommé Pybrake.

Le but est d'avoir une vue d'ensemble des performances de notre application :

- de détecter et d'identifier les lenteurs et erreurs
- de mesurer les performances de la base de données

- d'analyser les requêtes de la base de données SQL appelée ainsi que les durées d'exécution.
- de suivre les échecs et les durées des transactions

Pour l'intégrer dans notre application. Il nous a suffi de rajouter ces lignes de codes suivantes dans le fichier app.py.

```
import pybrake.flask
app = Flask(__name__)
app.config['PYBRAKE'] = dict(
    project_id=123,
    project_key='SOWAWA',)
app = pybrake.flask.init_app(app)
```

4.5 Déploiement de l'Application et outils de mise en production

La mise en production marque le moment où le code de développement passe en production. La création d'un environnement de production sera donc nécessaire. Ainsi les fichiers de développement passeront vers un serveur relié à Internet. Cette phase marque l'étape où l'application va être accessible au public. Dans notre prochain sprint nous avons prévu d'effectuer cette phase de déploiement sur Heroku.

- Pourquoi Heroku?

Cette plateforme d'hébergement de site est connue pour son accès rapide. Il donne la possibilité d'héberger notre application dans le Cloud.

Pour compléter ce processus de mise en production, nous avons également prévu de réaliser étape concernant l'achat d'un nom de domaine, d'un espace serveur...

5 Gestion de projet

5.1 Organisation en méthode Agile

Nous avons mis en œuvre la méthode Agile plus précisément de la méthodologie Scrum. Cette méthode est basée sur “l'amélioration continue des processus de production”. Elle est basée sur les quatre principes que sont :


- Des **rôles** (le Scrum master, le product owner et l'équipe de développement occupent chacun une fonction bien précise)
- Des **événements** (il s'agit des événements qui encadrent les échanges, l'organisation et la durée du projet comme les prints)
- Des **artefacts** comprennent: le sprint backlog, le product backlog et l'incrément. C'est dans cette étape du projet que notre équipe a déterminé le temps nécessaire pour la réalisation de notre application
- Des **règles fondamentales de scrum** nous ont le plus poussé à choisir cette méthode sont:
 - privilégier les individus et interactions plutôt que les processus et outils
 - opter pour des solutions opérationnelles plutôt qu'une documentation exhaustive
 - favoriser la collaboration avec le client plutôt que des négociations contractuelles
 - l'adaptation au changement

5.2 Réalisation du planning projet

Après la formalisation du besoin client, nous avons élaboré un rétro planning qui répertorie les différentes tâches à effectuer ainsi que le temps estimé pour la réalisation de chaque sprint.

Afin de nous assurer de bien répondre aux besoins et attentes du client recensé, nous avons d'abord créé un persona et élaboré des users stories.

Création des users stories

	# User story 1 en tant qu'utilisateur j'ai besoin d'avoir accès à l'application qui me permet de prédire la classe d'arbre
Nom green	# User story 2 en tant qu'utilisateur j'ai besoin de saisir mes données afin d'obtenir le résultat de la prédiction
Âge Entre 25 et 54 ans	# User story 3 en tant qu'administrateur j'ai besoin d'une base de donnée d'application pour authentifier les utilisateurs
Niveau d'études technicien	
Secteur d'activité Gestion de forêts	

Planification des sprints

Ensuite la planification de sprints du projet en 4 sprints nous a permis de limiter le nombre de tâches en cours. Chaque itération dure 2 semaines à l'issue desquelles une liste de fonctionnalités utilisables du produit est réalisée. Pour organiser ces travaux nous avons utilisé des outils de gestion comme Trello et (En annexes 4 les extraits du rétro planning et du trello).

Utilisation de Github

L'une des leçons importantes du projet a été l'utilisation efficace de Github pour la gestion du flux de travail et du code de l'application en équipe.

5.3 Comptes rendus et communications avec le client

Echange 1 avec le client

Pascal Diop<pdio@gmail.com>

10 sept. 2020 08:30

A moi

Bonjour Ava,

est-ce-que vous avez pu avancer sur le sujet?

Serait-il possible de nous partager ton rétro planning?

