|  |
| --- |
| [Tapez le nom de la société] |
| PROJET CHEF D’ŒUVRE RAPPORT E1 |
| Détection de recommandation d’un produit de prêt-à-porter |
|  |
| **Marianne DEPIERRE** |
| **20/01/2023** |

|  |
| --- |
| Titre professionnel RNCP "Développeur en intelligence artificielle" |

Table des matières

[1. Introduction 3](#_Toc126383558)

[2. Contexte, problématique et objectifs 3](#_Toc126383559)

[2.1. Présentation de l’entreprise 3](#_Toc126383560)

[2.2. Présentation du projet 3](#_Toc126383561)

[2.3. Enjeu du projet 3](#_Toc126383562)

[2.4. Cahier des charges 4](#_Toc126383563)

[3. Spécifications fonctionnelles 4](#_Toc126383564)

[3.1. Diagramme des cas d’utilisation 4](#_Toc126383565)

[3.2. Modèle de données 5](#_Toc126383566)

[3.3. Maquettes 5](#_Toc126383567)

[4. Spécifications techniques 6](#_Toc126383568)

[4.1. Flask 6](#_Toc126383569)

[4.2. Bootstrap 6](#_Toc126383570)

[4.3. MLflow Tracking 6](#_Toc126383571)

[4.4. Weights & Biases (WandB) 6](#_Toc126383572)

[4.5. MySQL Workbench 6](#_Toc126383573)

[4.6. Azure 6](#_Toc126383574)

[4.7. Tableau récapitulatif des outils de développement 7](#_Toc126383575)

[5. Gestion de projet 7](#_Toc126383576)

[5.1. Planning du projet 7](#_Toc126383577)

[5.2. Communication avec le client et comptes rendus 8](#_Toc126383578)

[5.3. Versioning du projet avec Git & GitHub 9](#_Toc126383579)

[6. Base de données 10](#_Toc126383580)

[7. Jeu de données et exploration des données 11](#_Toc126383581)

[7.1. Description générale du jeu de données et valeurs manquantes 11](#_Toc126383582)

[7.2. Observations dupliquées 12](#_Toc126383583)

[7.3. Déséquilibre de la répartition des classes dans la variable cible 13](#_Toc126383584)

[7.4. Analyse univariée et multivariée des variables numériques et catégorielles 13](#_Toc126383585)

[7.5. Analyse des données textuelles 13](#_Toc126383586)

[8. Traitement des données 15](#_Toc126383587)

[9. Modélisation 17](#_Toc126383588)

[9.1. Choix des modèles algorithmiques d’analyse de sentiments 18](#_Toc126383589)

[9.2. Entraînement et évaluation du modèle 18](#_Toc126383590)

[9.2.1. Entraînement du modèle 18](#_Toc126383591)

[9.2.2. Evaluation du modèle 18](#_Toc126383592)

[9.3. Performance des modèles testés et versioning du modèle 19](#_Toc126383593)

[9.3.1. Performance des modèles testés 19](#_Toc126383594)

[9.3.2. Versioning du modèle avec mlflow 19](#_Toc126383595)

[9.4. Evaluation du modèle retenu 20](#_Toc126383596)

[9.4.1. Learning curve 20](#_Toc126383597)

[9.4.2. Rapport de classification, matrice de confusion, AUC et courbe ROC 20](#_Toc126383598)

[9.4.3. Interprétation du modèle 22](#_Toc126383599)

[10. Application et déploiement 25](#_Toc126383600)

[10.1. Application 25](#_Toc126383601)

[10.1.1. Arborescence de l’application 25](#_Toc126383602)

[10.1.2. Vérification des entrées du formulaire recueillant l’avis du client 25](#_Toc126383603)

[10.1.3. Fonctionnement de l’application 26](#_Toc126383604)

[10.2. Tests unitaires 27](#_Toc126383605)

[10.3. Déploiement 28](#_Toc126383606)

[11. Monitoring du modèle avec WandB 28](#_Toc126383607)

[12. Bilan 29](#_Toc126383608)

[13. Annexes 30](#_Toc126383609)

# Introduction

Ce rapport est décrit un projet, appelé « Projet chef d’œuvre  » développé dans le cadre de formation en Intelligence Artificielle de 19 mois de Simplon. Ce projet m’a permis de mettre en application la plupart des techniques et connaissances accumulées tout au long de cette période de formation, notamment en deep learning (ou apprentissage profond). L’objectif principal de ce projet est donc de valider les compétences pour le titre professionnel RNCP « Développeur en Intelligence Artificielle ».

# Contexte, problématique et objectifs

## Présentation de l’entreprise

Sanofi est une entreprise pharmaceutique française. Le groupe s’est construit progressivement, suite aux différentes acquisitions et fusions. Sanofi se compose actuellement de quarante-neuf sites en France, industriels et tertiaires. Cette entreprise est aussi très présente à l’international, puisque située dans cent pays. L’intelligence artificielle et l’analyse des données sont donc utilisées pour des tâches et des buts très variés : en recherche et développement pour l’innovation pharmaceutique, mais aussi pour des tâches plus tertiaires. Ainsi mes missions dans cette entreprise concernent la création et la maintenance de chatbots (ou robot conversationnels) pour diverses unités de l’entreprise : finance, logistique, ressources humaines, qualité…

## Présentation du projet

Aucune des missions que j’ai exécutées chez Sanofi ne me permet de valider toutes les compétences du projet chef d’œuvre à réaliser pour la certification. En effet, la construction des chatbots se fait grâce au module IBM Watson Assistant des solutions IBM Watson. Par conséquent, les modèles de Traitement Automatique du Langage Naturel (TALN ou NLP en anglais) utilisés sont ceux d’IBM Watson. Cependant un besoin a émergé lors des recherches pour un nouveau projet en vu de la certification : celui d’évaluer le ressenti des utilisateurs face aux chatbots à partir des commentaires qu’ils ont laissé lors de leurs utilisations. Après analyse des bases de données des différents chatbots, les données se sont révélées inutilisables pour un projet d’analyse de sentiment (ou sentiment analysis/opinion mining). En attendant des données valides, un Proof Of Concept permet de vérifier la viabilité du projet.

Le présent projet est donc un PoC montrant ce qu’il est possible de réaliser avec un modèle de TALN pour des problématiques d’analyse de sentiments. Le choix s’est porté sur des données concernant des commentaires sur des produits de prêt-à-porter.

## Enjeu du projet

L’analyse de sentiment est le processus déterminant l’opinion, le jugement ou l’émotion exprimé par le langage naturel. Lorsque les clients laissent des avis en ligne et formulent des commentaires sur un produit, leurs appréciations reflètent un sentiment, positif ou négatif. A terme, l’analyse de sentiment permet d’extraire des informations à partir d’un grand volume de données textuelles et de les généraliser de façon automatique. A partir de ces informations, l’entreprise peut prendre des décisions stratégiques relatives à un produit ou un domaine.

L’enjeu de ce projet est d’intégrer dans une application un modèle de deep learning capable de détecter la recommandation (sentiment positif) ou non (sentiment négatif) d’un produit de prêt-à-porter à partir du commentaire laissé par un client.

## Cahier des charges

Plusieurs objectifs ont été fixés pour ce projet :

* Le modèle de deep learning détecte un sentiment positif (produit recommandé) ou un sentiment négatif (produit non recommandé).
* Le modèle est déployé sous forme d’application web.
* Un formulaire permet de soumettre un commentaire et d’estimer son sentiment.
* L’affichage des résultats de la prédiction dans l’application est clair et rapide.
* Une base de données recueille les prédictions du modèle.
* Si le monitoring (ou suivi) du modèle montre un score inférieur à un seuil, un email est automatiquement reçu et alerte de la baisse de performances du modèle (alerting).
* Les livrables attendus sont : l’application, le modèle, le présent rapport « chef d’œuvre » et une présentation orale du projet.

# Spécifications fonctionnelles

## Diagramme des cas d’utilisation

L’expression du besoin a été traduite en diagramme UML de cas d’usages.

|  |  |
| --- | --- |
|  | Il y a deux cas d’utilisation :   * Cas du visiteur non inscrit :   Il peut consulter la page d’accueil de l’application web et s’inscrire.   * Cas de l’utilisateur :   Il peut consulter la page d’accueil et effectuer une prédiction s’il s’est authentifié. |

## Modèle de données

Ci-dessous se trouve le Modèle Physique de Données (MPD) de ma base de données.



Le MPD prévoit 6 tables :

* clothes : contient les informations relatives à un vêtement, soit son identifiant, sa division, son département et sa classe.
* reviews : regroupe les informations concernant l’avis qu’un client a soumis.
* divisions : correspond à l’identifiant et le nom de la division.
* classes : contient l’identifiant et le nom de la classe.
* departments : contient l’identifiant et le nom du département.
* predictions : regroupe les prédictions que le modèle de l’application a réalisé. La table contient le texte soumis, le résultat et le score de la prédiction du modèle.

## Maquettes

|  |  |
| --- | --- |
| C:\Users\Admin\AppData\Local\Microsoft\Windows\INetCache\Content.Word\wireframe_accueil_app-moqups-com.png | C:\Users\Admin\AppData\Local\Microsoft\Windows\INetCache\Content.Word\wireframe_recommandation_app-moqups-com.png |
| *Page d’accueil (utilisateur non connecté)* | *Page de recommandation* |

Des maquettes ont été réalisées pour valider l’apparence de l’application avec le client. Le support ordinateur est préféré par le client en accord avec le besoin exprimé. Sur les pages du site internet, une image est affichée en fond. Le menu est centré en haut de page et un bloc centré et fixé en bas de page permet d’afficher le texte ou le formulaire de la page.

# Spécifications techniques

Le choix des technologies suivantes s’est fait en accord avec les besoins exprimés par le client et avec le temps disponible restreint pour réaliser ce projet. Les technologies privilégiées sont donc ciblées pour la variété des options des options qu’elles proposent, la rapidité d’apprentissage et de développement et leur prix (gratuité).

## Flask

Flask est un framework d'application web écrit en Python. Un framework est une collection de bibliothèques et de modules qui permettent aux développeurs d'applications web d'écrire des applications en simplifiant et unifiant leur travail. Flask est basé sur la boîte à outils WSGI Werkzeg et le moteur de template Jinja2. Flask est considéré comme un micro-framework, car, par exemple qu’il n’apporte pas de base de données ou d’ORM prédéfinis par exemples. L’apprentissage de Flask est relativement simple, comparé à Django par exemple, et c’est pourquoi le choix s’est porté sur cet outil.

## Bootstrap

Bootstrap est un outil de développement front-end. Il met à disposition un système de grille, des composants préconstruits et des plugins JavaScript permettant de coder un site responsive efficacement. Il est compatible avec la majorité des navigateurs.

## MLflow Tracking

MLflow Tracking est une API et une interface utilisateur permettant de consigner des paramètres, des versions de code, des métriques ou des fichiers de sortie lors de l'exécution du script d'apprentissage automatique. MLflow Tracking fonctionne par «  run », c’est-à-dire l’exécution d'une partie du script. Chaque exécution enregistre les informations codées par le développeur.

## Weights & Biases (WandB)

Tout comme MLflow Tracking, Weights & Biases est un outil qui permet de suivre et facilement enregistrer les performances de modèles. Les hyperparamètres, les mesures de sorties et les logs sont sauvegardés. Weights & Biases propose également des outils de visualisation. Cet outil propose aussi gratuitement un service d’envoi d’alerte email.

## MySQL Workbench

MySQL Workbench est propose une interface pour créer, gérer et administrer les bases de données MySQL (un système de gestion de base de données relationnelles). Il fournit des outils pour la modélisation des données, le développement SQL et pour l'administration et la configuration du serveur.

## Azure

Azure est la plateforme de cloud computing de Microsoft. Elle intègre des services de plateforme (PaaS) et d’infrastructure (IaaS) pour par exemple exécuter ou provisionner des machines virtuelles. Avec les services App, Azure propose un ensemble de services pour la création, la gestion, l’administration et le déploiement des applications. Azure Sevices a été sélectionnée car les modèles de deep learning sont particulièrement lourds et le client est un partenaire de Microsoft.

## Tableau récapitulatif des outils de développement

Pour répondre aux besoins du client, un environnement de développement a été défini comme suit.

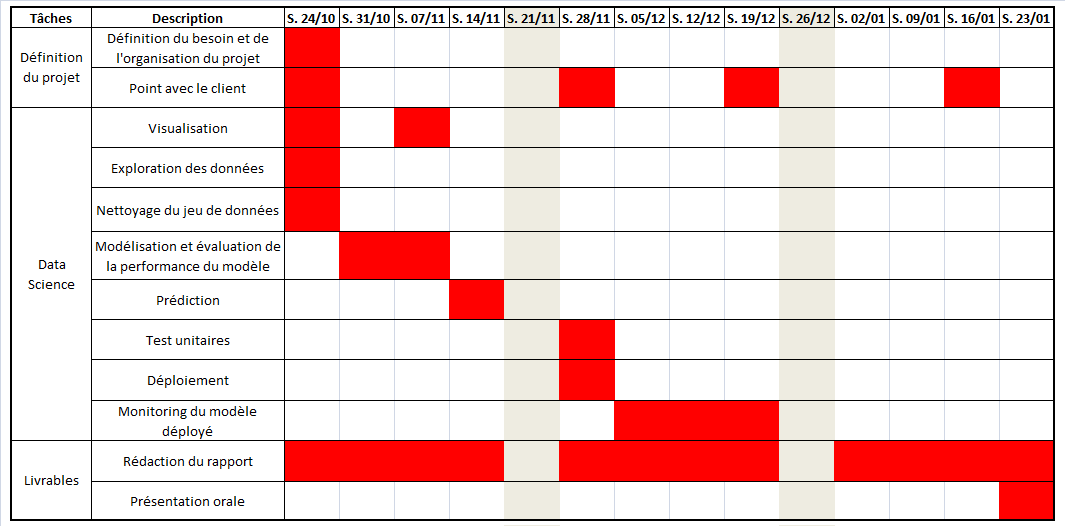
|  |  |  |
| --- | --- | --- |
| Langages | Outils | Librairies python |
| Python 3.8 | IDE : Jupyter Notebook et Visual Studio Code | Monitoring / Versioning : mlflow, wandb |
| HTML5 | Frameworks : Flask et Bootstrap | Modélisation / traitement des données :  scikit-learn,tensorflow, imblearn, nltk, LIME |
| CSS3 | Azure | Visualisation : matplotlib, seaborn |
|  | Base de données : MySQL Workbench, sqlite | Pickle : pickle |
|  | Versioning : Git et Github | Tests unitaires : unittest |
|  |  | Utilitaires : pandas, numpy, sys, os |
|  |  | Base de données : mysql, sqlite |
|  |  | Application : flask, flask\_login, flask\_sqlalchemy |
|  |  | Environnement virtuel : virtualenv |

Les langages, les librairies python et les outils mentionnés permettent de créer et évaluer un modèle de deep learning qui détecte le sentiment d’un texte. Ils permettent également le développement et le déploiement d’une application. Enfin ils remplissent les objectifs de mise en place d’une base de données, de monitoring du modèle et d’alerting.

# Gestion de projet

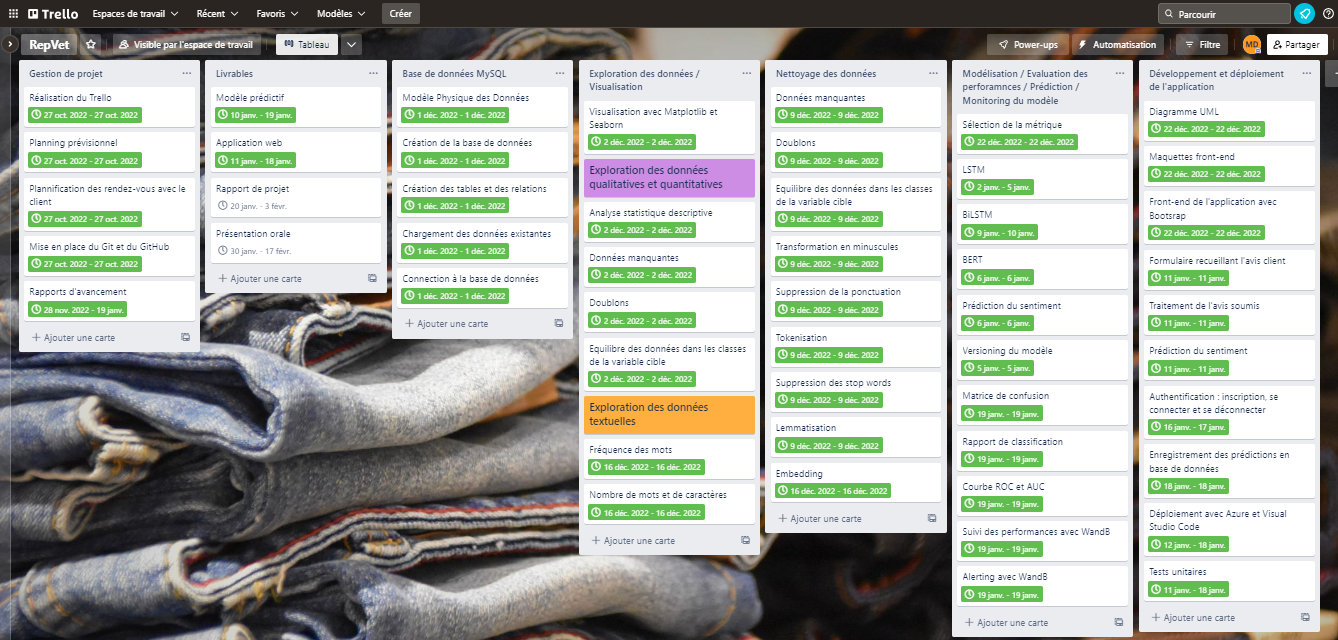
## Planning du projet

Le planning prévisionnel du projet a été défini comme suit.



Ce planning représente en rouge les « semaines » consacrées à une tâche du projet. En grisé ce sont les semaines de congés, non dédiées à ce projet. Excepté le mois de janvier, le nombre de jours consacrés au projet est de deux jours maximum par semaine. Les autres jours de ces semaines, j’ai travaillé sur d’autres projets de Sanofi.

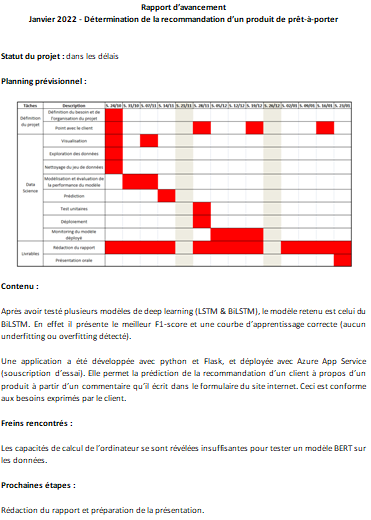
Un planning réel a été effectué avec l’outil Trello. Il s’agit d’un outil de gestion de projet en ligne qui organise des projets avec des cartes représentant des tâches.



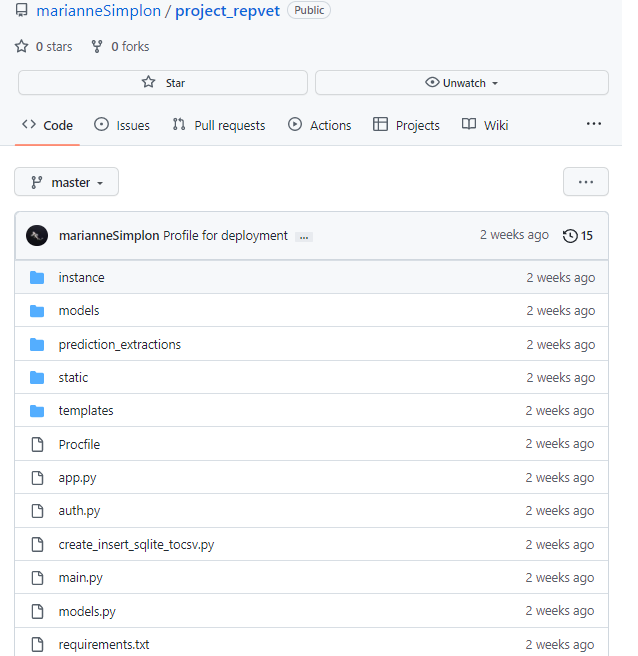
En résumé, un total de 29 jours ont pu être consacrés au projet, de sa définition à l’élaboration des livrables. Le projet a effectivement débuté le 27 octobre 2022 avec l’organisation du projet. Cependant les tâches techniques ont commencé en décembre. Globalement, l’exploration et le traitement des données se sont faites en décembre. La modélisation et l’application ont été réalisées au mois de janvier 2023. Enfin la rédaction du rapport ne s’est pas faite progressivement mais à partir du 20 janvier et jusqu’à la date du 3 février. La préparation de l’oral a débuté le 30 janvier et se poursuit actuellement.

## Communication avec le client et comptes rendus

La communication avec le client s’est effectuée par échanges directs tous les mois et par échanges d’emails plus réguliers. La communication avec le reste de l’équipe s’est fait avec Microsoft Teams (visioconférences et messagerie). Des comptes rendus mensuels ont été écrits à l’attention du client pour faire état de l’avancement du projet. Un exemple de rapport d’avancement est présenté ci-dessous (voir les annexes XXX et XXX pour les autres rapports).



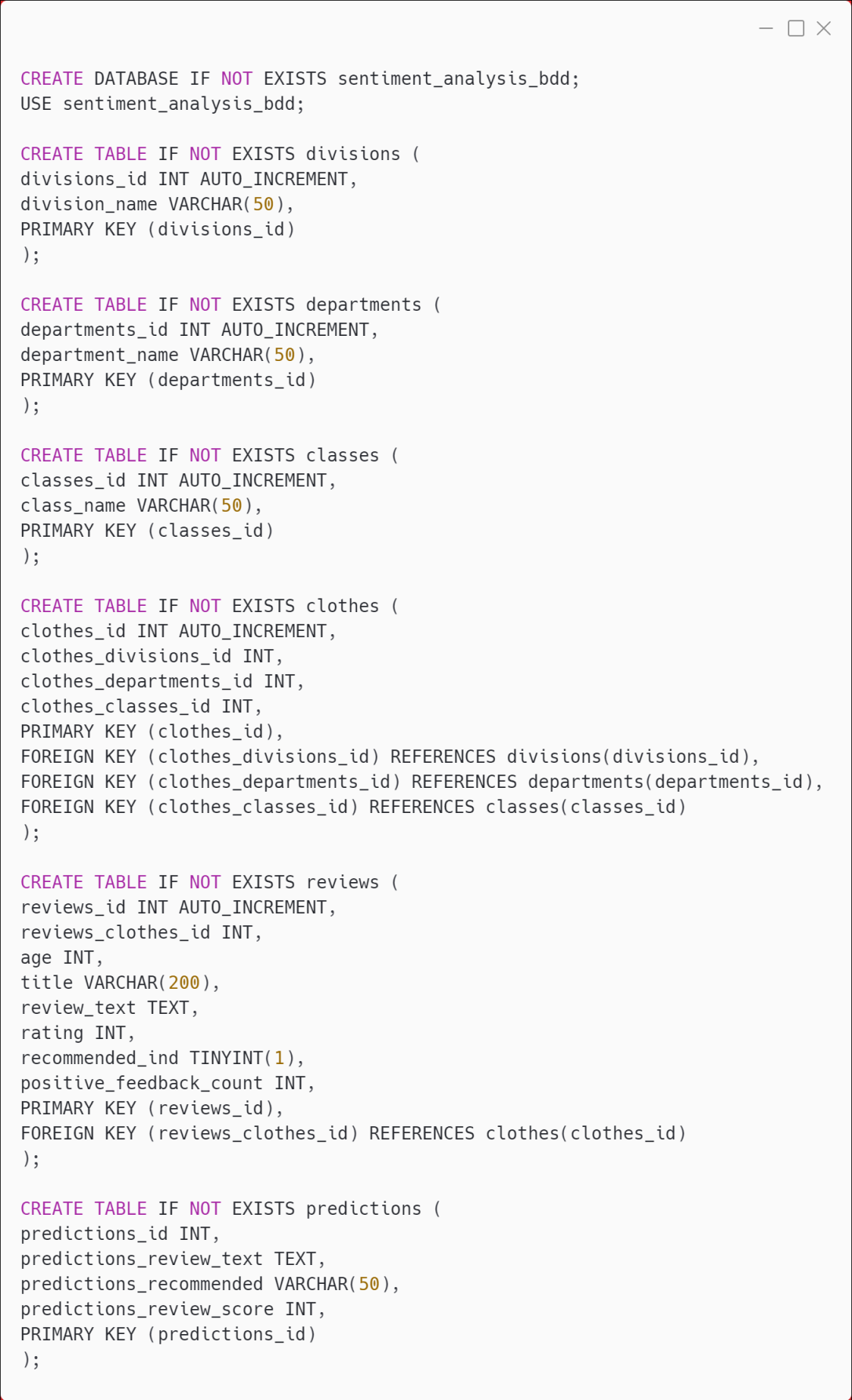
## Versioning du projet avec Git & GitHub