Cordialement,

Pascal

Awa Sow<sowawa61@gmail.com>

10 sept. 2020 08:45

A Pascal

Bonjour Pascal,

Nous avons procédé à l'analyse approfondie des données mises à notre disposition pour la réalisation du programme d'LA et vous confirmons qu'elles sont en adéquation avec le projet.

L'ensemble des tâches nécessaires aux fonctionnalités de l'application ont également été définies et répertoriées dans le rétro-planning ci-joint.

Vous verrez aussi la durée de réalisation estimée pour chaque sprint.

je reste à votre disposition pour des questions.

Cordialement,

Awa

Echange 2

Awa Sow<sowawa61@gmail.com>

30 sept. 2020 08:00

A Pascal

Bonjour Pascal,

Suite à notre dernière réunion, nous avons répertorié toutes vos attentes conformément au cahier des charges que vous nous avez transmis.

La bonne nouvelle est que cette semaine, notre équipe a finalisé le fonctionnement back-end de l'application. La sécurité a été renforcée. Vous trouverez les détails de l'avancement du projet dans le compte rendu ci-joint.

nous restons à votre disposition.

Cordialement,

Awa

Pascal Diop<pdio@gmail.com>

30 sept. 2020 10:30

A moi

Bonjour Awa,

Je vous remercie pour le compte rendu.

A propos de l'interface utilisateur, nous voulons que vous prévoyez des couleurs sur la page d'accueil.

Nous voulons aussi que vous rajoutiez un champ 'date' dans la table utilisateur, pour avoir une visibilité sur les dates et heures de consultation du site.

Bien à vous,

Pascal

Echange 3

Awa Sow<sowawa61@gmail.com>

1 nov. 2020 08:00

A Pascal

Bonjour Pascal,

Les travaux de développement de l'interface de l'application et de la barre de navigation du site sont fonctionnels et conformes au cahier des charges.

Cependant, nous tenons à vous informer que la livraison prévue le 15 novembre risque d'être retardée de 2 jours. En effet, un problème inattendu a été détecté au cours des tests.

Je vous invite à en discuter lors de notre prochaine réunion.

Nous mettons tout en œuvre pour résoudre ce problème au plus tôt et pour une qualité du produit final.

Cordialement,

Awa

Pascal Diop<pdio@p@gmail.com>

1 nov. 2020 10:30

A moi

Bonjour Awa,

Nous regrettons un tel incident,

Nous discuterons plus amplement des modalités pour la suite du projet.

Bien à vous,

Pascal

Awa Sow<sowawa61@p@gmail.com>

4 nov. 2020 09:00

A Pascal

Bonjour Pascal,

je reviens vers vous suite à notre dernière réunion pour vous confirmer que les problèmes rencontrés étaient dus à des failles de sécurité du système de gestion des formulaires utilisateur. Heureusement, ces problèmes ont été résolus et la livraison pourra se faire dans les délais.