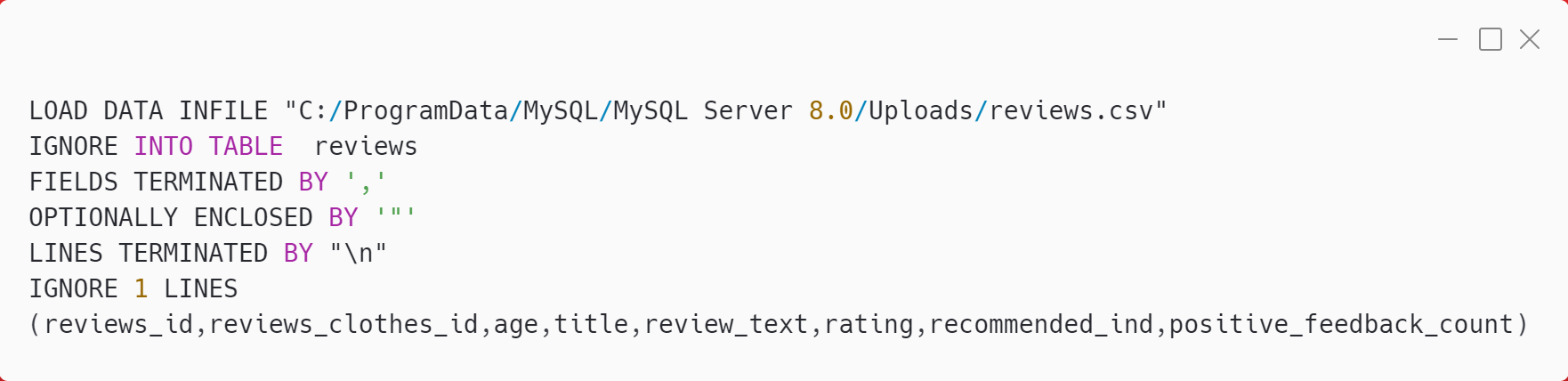
Le suivi du projet a été réalisé avec Git, un logiciel de gestion de versions. Les versions du code pu être régulièrement consultées par l’équipe et gérées en ligne avec GitHub (voir ci-contre).

# Base de données

La base de données de l’application a été créée selon le Modèle Physique de Données décrit dans la section « Spécifications fonctionnelles ». Un extrait du script permettant de créer la base de données et les tables est disponible ci-dessous (voir annexe XXX pour l’entièreté du script).



Les données sont diffusées sous format CSV (Comma Separated Values). A l’aide d’un script SQL et de MySQL Workbench, elles sont chargées dans la base de données MySQL. Ci-dessous se trouve l’un des scripts SQL permettant de charger les données pour la table « reviews ».



Après avoir mis en place un environnement virtuel avec la librairie virtualenv, les données sont lues dans un notebook. Pour charger les données dans Jupyter Notebook, une connexion à la base de données mysql est établie, puis les données sont importées dans un DataFrame de pandas en utilisant les deux fonctions ci-dessous.





# Jeu de données et exploration des données

La visualisation et l’exploration des données sont indispensables dans un projet de machine ou deep learning. Elles permettent de comprendre le contenu et de traiter les données en adéquation avec le projet. Elles aident à détecter les valeurs aberrantes ou erronées, et à comprendre les variables de l’ensemble de données et les relations entre elles.

## Description générale du jeu de données et valeurs manquantes

Le jeu de données regroupe des commentaires de client écrits à propos de produits de prêt-à-porter. Les données ont auparavant été rendues anonymes. Le jeu de données se compose de 23 486 lignes (ou observations) et de 10 colonnes, soit une variable cible (ou « target ») et 9 variables explicatives (ou « features »). Chaque observation correspond à un avis laissé par un client à propos d’un produit de prêt-à-porter.

Les cinq premières lignes du jeu de données sont affichées ci-dessous.

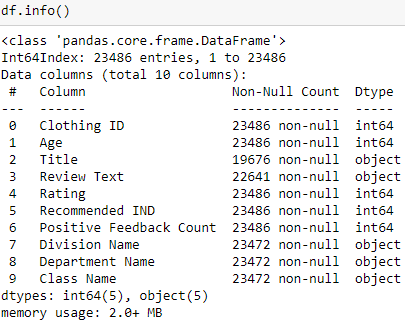


Le tableau suivant récapitule les informations pour chacune des variables.

|  |  |
| --- | --- |
| Variable | Description |
| Clothing ID | Référence du produit critiqué. |
| Age | Age du critique (ou « reviewer »). |
| Title | Titre de la critique |
| Review Text | Texte de la critique. |
| Rating | Note du produit accordée par le client, de 1 « Pire », à 5 « Meilleur ». |
| Recommended IND | Indique si le client recommande le produit (1) ou s’il ne le recommande pas (0). |
| Positive Feedback Count | Nombre de clients qui ont trouvé l’avis positif. |
| Division Name | Nom de la plus haute division du produit. |
| Department Name | Nom du département du produit. |
| Class Name | Nom de la classe du produit. |

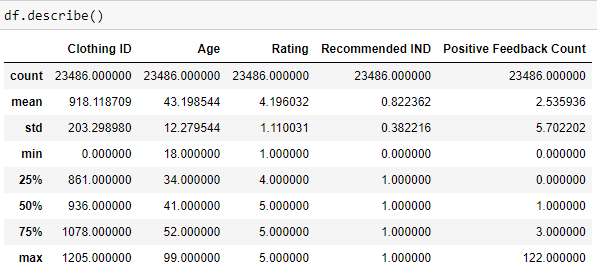
La variable cible se nomme « Recommended IND ». Elle est de type booléen. Contrairement à « Rating », cette variable permet de classer les produits selon deux sentiments, positif ou négatif. Pour prédire si un produit est recommandé ou non, le modèle se base sur la variable textuelle « Review Text ».

La fonction pandas.core.frame.DataFrame.info() donne des informations sur, entres autres, la structure du DataFrame, le nom des colonnes, le nombre de valeurs non-nulles, le type de données.



Le résultat de cette fonction montre qu’il existe des valeurs manquantes pour les variables « Title », « Review Text », « Division Name », « Department Name » et « Class Name ». Il y a 5 variables qualitatives et 5 variables quantitatives.

La fonction pandas.core.frame.DataFrame.describe() permet d’obtenir des statistiques descriptives pour chacune des variables du jeu de données : le nombre d’observations, la moyenne, l’écart-type, la valeur minimale, le premier quartile, la médiane, le troisième quartile et la valeur maximale.



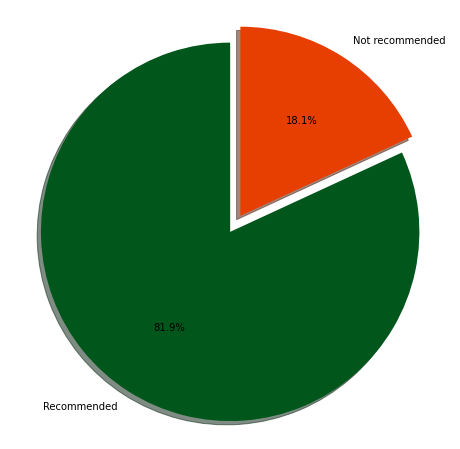
Ce tableau renseigne également sur la distribution des variables, tout comme les boîtes à moustaches (ou « boxplot ») en annexe XXX. Ces derniers montrent également les données aberrantes (ou « outliers ») qui sont représentés par des points de valeurs supérieures à l’extrémité supérieure (3e quartile multiplié par 1,5 fois l’espace interquartile).

## Observations dupliquées

L’utilisation du code a permis de mettre en évidence l’existence d’un unique doublon dans le jeu de données.



## Déséquilibre de la répartition des classes dans la variable cible



La répartition des observations dans les deux classes de la variable cible est déséquilibrée : il y a près de quatre fois plus de produits recommandés (18540) que de produits non recommandés (4101) dans le jeu de données. Ce déséquilibre est représenté par le graphique ci-dessous, créé avec la bibliothèque matplotlib.

## Analyse univariée et multivariée des variables numériques et catégorielles

Excepté pour l’âge, aucune variable ne semble distribuée normalement. Le « pair plot » (Annexe XX) ne met pas en évidence de corrélation entre la variable cible et d’autres variables, sauf concernant la variable « Rating ».

La fréquence cumulée des produits recommandés et non recommandés répartie par note montre que plus la note est élevée, plus il y a de produits commentés (Annexe XXX). Globalement les produits non recommandées sont répartis dans les notes de 1 et 2, les produits recommandés dans les notes 4 et 5. La note 3 est attribuée pour des vêtements autant recommandés que non recommandés.

Le graphique en Annexe XXX montre que les robes sont les produits les plus commentés (6145), suivi par les jupes (4626) et les blouses (2983). Il ne semble pas y avoir de type de vêtement particulièrement recommandé ou non recommandé.

L’hypothèse formulée avec l’analyse des graphiques précédents est confirmée avec la matrice de corrélation (Annexe XXX) : il existe une corrélation positive de 0,79 entre la note attribuée par le client et la recommandation. Il n’existe aucune autre corrélation entre d’autres variables.

## Analyse des données textuelles

Pour également comprendre sur quels éléments le modèle va se baser pour prédire un sentiment positif ou négatif, une exploration des données textuelles est également nécessaire. Cette analyse détermine aussi quel traitement doit être appliqué aux données textuelles afin d’optimiser les performances du modèles.

Il y a plusieurs traitements couramment utilisés en TAL :

* Retirer les « stop-words ». Les stop-words sont un ensemble de mots couramment utilisés dans toute langue. Par exemple, "the", "is" et "and" sont des stop-words en anglais. Supprimer les stop-words élimine les mots sans importance pour le modèle pour qu’il se concentre sur ceux qui ont une signification.
* Racinisation des mots (ou « stemming »). Le mot est tronqué de toute déclinaison, accord (flexions) et dérivation. Par exemple la forme racinisé de « jouons » est « jou ».
* Lemmatisation des mots. Le mot est ramené à un terme de forme la plus simple, quels que soient ses accords, déclinaisons, etc. Par exemple la forme lemmatisée de « jouons » est « jouer ».

Contrairement à la racinisation, un mot lemmatisé est transformé en un mot existant. Cette forme du mot dépend de son contexte dans la phrase. Par conséquent, ce traitement est très sensible aux fautes de langue.

.

|  |
| --- |
| C:\Users\Admin\Documents\marianneSimplon\simplon\sentiment_analysis\Rapport_E1\figures\eda\text\frequence_mot_pos.png |
| C:\Users\Admin\Documents\marianneSimplon\simplon\sentiment_analysis\Rapport_E1\figures\eda\text\frequence_mot_neg.png |

Après avoir retiré les stop-words, les graphiques représentant les mots les plus fréquemment utilisés dans le corpus de texte sont créés. Dans les vingt-cinq premiers mots les plus fréquemment utilisés, il existe des mots spécifiques pour chacun des sentiments. En effet, « great », « perfect », « little », « buy », « well » et « soft » sont plutôt utilisés pour décrire des produits recommandés. Les mots « return », « material », « much », « shirt » et « try » sont employés pour des commentaires négatifs. Le mot « love » n’est pas spécifique au sentiment positif. Cela pourrait s’expliquer par la présence de commentaires mitigés dans le jeu de données.

Le tableau suivant compare les statistiques descriptives de différents traitements de données. Les valeurs de comparaison sont :

* word\_count : nombre total de mot dans un commentaire.
* character\_count : nombre total de caractères (excepté les espaces) dans un commentaire.

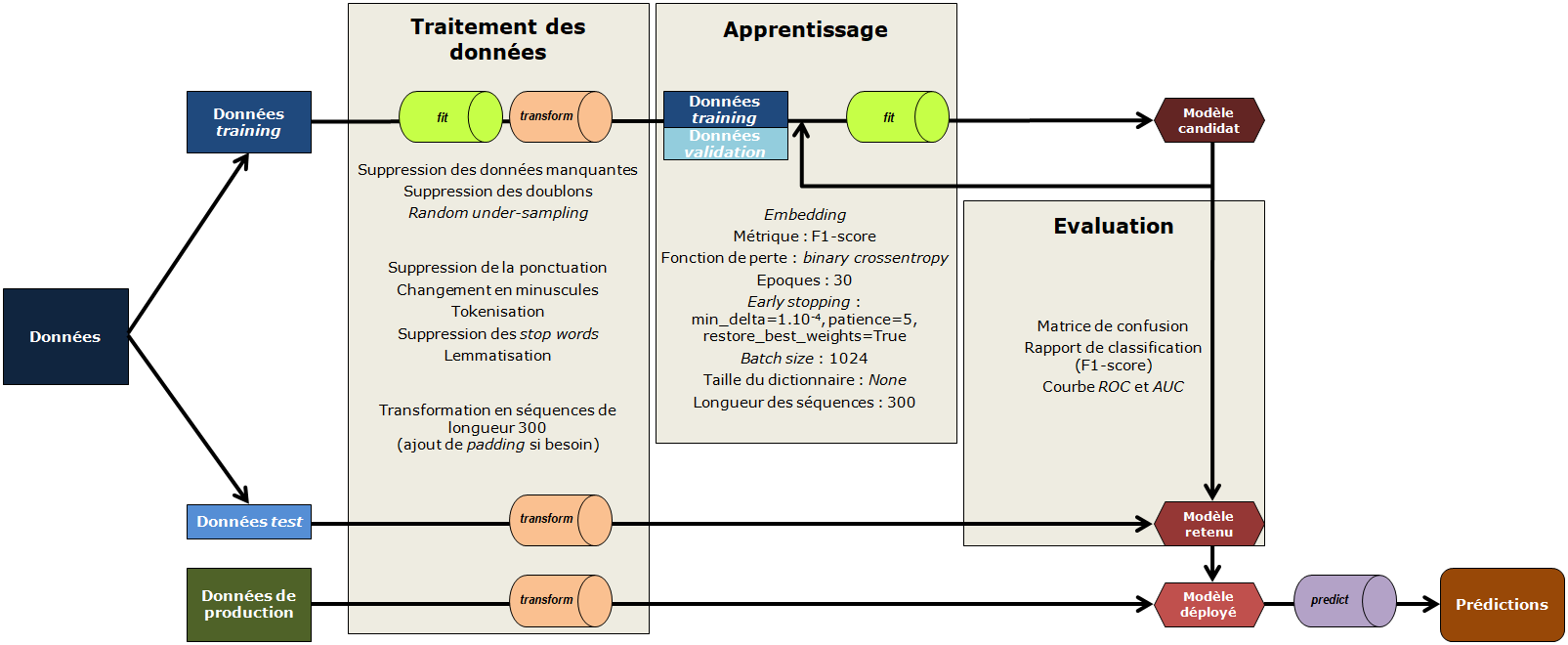
|  |  |  |  |
| --- | --- | --- | --- |
|  | Pas de traitement  (Tokenisé) | Retrait des  stop-words | Lemmatisé |
| Négatif | C:\Users\Admin\Documents\marianneSimplon\simplon\sentiment_analysis\Rapport_E1\figures\eda\text\stats_base_neg.PNG | C:\Users\Admin\Documents\marianneSimplon\simplon\sentiment_analysis\Rapport_E1\figures\eda\text\stats_stopwords_neg.PNG | C:\Users\Admin\Documents\marianneSimplon\simplon\sentiment_analysis\Rapport_E1\figures\eda\text\stats_lemmatized_neg.PNG |
| Positif | C:\Users\Admin\Documents\marianneSimplon\simplon\sentiment_analysis\Rapport_E1\figures\eda\text\stats_base_pos.PNG | C:\Users\Admin\Documents\marianneSimplon\simplon\sentiment_analysis\Rapport_E1\figures\eda\text\stats_stopwords_pos.PNG | C:\Users\Admin\Documents\marianneSimplon\simplon\sentiment_analysis\Rapport_E1\figures\eda\text\stats_lemmatized_pos.PNG |

Globalement, les calculs ne montrent pas de différence marquée entre les avis positifs et négatifs (différences inférieures aux écart-types). Quels que soient les traitements appliqués, les moyennes du nombre de mots et le nombre de caractères par commentaires est plus élevé dans les avis négatifs que positifs.

Le ratio « upper-case » - « lower-case » est globalement équivalent entre les commentaires négatifs (4,04.10-3) et positifs (4,02.10-3). La moyenne du nombre total de symboles de ponctuation dans un avis positif (10, 12) est aussi est comparable à celle des avis négatifs (10,10) (voir les annexes XXX et XXX pour les statistiques complètes).

# Traitement des données

Seule la variable « Review Text » est conservée pour prédire la variable cible « Recommended IND ». Le traitement des données est effectué selon les conclusions déduites de l’exploration de données. L’extrait de schéma ci-dessous représente les transformations effectuées sur les données textuelles avant apprentissage du modèle.

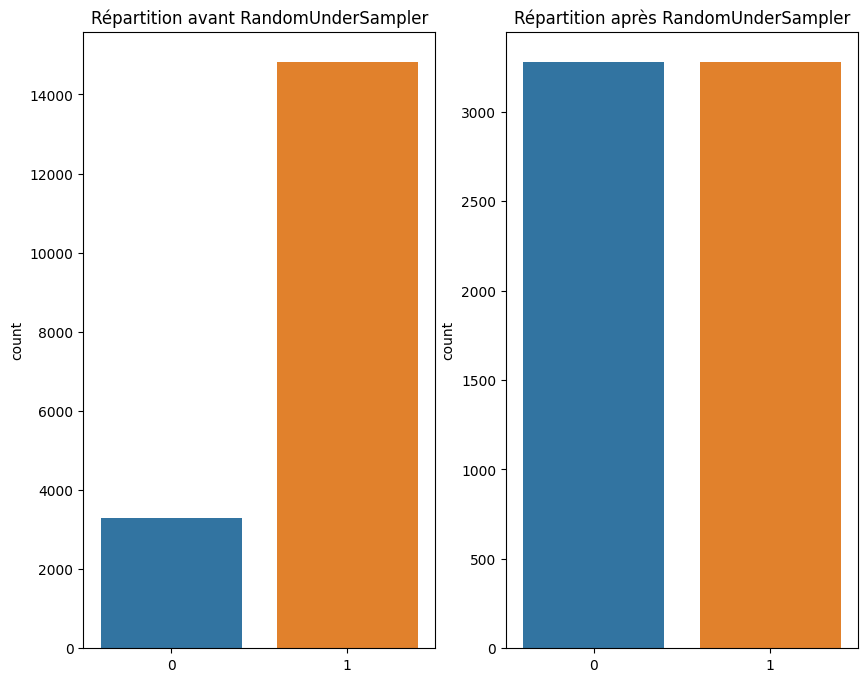


Les données manquantes et dupliquées sont supprimées. Les données textuelles sont changées en minuscules et la ponctuation est supprimée. Elles sont ensuite tokenisées, les stop-words sont retirés et les mots restants sont lemmatisés (Annexe XXX). Ainsi avant d’être lemmatisés, le rôle de chacun des mots est évalué, et le mot est transformé selon le contexte (nom, verbe, adjectif ou adverbe).

Voici ci-dessous un extrait du jeu de données final.



Comme vu précédemment, la répartition des données dans les classes est déséquilibrée. Pour corriger ce problème, un traitement de sous-échantillonnage avec imblearn.under\_sampling.RandomUnderSampler() est appliqué aux données. Les arguments de cette fonction sont définis de sorte que celle-ci mélange le jeu de données et sélectionne aléatoirement autant d’observations surnuméraires que le nombre de données de la classe minoritaire.



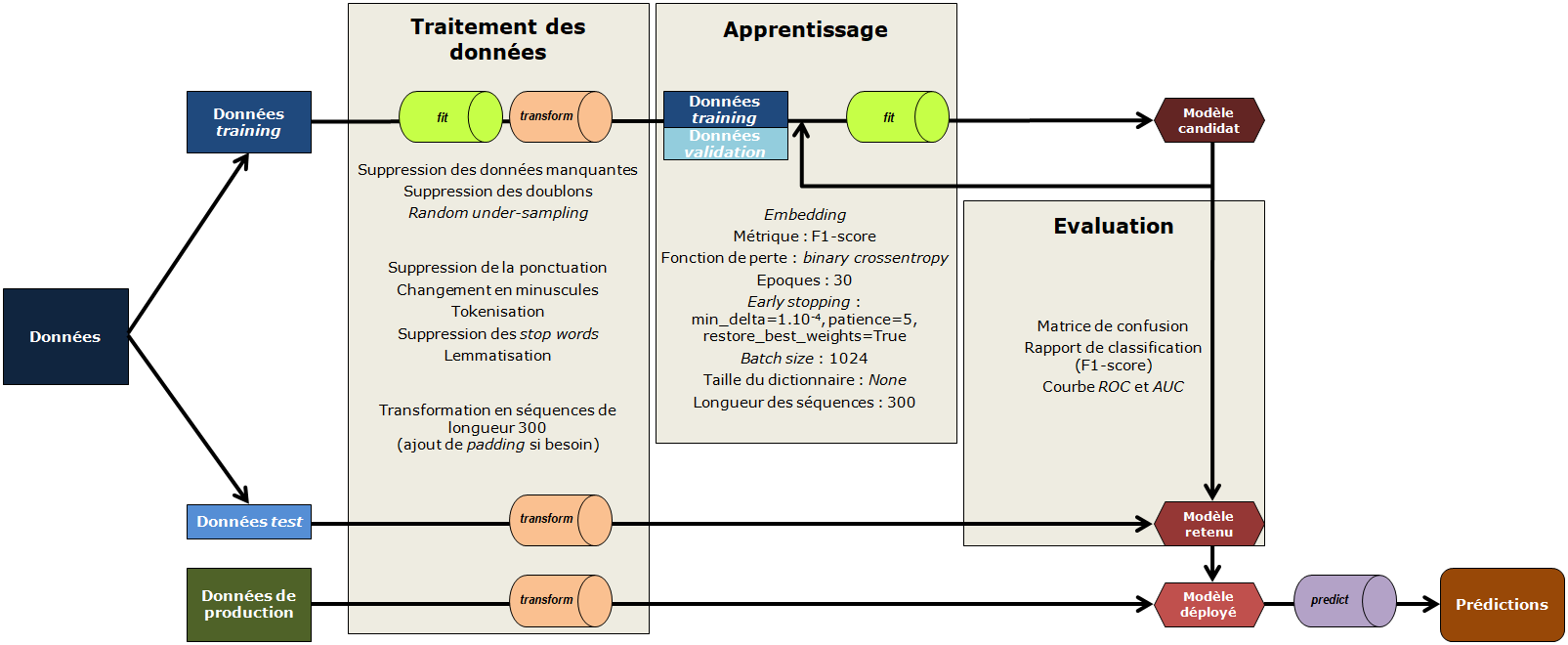
Un imblearn.over\_sampling.SMOTE() a aussi été testé mais cette solution n’a pas été retenue. En effet, les performances du modèle se sont révélées meilleures avec RandomUnderSampler().