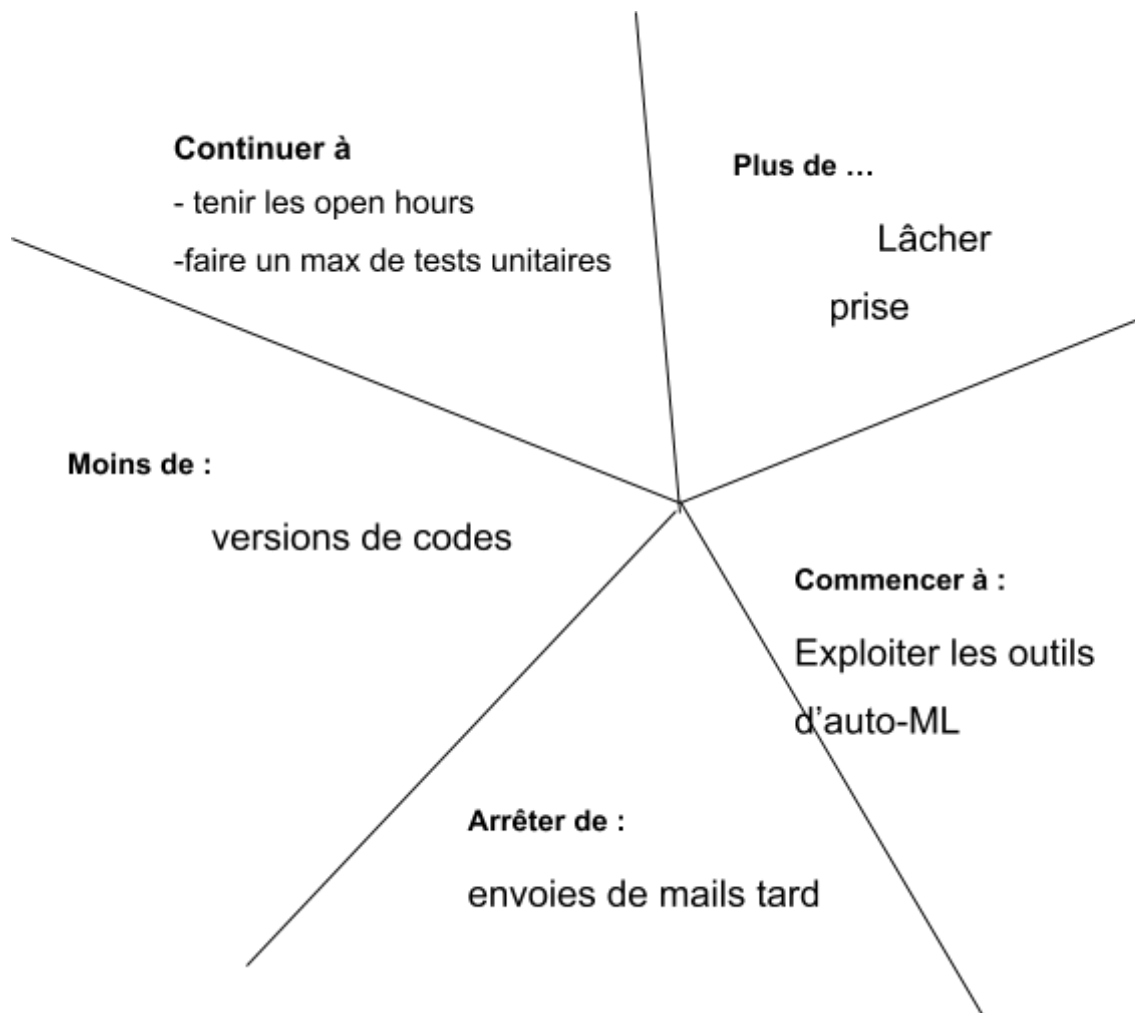
Je reste à votre écoute pour d'éventuelles questions.

Cordialement,

Awa

5.4 Rétrospectives

Ce graphisme suivant représente un starfish. Il comprend 5 zones dans lesquelles se résument un peu les points positifs et négatifs observés pendant la réalisation du projet.



CONCLUSION

Intégrer un programme d'IA dans les fonctionnalités d'une application est réalisable avec le langage python et des outils comme Flask. En suivant le processus déroulé dans cette présentation, il est tout à fait possible de mettre en œuvre une application fonctionnelle permettant de classifier des espèces d'arbre. Cette application donne des résultats satisfaisants qui correspondent à la réalité. Elle prouve que l'IA peut aider les humains à dépasser leurs limites pour assurer la continuité de leur mission de protection de l'environnement.

ANNEXES

Liens

<https://www.kaggle.com/uciml/forest-cover-type-dataset>

<https://inventaire-forestier.ign.fr/>

<https://firecaster.universita.corsica/>

<https://www.microsoft.com/en-us/ai/ai-for-earth>

<https://openclassrooms.com/fr/courses/4425066-concevez-un-site-avec-flask/>

https://python-adv-web-apps.readthedocs.io/en/latest/flask_db3.html

<https://www.sciencedirect.com/science/article/abs/pii/S0957417419308693>

Bibliographie

- *Machine Learning avec Scikit-Learn - 2e éd. - Mise en œuvre et cas concrets* (Aurélien Géron)
- *Python pour le data scientist Des bases du langage au machine learning* Parution : juillet 2019 (Emmanuel Jakobowicz)
- *Data Mining et statistique décisionnelle* (Stéphane Tufféry) Paru en octobre 2017