Dans le domaine du TALN, les données doivent subir encore une transformation avant de devenir des entrées pour un modèle : l’embedding. Plusieurs méthodes existent : Word2vec, Glove, etc. La technique retenue est celle de l’embedding grâce à une couche « Embedding » de Tensorflow. Il s’agit de la méthode la plus simple à mettre en place car elle ne fait pas appel à un autre modèle.

Avant d’appliquer les transformations aux données, le jeu de données est divisé en deux parties. La première, regroupant la majorité des observations, devient le jeu de données d’entraînement. Ce sont sur celles-ci que les entrainements vont être pratiqués, tant pour le traitement que pour la modélisation. La deuxième partie correspond au jeu de données de test, qui regroupe les observations qui servent à déterminer les performances du modèle sélectionné. La séparation du jeu de données est réalisée avec la fonction sklearn.model\_selection.train\_test\_split(). Les arguments utilisés pour cette fonction déterminent la « seed » utilisée pour la répartition aléatoire des données dans chacun des jeux de données, tout en préservant la balance des classes de la variable cible. Le jeu de données de test est de taille égale à 20% du jeu de données global. Aucun entraînement n’est pratiqué sur cette fraction du jeu de données.

Ainsi, les transformations sont ajustées aux données d’entraînement puis appliquées aux données d’entrainement et de test. Les valeurs de la variable cible sont conservées sous leur forme brute.

# Modélisation

Une fois les données modifiées et conformes au format d’entrées de modèle, il faut déterminer quels modèles utiliser et les entraîner sur les données d’entraînement. Le modèle retenu prédit le sentiment positif ou négatif d’un commentaire placé en entrée. Le schéma ci-dessous illustre la méthode employée pour effectuer la modélisation et l’évaluation du modèle ainsi créé.

## Choix des modèles algorithmiques d’analyse de sentiments

L’analyse de sentiments est un problème de classification de texte. Historiquement, ce genre de problème est traité en transformant les données textuelles en vecteurs, puis en utilisant un modèle de classification comme Naïve Bayes, Random Forest ou XGBoost. Comme expliqué dans le rapport de veille (rapport E3), aujourd’hui les modèles de deep learning qui sont couramment utilisés en TALN sont de types Réseau de Neurones Récurrents (ou « Recurrent Neural Network », « RNN », en anglais), et plus récemment de type Transformer. Ces architectures ont un système de « mémoire », permettant de prendre en compte le contexte d’un mot dans une phrase.

En accord avec cette veille, un modèle BERT (Bidirectional Encoder Representations from Transformers) a d’abord été envisagé pour notre problématique. Ce modèle est utilisé comme base dans le cadre d’un transfert d’apprentissage (ou transfer learning). Cependant ce type d’analyse est très coûteux en termes de temps de calculs.

La solution des modèles de RNN a donc été retenue. Compte tenu du temps imparti, des modèles LSTM et BiLSTM ont été testés avec la bibliothèque Tensorflow. Cette librairie est l’une des plus populaires pour traiter des problématiques de deep learning.

## Entraînement et évaluation du modèle

### Entraînement du modèle

Différentes architectures du réseau de neurones sont testées en variant : le nombre de neurones, le nombre de couches et le nombre de couches « Dropout ». La taille du dictionnaire n’est pas limitée et la longueur des séquences en entrée du réseau de neurones est fixée à 300. L’optimizer (ou optimiseur) utilisé est Adam et la loss function (ou fonction de perte/coût) employée est celle de l’ binary crossentropy (ou entropie croisée binaire). Ce sont les options couramment utilisées pour un problème de classification binaire. La métrique sélectionnée est le F1-score. En effet, même s’il a été corrigé il s’agit de la métrique préférée lorsqu’il y a un déséquilibre dans le jeu de données.

Lors de l’entraînement pour la modélisation, le jeu de données d’entraînement est une nouvelle fois subdivisé pour obtenir un jeu de données de validation (ici 30% du jeu de données d’entraînement). Cette fraction du jeu de données sert à ajuster les paramètres du modèle afin de minimiser l’erreur lors de l’apprentissage.

Un hyperparamètre a été testé : la taille des lots (ou batch size). Le nombre d’epochs (ou époques) est défini à 30 et un dernier élément a été ajouté pour limiter l’overfitting (ou surajustement) : un early stopping (ou arrêt précoce). Voici les paramètres modifiés de l’early stopping :

|  |  |  |  |
| --- | --- | --- | --- |
| **Paramètre** | **min\_delta** | **patience** | **restore\_best\_weights** |
| **Valeur** | 1.10-4 | 5 | True |

### Evaluation du modèle

Le modèle est évalué avec les outils couramment utilisés pour les problèmes de classification :

* La matrice de confusion
* Le rapport de classification
* L’AUC (« Area Under the Curve »)
* La courbe ROC (« Receiving Operator Characteristic »)

La courbe d’apprentissage (ou learning curve) est également tracée pour évaluer un potentiel underfitting (sous-ajustement) ou overfitting du modèle aux données d’entraînement.

## Performance des modèles testés et versioning du modèle

### Performance des modèles testés

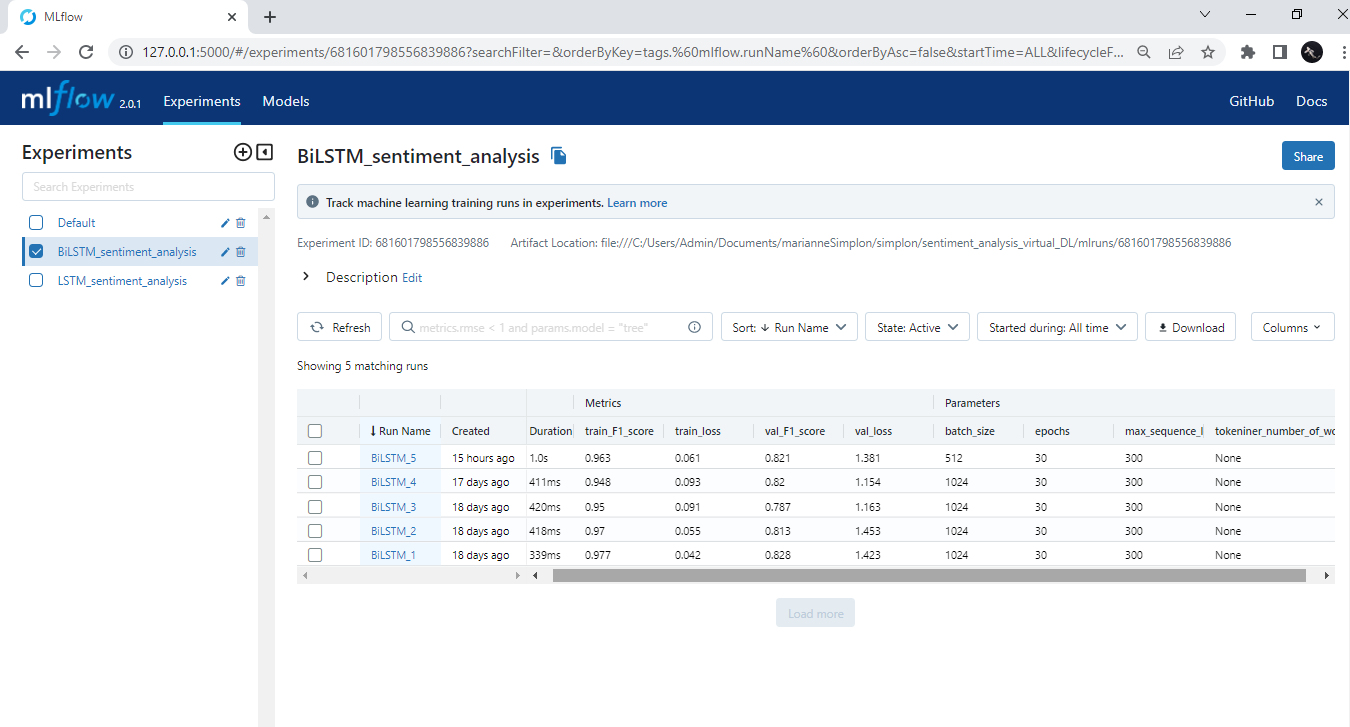
Le tableau ci-dessous récapitule les modèles testés et leur score sur les données de validation.

|  |  |  |
| --- | --- | --- |
| Architecture | Batch size | F1-score |
| 1 couche LSTM de 30 neurones  1 couche LSTM de 20 neurones  1 couche LSTM de 10 neurones | 1024 | 0.748 |
| 1 couche LSTM de 10 neurones | 1024 | 0.815 |
| 1 couche LSTM de 100 neurones | 1024 | 0.819 |
| 1 couche BiLSTM de 60 neurones  1 couche BiLSTM de 40 neurones  1 couche BiLSTM de 20 neurones | 1024 | 0.787 |
| 1 couche BiLSTM de 50 neurones  1 couche Dropout de 0.3  1 couche BiLSTM de 50 neurones | 1024 | 0.813 |
| 1 couche BiLSTM de 100 neurones  1 couche Dropout de 0.3 | 1024 | 0.82 |
| 1 couche BiLSTM de 100 neurones | 512 | 0.821 |
| 1 couche BiLSTM de 100 neurones | **1024** | **0.828** |

Le modèle retenu est donc composé d’une couche BiLSTM de 100 neurones. Il présente en effet le meilleur F1-score sur les données de validation (soit 0.828).

### Versioning du modèle avec mlflow

Le suivi des versions des modèles est effectué avec la bibliothèque mlflow. Celui-ci m’a permis d’enregistrer toutes les architectures que j’ai testées associées à leur score.

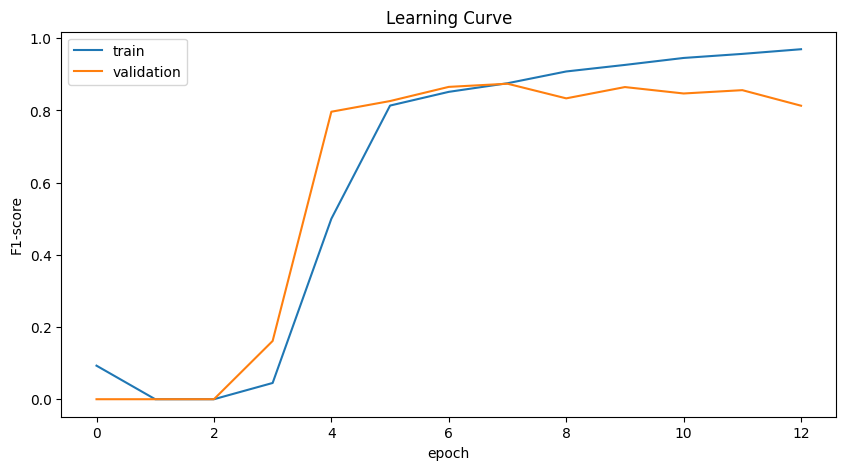


## Evaluation du modèle retenu

Le tableau ci-dessous récapitule toutes les caractéristiques du modèle retenu.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Architecture | Batch Size | Epoch | Early stopping | Taille du dictionnaire | Taille des séquences en entrée | Optimizer | Loss function | Métrique | Taille de validation |
| 1 couche BiLSTM de 100 neurones | 1024 | 12 (30 initialement) | min\_delta=1.10-4, patience=5,  restore\_best\_weights=True | None | 300 | Adam | binary crossentropy | F1-score | 30% |

### Learning curve



L’entraînement du modèle s’est arrêté à la douzième époque sous l’action du early stopping. Avec une patience de 5, le modèle enregistre les meilleurs poids et biais à la septième époque. Le biais entre les courbes de « train » et de « validation » augmente mais pas significativement pour suspecter de l’overfitting. En faisant abstraction des performances de calcul limitantes de mon ordinateur, il serait intéressant de supprimer le early stopping afin d’observer le comportement du modèle sur plus d’époques.

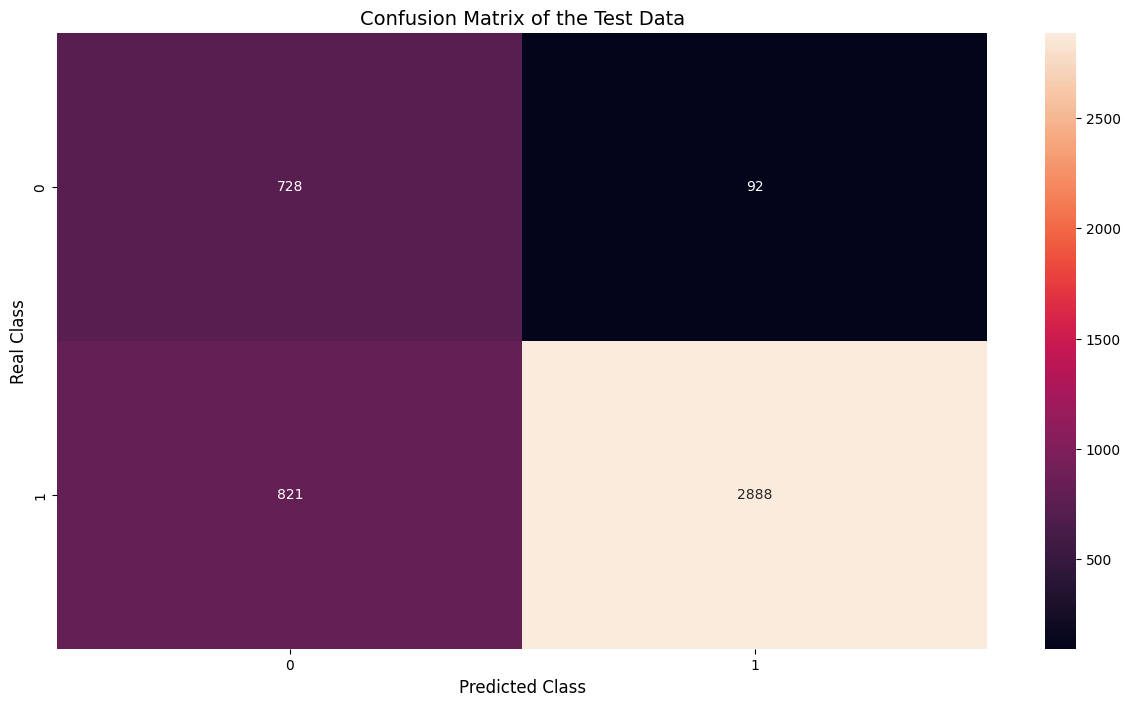
### Rapport de classification, matrice de confusion, AUC et courbe ROC

La matrice de confusion est un des outils d’évaluation d’un modèle de classification.

Les résultats d'une matrice de confusion sont classés en quatre grandes catégories : les vrais positifs, les vrais négatifs, les faux positifs et les faux négatifs :

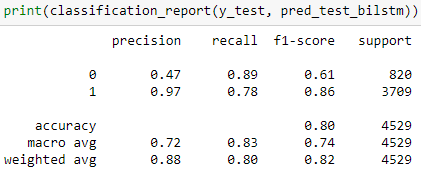
* Les vrais positifs correspondent aux prédictions positives pour des valeurs réelles effectivement positives.
* Les vrais négatifs correspondent aux prédictions négatives pour des valeurs réelles effectivement négatives
* Les faux positifs sont les prédictions positives mais pour des valeurs réelles négatives.
* Les faux négatifs sont les prédictions négatives mais pour des valeurs réelles positives.

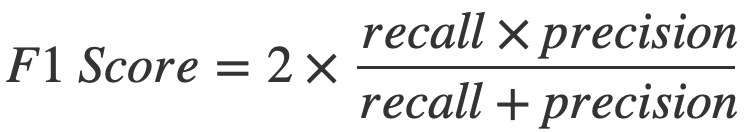
Celle du modèle retenu appliqué aux données de test est la suivante :



Sur la matrice de confusion, la diagonale principale sont les prédictions correctement classées par le modèle. Ainsi, 728 commentaires se sentiment négatif et 2888 commentaires positifs ont été parfaitement classés. En résumé, le modèle a bien prédit sur 3616 des 4529 données, soit près de 80%. Mais 821 avis négatifs sont prédits comme positifs et 92 avis positifs sont rédist comme négatifs.

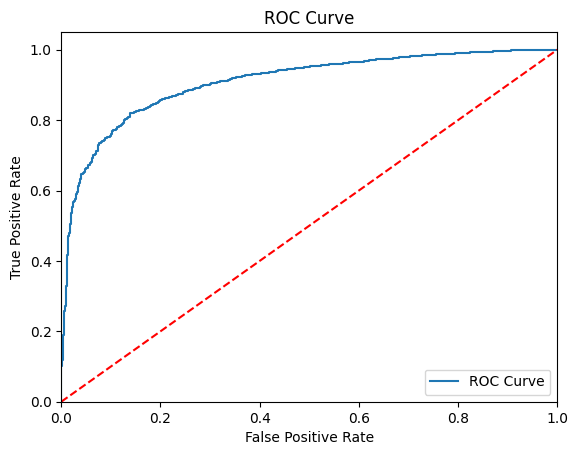
Le rapport de classification pour le modèle retenu est visible ci-dessous.



* L’accuracy est la part des observations prédites correctement sur total d’observations. Dans notre cas, celle-ci est de 80%, ce qui est satisfaisant pour notre probématique d’analyse de sentiment.
* La précision représente les vrais positifs prédits sur le total des observations prédites comme positives. Plus elle est élevée, plus le modèle minimise le nombre de faux positifs. Ainsi, 97% des sentiments positifs prédits sont effectivement positifs, mais seulement 47% des sentiments négatifs prédits sont réellement négatifs.
* Le recall (ou rappel) est la part des vrais positifs sur toutes les observations réellement positives (vrais positifs et faux négatifs). Plus il est élevé, plus le modèle maximise le nombre de vrais positifs. Ainsi, 89% des sentiments négatifs ont été classés correctement par le modèle, 78% pour les sentiments positifs.
* Le F1-score est une sorte de moyenne de la précision et du recall (moyenne harmonique), dont voici la formule :

Il permet de se faire une idée de la performance du modèle en prenant compte à la fois de la précision et du recall. Contrairement aux autres scores, le F1-score est moins sensible aux déséquilibres de jeu de données. Plus le F1-score est élevé, le plus le modèle est performant. Ainsi, 0.86 est satisfaisant, mais 0.63 est un peu faible.

Une courbe ROC a également été tracée pour notre modèle.



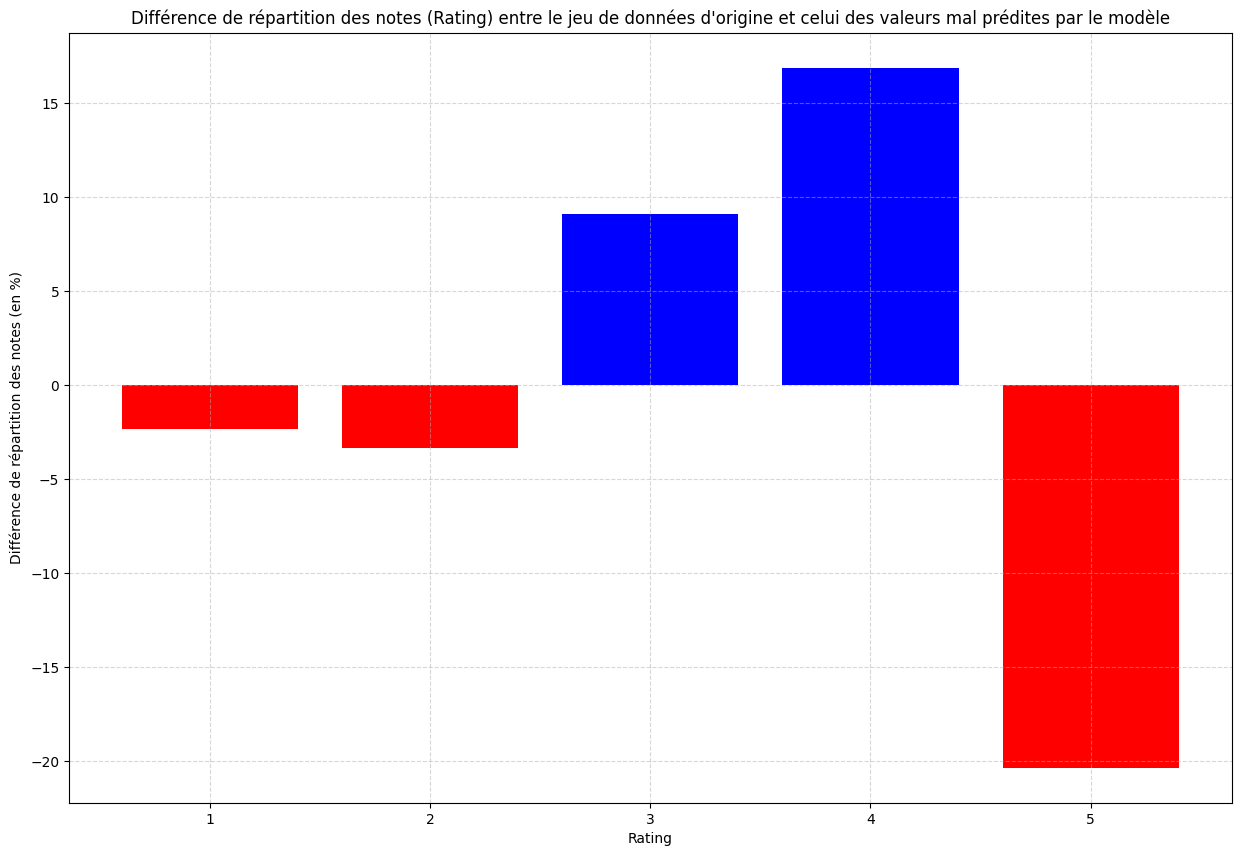
Cette courbe trace le taux de vrais positifs par rapport au taux de faux positifs. L’aire sous la courbe est une façon de mesurer la performance d’un modèle de classification. Un modèle parfait aurait une valeur égale à 1, tandis qu’un classement aléatoire aurait pour valeur 0,5, soit le trait en rouge sur le graphique. Ici, elle est égale à 0.907 environ ce qui est très satisfaisant, notre modèle est donc un bon classificateur d’avis positifs.

### Interprétation du modèle

Après examen des observations correspondant aux prédictions erronées du modèle, deux principales hypothèses expliquant ces erreurs sont envisagées.

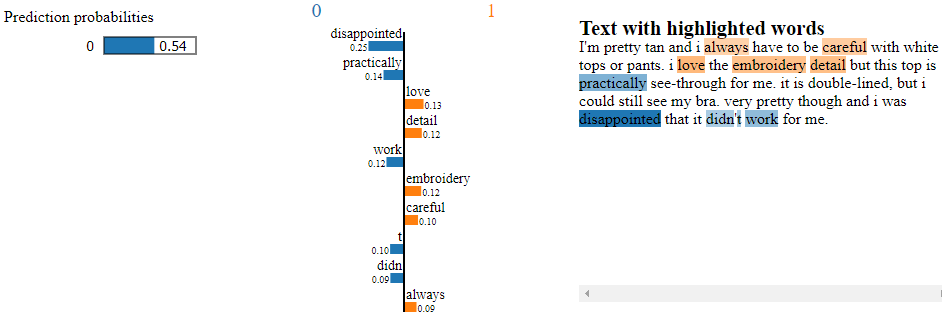
#### Les avis nuancés

Le modèle semble se tromper de recommandation pour les commentaires nuancés. Tout d’abord, les notes de ces commentaires montrent des fréquences différentes de celles originelles.