Annexe 1 : Fonction de détection des valeurs aberrantes

```
: def outlier_function(df, col_name):
    ''' Cette fonction detecte le premier et le 3e quartile et extrait les valeurs qui sont en dessous ou au
    dessus de ces dernières puis retourne leurs nombres respectifs
    '''
    first_quartile = np.percentile(np.array(df[col_name].tolist()), 25)
    third_quartile = np.percentile(np.array(df[col_name].tolist()), 75)
    IQR = third_quartile - first_quartile

    upper_limit = third_quartile+(3*IQR)
    lower_limit = first_quartile-(3*IQR)
    outlier_count = 0

    for value in df[col_name].tolist():
        if (value < lower_limit) | (value > upper_limit):
            outlier_count +=1
    return lower_limit, upper_limit, outlier_count

: # choix des variables continues uniquement
df_tree = df.iloc[:, :10]
# loop through all columns to see if there are any outliers
for column in df_tree.columns:
    if outlier_function(df_tree, column)[2] > 0:
        print("Il y a {} outliers dans {}".format(outlier_function(df_tree, column)[2], column))

Il y a 275 outliers dans Slope
Il y a 414 outliers dans Horizontal_Distance_To_Hydrology
Il y a 5339 outliers dans Vertical_Distance_To_Hydrology
Il y a 1027 outliers dans Hillshade_9am
Il y a 1191 outliers dans Hillshade_Noon
Il y a 10 outliers dans Horizontal_Distance_To_Fire_Points
```

Annexe 2: Script de création de la base de données et d'insertion dans la table

```
-- 1 création Base de données nommée projet_final;
CREATE DATABASE projet_final;

-- 2 création table pour intégrer les données
CREATE TABLE projet_final.covtype
(Elevation int(10), Aspect int(12), Slope int(15), Horizontal_Distance_To_Hydrology int(10),
Vertical_Distance_To_Hydrology int(10), Horizontal_Distance_To_Roadways int(10), Hillshade_9am
int(14),Hillshade_Noon int(14), Hillshade_3pm int(16), Horizontal_Distance_To_Fire_Points
int(10),Wilderness_Area1 int(10), Wilderness_Area2 int(10), Wilderness_Area3 int(10),
Wilderness_Area4 int(10),Soil_Type1 int(10), Soil_Type2 int(10), Soil_Type3 int(10),
Soil_Type4 int(10), Soil_Type5 int(10),Soil_Type6 int(10), Soil_Type7 int(10),
Soil_Type8 int(10),Soil_Type9 int(10), Soil_Type10 int(10) ,Soil_Type11 int(10),
Soil_Type12 int(10) , Soil_Type13 int(10), Soil_Type14 int(10),Soil_Type15 int(10),
Soil_Type16 int(10), Soil_Type17 int(10), Soil_Type18 int(10),Soil_Type19 int(10),
Soil_Type20 int(10), Soil_Type21 int(10), Soil_Type22 int(10),Soil_Type23 int(10),
Soil_Type24 int(10), Soil_Type25 int(10), Soil_Type26 int(10),Soil_Type27 int(10) ,
Soil_Type28 int(10), Soil_Type29 int(10), Soil_Type30 int(10),Soil_Type31 int(10),
Soil_Type32 int(10), Soil_Type33 int(10), Soil_Type34 int(10),Soil_Type35 int(10),
Soil_Type36 int(10), Soil_Type37 int(10), Soil_Type38 int(10),
Soil_Type39 int(10), Soil_Type40 int(10), Cover_Type int(10) ) ;

-- 3 insertion des données initiales
```



```

USE projet_final;
LOAD DATA INFILE 'C:\Users\sowaw\data_covtype\covtype.csv' -- chemin du fichier en local
INTO TABLE covtype
CHARACTER SET 'utf8'
FIELDS ESCAPED BY '\\'
TERMINATED BY ';'
OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\r\n'
;

```

Affichage résultat après exécution du script

```

25     TERMINATED BY ';'
26     OPTIONALLY ENCLOSED BY '"'
27     LINES TERMINATED BY '\r\n'
28     ;
29 • Select * from projet_final.covtype
30     limit 3