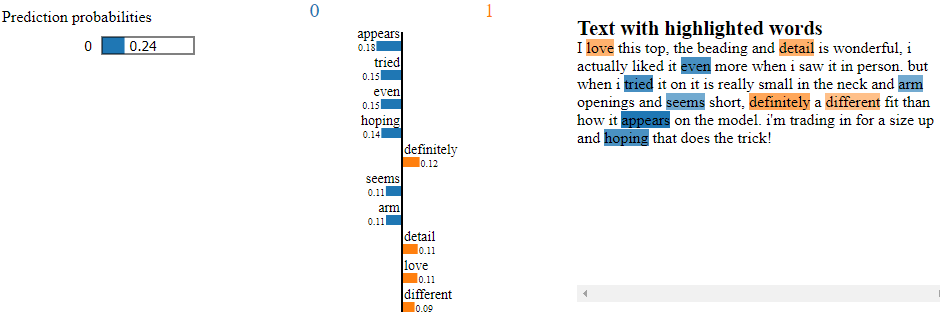


Les avis contrastés sont moins fréquents dans le jeu des données mal prédites que celui d’origine. En effet, pour les notes 1, 2 et 5, il y a une baisse de fréquence d’environ 2, 3 et 20% respectivement. A l’inverse, les commentaires nuancés sont plus fréquents. Les notes 3 augmentent de 9% et celles égales à 4 de 17%.

Une analyse complémentaire est réalisée avec LIME (Local Interpretable Model-agnostic Explanations) qui aide à comprendre quels mots dans un texte en entrée ont le plus d’influence sur la prédiction finale du modèle. Voici ci-dessous deux exemples représentatifs d’entrées que le modèle a mal prédites.



Dans ce premier exemple, le modèle a prédit que le client a recommandé le produit avec un score de 0.54. Dans cet avis, il y a autant de termes jugés positifs que négatif. Ce commentaire est nuancé, le client parle du produit autant positivement que négativement. La note attribuée par le client est de 3. Mais son véritable avis est dispensé synthétiquement en fin de commentaire, avec le mot « disappointed ». Ce mot est jugé très important par LIME, mais pas suffisamment pour contrebalancer les mots positifs.

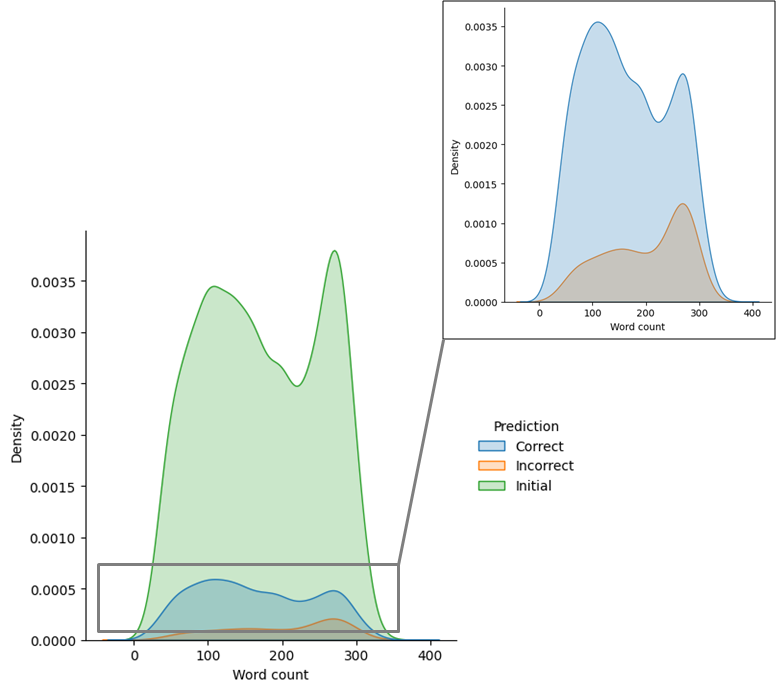


Dans ce deuxième exemple, le modèle a estimé que le vêtement n’est pas recommandé avec un score de 0.24. Il s’agit également d’un avis nuancé, malgré une note 4.

Enfin LIME a mis en évidence que le modèle retenu ne sait pas interpréter le sarcasme.



#### Les avis longs

Le modèle semble également prédire des recommandations erronées pour les avis longs, c’est-à-dire avec beaucoup de mots.

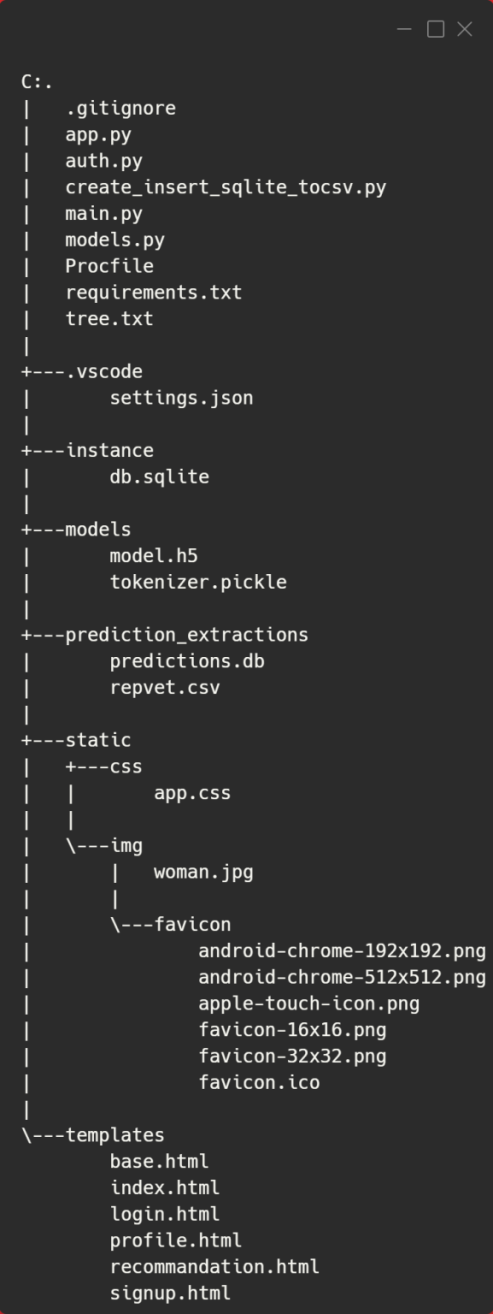
La comparaison de la distribution des entrées mal prédites avec les distributions des données correctement estimées et l’ensemble du jeu de données originel semble confirmer cette hypothèse. Elle montre qu’il y a relativement plus de commentaires longs (plus de 230 mots) incorrectement estimés par le modèle. En parallèle, plus de données correctement prédites sont entre des longueurs d’avis de 50 à 200 mots.

# 10. Application et déploiement

## 10.1. Application

### 10.1.1. Arborescence de l’application

L’arborescence choisie pour l’application est celle communément utilisée pour les applications Flask (voir figure ci-dessous).

A la racine, il y a six dossiers :

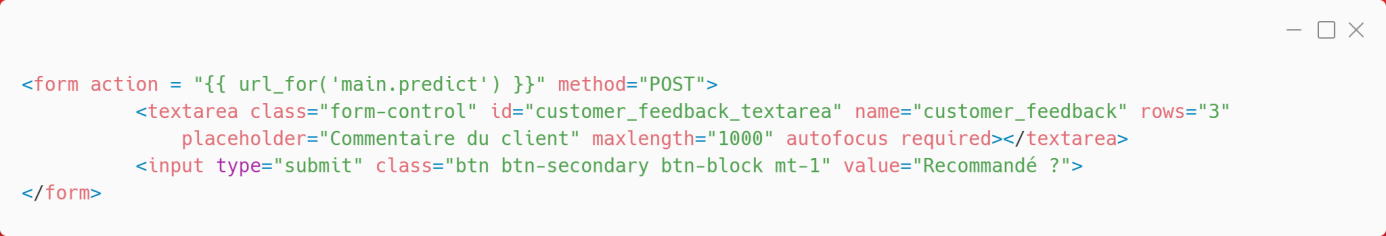
* .vscode : contient les paramètres de Visual Studio Code pour l’application au format json. Il permet notamment d’associer le dossier de l’application avec l’App Service de Azure.
* instance : contient la base de données sqlite des utilisateurs de l’application
* models : comprend les pickles du modèle et du tokenizer.
* prediction\_extractions : regroupe la base de données contenant les prédictions de l’application, ainsi que le fichier CSV pour l’enregistrement de ces commentaires dans la base de données MySQL Workbench.
* static : inclut les fichiers CSS et les images de l’application
* templates : comporte les templates de l’application. A la différence d’un simple fichier HTML, un template affiche un contenu selon les conditions vérifiées ou non. Un template de base (base.html) contient les éléments statiques du site. Selon la page à afficher, le contenu du body ou de la barre de navigation sont ajustés.

Il y a également huit fichiers à la racine du projet :

* .gitignore : fichier qui permet d’indiquer à Git quels éléments du projet doivent être ignorés.
* les fichiers au format Python (.py) : il s’agit des fichiers back-end de l’application. Ils créent l’instance Flask, traitent les requêtes web entrantes et répondent à celles-ci via un système de routage, etc.
* Procfile : spécifie les commandes qui sont exécutées par l'application au démarrage, il déclare notamment le serveur web de l’application (gunicorn).
* requirement.txt : liste toutes les librairies python du projet et indique quelles versions sont à installer.

### 10.1.2. Vérification des entrées du formulaire recueillant l’avis du client

Il existe plusieurs niveaux de vérification des entrées d’un utilisateur dans l’application. Du côté du front-end, des vérifications sont réalisées en HTML avec l’attribut « required ».

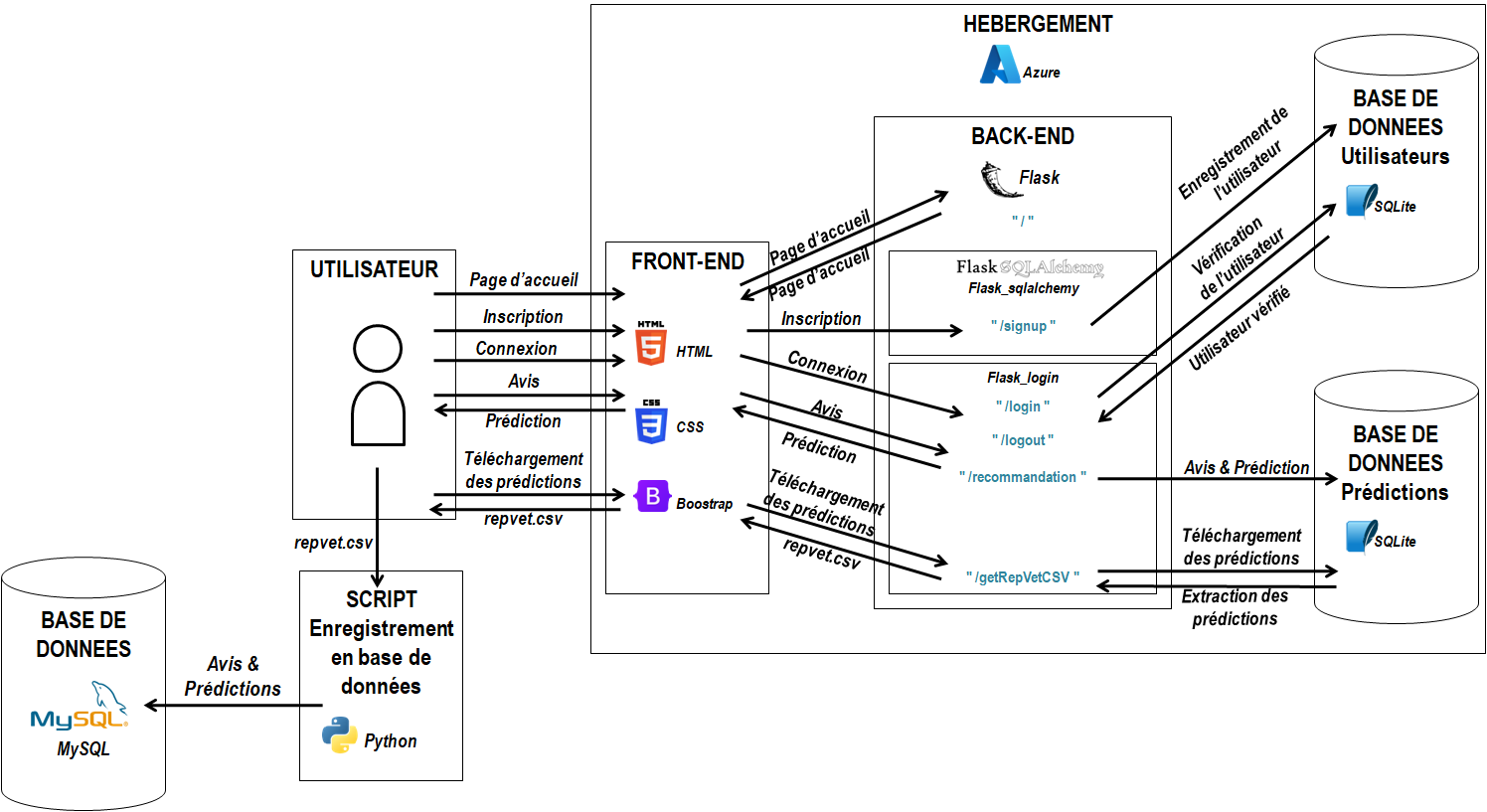


Cet attribut oblige l’utilisateur à entrer une valeur dans le champ « textarea » du formulaire pour qu’il puisse être soumis. Un message apparaît si ces éléments ne sont pas respectés.