```

Elevation	Aspect	Slope	Horizontal_Distance_To_Hydrology	Vertical_Distance_To_Hydrology	Horizontal_Distance_To_Roadways	Hillshade_9am	Hillshade_Noon	Hillshade_3pm	Horizontal_Distance
2596	51	3	258	0	510	221	232	148	6279
2590	56	2	212	-6	390	220	235	151	6225
2804	139	9	268	65	3180	234	238	135	6121

Annexe 3 : Script Entraînement modèle Machine Learning avec Gridsearch

TRAINING MODEL

```

5]: # EXTRATREE CLASSIFIER MODEL
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.metrics import classification_report
from sklearn.model_selection import cross_validate, GridSearchCV, KFold, RandomizedSearchCV
# setting parameters
etc_para = [{'n_estimators':[20,30,100], 'max_depth':[5,10,15], 'max_features':[0.1,0.2,0.3]}]
# Default number of features is sqrt(n)
ETC = GridSearchCV(ExtraTreesClassifier(),param_grid=etc_para, cv=5, n_jobs=-1)
ETC.fit(X_train_res, y_train_res)

C:\Users\sowaw\anaconda3\envs\env_projet_final\lib\site-packages\sklearn\model_selection\_search.py:765: DataConversionWarning:
A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().
    self.best_estimator_.fit(X, y, **fit_params)

5]: GridSearchCV(cv=5, estimator=ExtraTreesClassifier(), n_jobs=-1,
    param_grid=[{'max_depth': [5, 10, 15],
                  'max_features': [0.1, 0.2, 0.3],
                  'n_estimators': [20, 30, 100]}])

5]: ETC.best_params_

5]: {'max_depth': 15, 'max_features': 0.3, 'n_estimators': 100}

71: ETC.best_score_

```

Annexe 4 : Requête d'accès aux données avant entraînement modèle

```
import pymysql
import pandas as pd
import os

host = 'localhost'
password = 'adama'
database = 'projet_final'
conn = pymysql.connect(
    host=host,
    port=int(3306),
    user="root",
    passwd=password,
    db=database,
    charset='utf8mb4')
df = pd.read_sql_query( "SELECT * FROM covtype", conn)
```

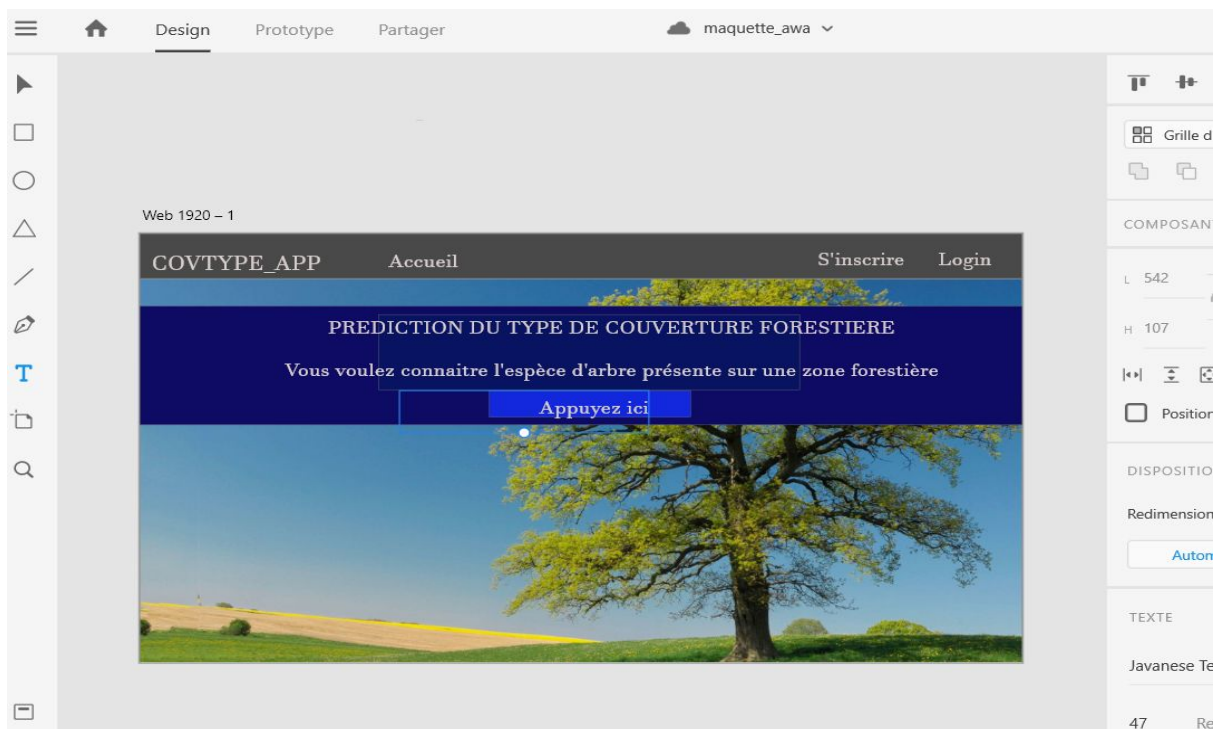
```
df = pd.read_sql_query("SELECT * FROM covtype",
    conn)
```

df

	Elevation	Aspect	Slope	Horizontal_Distance_To_Hydrology	Vertical_Distance_To_Hydrology	Horizontal_Distance_To_Roadways	Hillshade_9am	Hillshade
0	2596	51	3	258	0	510	221	
1	2590	56	2	212	-6	390	220	
2	2804	139	9	268	65	3180	234	
3	2785	155	18	242	118	3090	238	
4	2595	45	2	153	-1	391	220	
...
581007	2396	153	20	85	17	108	240	
581008	2391	152	19	67	12	95	240	
581009	2386	159	17	60	7	90	236	
581010	2384	170	15	60	5	90	230	
581011	2383	165	13	60	4	67	231	

581012 rows × 9 columns

Annexe 5 : Maquettes application



Pour prédire l'espèce d'arbre, veuillez renseigner le formulaire ci-dessous

Elévation	Quel est l'élévation du sol en mètres ?
Pente	Quel est la pente du sol en mètres ?
Eclairage	Quelle est l'éclairage '0-255' à 9h ?
Distance1	Distance horizontale avec le plus proche point d'eau ?
Distance2	Verticale horizontale avec le plus proche point d'eau ?
Distance3	Distance horizontale avec la plus proche route ?
Sol rocheux	<input type="radio"/> OUI <input type="radio"/> NON
Sol argileux	<input type="radio"/> OUI <input type="radio"/> NON

Soumettre

La maquette présente une interface web avec un header sombre contenant le logo 'COVTYPE_APP' et les liens 'Accueil', 'S'inscrire' et 'Login'. Le contenu principal est sur un fond bleu foncé avec le titre 'PREDICTION DU TYPE DE COUVERTURE FORESTIERE' et le texte 'Vous voulez connaître l'espèce d'arbre présente sur une zone forestière'. Un bouton 'Appuyez ici' est visible. En dessous, une image d'un arbre sur une colline est affichée. À droite, une barre latérale contient des options de design comme 'Grille d', 'COMPOSAN', 'L 542', 'H 107', 'Position', 'DISPOSITIO', 'Redimension', 'Auton', 'TEXTE', 'Japanese Te', '47' et 'Re'.

La classe d'arbre prédite est:

L'Epicéa



Annexe 5 : Schéma dossier développement application

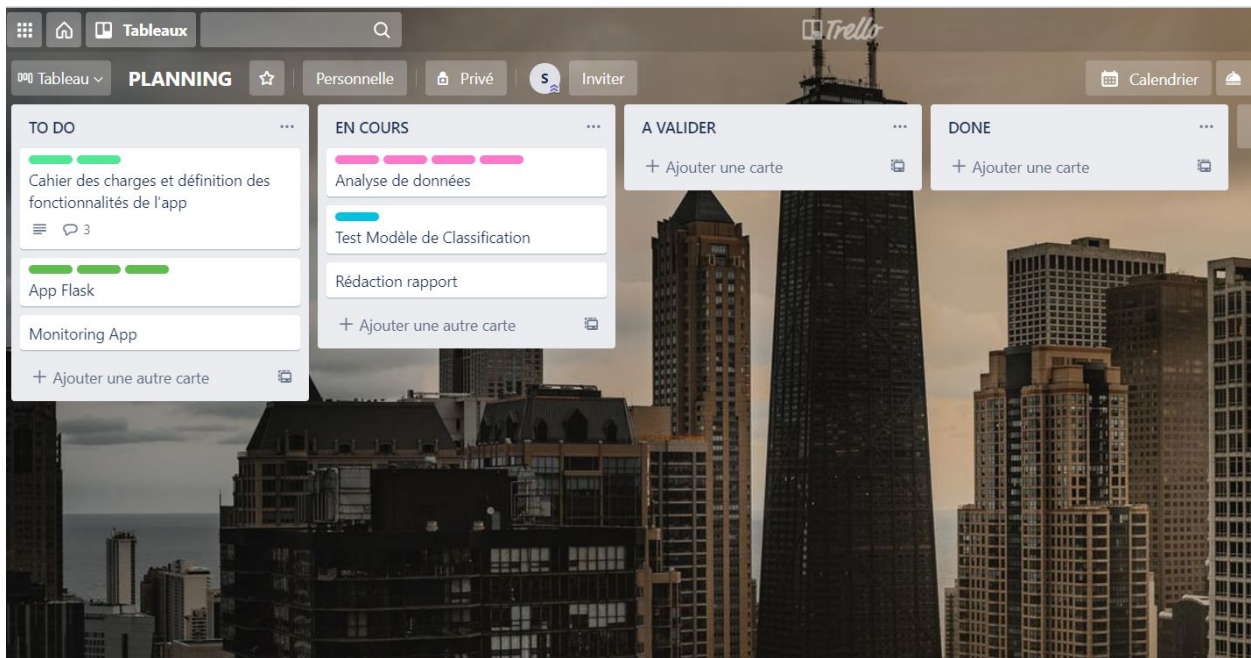
```
├── APP_AWA
│   ├── venv #environnement
│   └── simplon
│       ├── __init__.py # setup app
│       ├── routes.py # routes application
│       ├── models.py # model application
│       ├── awa.db # base de donnée
│       └── templates
│           ├── homepage.html # graphisme du page d'accueil
│           ├── index.html # formulaire pour la prédiction du couvert forestier
│           └── after.html # affichage résultat de la prédiction
│       ├── login.html # page d'authentification
│       └── signup.html # confirmation inscription
├── config.py # fichier de configuration de l'application
└── tests # dossier test à la racine du projet
```

Annexe 6 : Outils gestion de projet: Rétro planning

Rétro Planning

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	
		OCTOBRE 2020																																		
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	1	2	3	
Tâche à accomplir	État																																			
Compréhension du besoin (8 jours)																																				
Modélisation du besoin	Terminée																																			
Rédaction du cahier des charges	Terminée																																			
Création d'un Trello pour gestion de projet	terminée																																			
Management /Exploration des données analytiques 8 jrs																																				
Compréhension des données	Terminée																																			
Import/nettoyage	Terminée																																			
analyse exploratoire des données	Terminée																																			
visualisations	Terminée																																			
Stockage des données	Terminée																																			
développement programme d'IA (8 jours)																																				
entrainement Modeles Classification multi	Terminée																																			
Comparaison modeles	Terminée																																			
Validation /tests	Terminée																																			
Evaluation modèle	Terminée																																			
Développement application (21 jours)																																				
Choix Nom/ maquette appli	Terminée																																			
Choix des spécification fonctionnelles	En cours																																			
programmation app Flask	En cours																																			
Creation base de données d'application	En cours																																			
Controle appli/suivi/Monitoring	A faire																																			

[illegible]



manaya2019 / Chef_d_oeuvre_algos Private

[Code](#)
[Issues](#)
[Pull requests](#)
[Actions](#)
[Projects](#)
[Security](#)
[Insights](#)
[Setting](#)

[master](#)
[1 branch](#)
[0 tags](#)
[Go to file](#)

File	Description
data	Ajout data
.gitignore	on nettoie le repository
README.md	Create README.md
Untitled-Copy1.ipynb	Modele ml with resampled data
classification_model.ipynb	test smote pour rééquilibrage donnees
exploration_dataset_and_first_train.ip...	Modele ml with resampled data
model_classif_resampled_data.ipynb	Modele ml with resampled data
modele2_sgd.ipynb	Modele ml with resampled data
sgd_model1.sav	test smote pour rééquilibrage donnees