En back-end, la prédiction et l’enregistrement de celle-ci en base de données sont exécutés uniquement si la variable « request.form[‘customer\_feedback’] » existe, autrement dit, si le champ n’est pas vide.



### 10.1.3. Fonctionnement de l’application



L’application offre plusieurs fonctionnalités à l’utilisateur :

* L’affichage de la page d’accueil. Tout utilisateur - qu’il soit non inscrit, non authentifié ou authentifié - a accès à la page d’accueil qui correspond à la route « / » de l’application. Cette dernière renvoie uniquement le template index.html.
* L’authentification de l’utilisateur. Dans un premier temps l’utilisateur s’inscrit. La fonction définit dans la route « /signup » crée un utilisateur dans la base de données Utilisateurs grâce à Flask\_sqlalchemy. Ensuite un utilisateur inscrit peut se connecter et se déconnecter. Flask\_login fournit les fonctions login\_user() et logout\_user() utilisées dans les routes « /login » et « /logout ». Lorsque l’utilisateur tente de se connecter, ses informations d’authentification sont comparées à celles existantes dans la base de données. L’utilisateur est connecté si ces informations correspondent à une entrée de la base de données.
* La soumission d’un avis pour prédiction du sentiment. Conformément au diagramme UML, l’utilisateur doit être authentifié pour utiliser cette fonctionnalité. Cela est vérifié avec le « decorator » nommé @login\_required. L’utilisateur remplit le formulaire d’un avis client. Après vérification des champs et soumission du formulaire, l’application affiche la valeur prédite par le modèle. Dans un même temps, l’avis et la prédiction associée sont enregistrés en base de données.
* Le téléchargement des prédictions au format CSV pour enregistrement en base de données MySQL. Via la route « /getRepVetCSV », l’utilisateur télécharge repvet.csv qui correspond à une extraction de la base de données des prédictions. L’utilisateur peut ensuite enregistrer les avis et les prédictions dans la base de données MySQL en exécutant le script python load\_csv\_to\_mysqlworkbench.py.

## 10.2. Tests unitaires

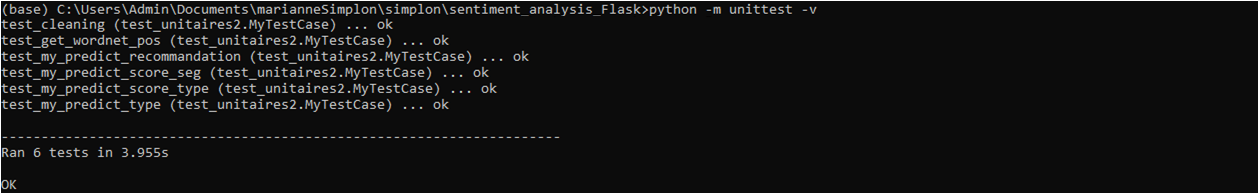
Des tests unitaires sont codés avec la librairie unittest. Ils vérifient la conformité :

* Du rôle du mot lemmatisé renvoyé à partir de celui identifié par pos\_tag.
* Du type de la variable retournée après traitement textuel du commentaire de l’utilisateur (retirer la ponctuation, mettre le texte en minuscules, tokeniser, supprimer les « stop-words », lemmatiser).
* Du type de la variable retournée par la fonction de prédiction.
* Du segment du score retourné par la fonction de prédiction.
* Du type du score retourné par la fonction de prédiction.
* De la valeur de la recommandation retournée par la fonction de prédiction.

Ci-dessous un extrait du code des tests unitaires réalisés.



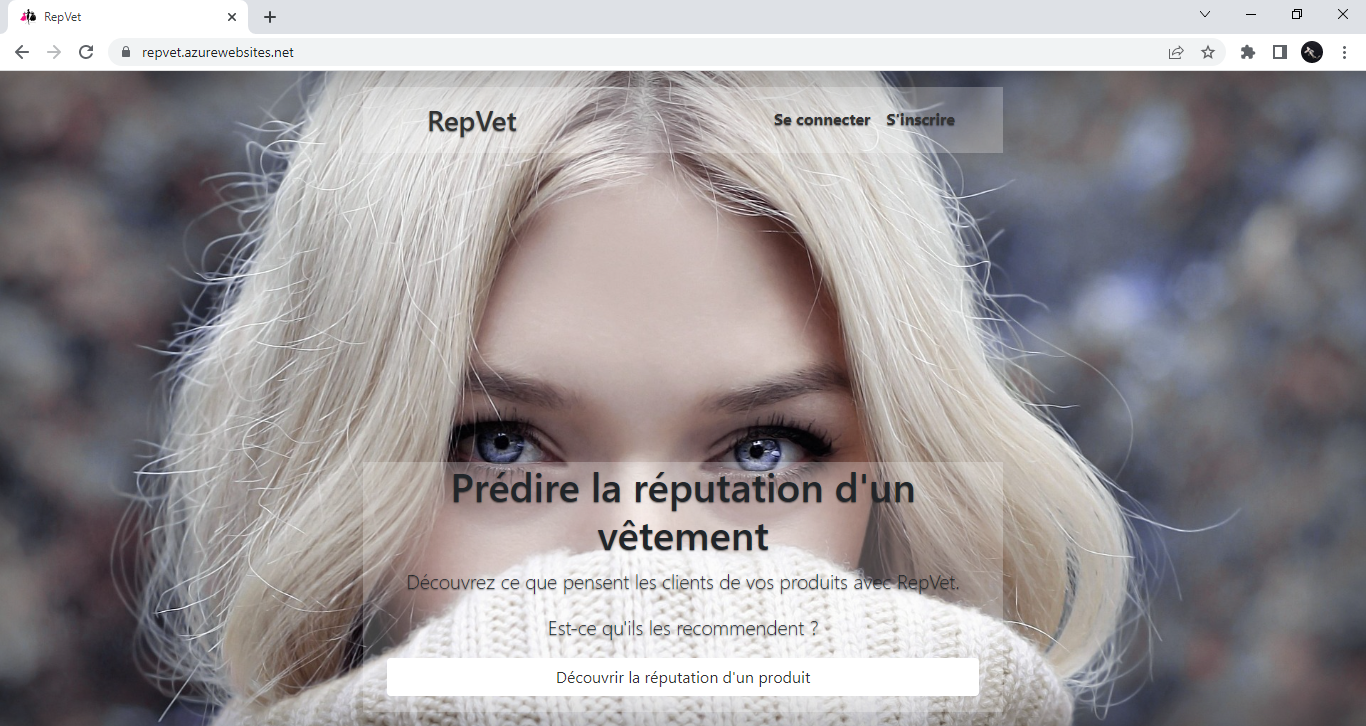
Après exécution du fichier python codant les tests unitaires (Annexe XXX), les tests ont été passés avec succès.



## 10.3. Déploiement

Comme indiqué précédemment, le déploiement a été exécuté grâce à Azure et Visual Studio code. Azure App Service est un service d’hébergement web prenant en charge les applications python sur un serveur Linux. Pour cela il faut posséder un compte Azure, un groupe de ressources et un abonnement. Il faut également choisir une grille tarifaire. Compte tenu du poids des bibliothèques utilisées avec un modèle de deep learning, le niveau tarifaire de Base (B1) est choisi. Une fois l’Azure App Service créé, l’accès à l’application peut également se faire via le Portail Azure, qui facilite la gestion de l’application en centralisant tous les service

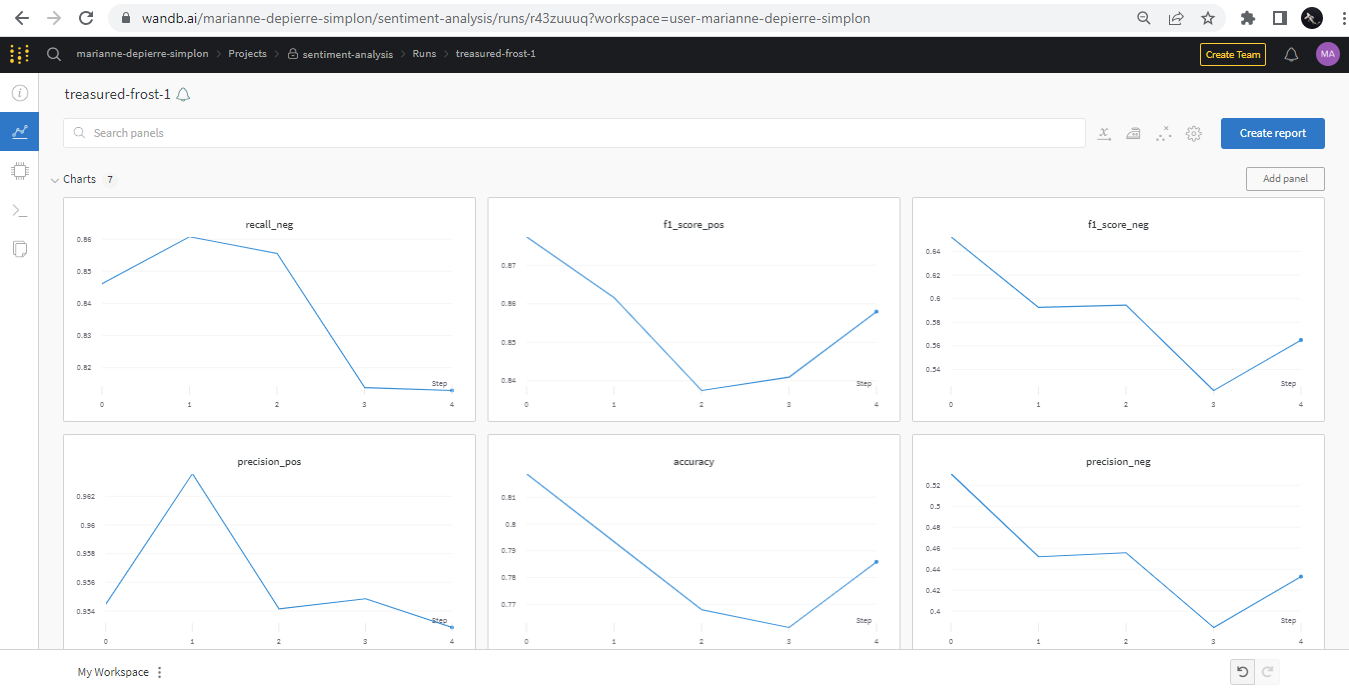
Dès lors que l’application est déployée, l’accès est possible avec un navigateur web à l’URL <https://repvet.azurewebsites.net> (URL « http » également disponible).



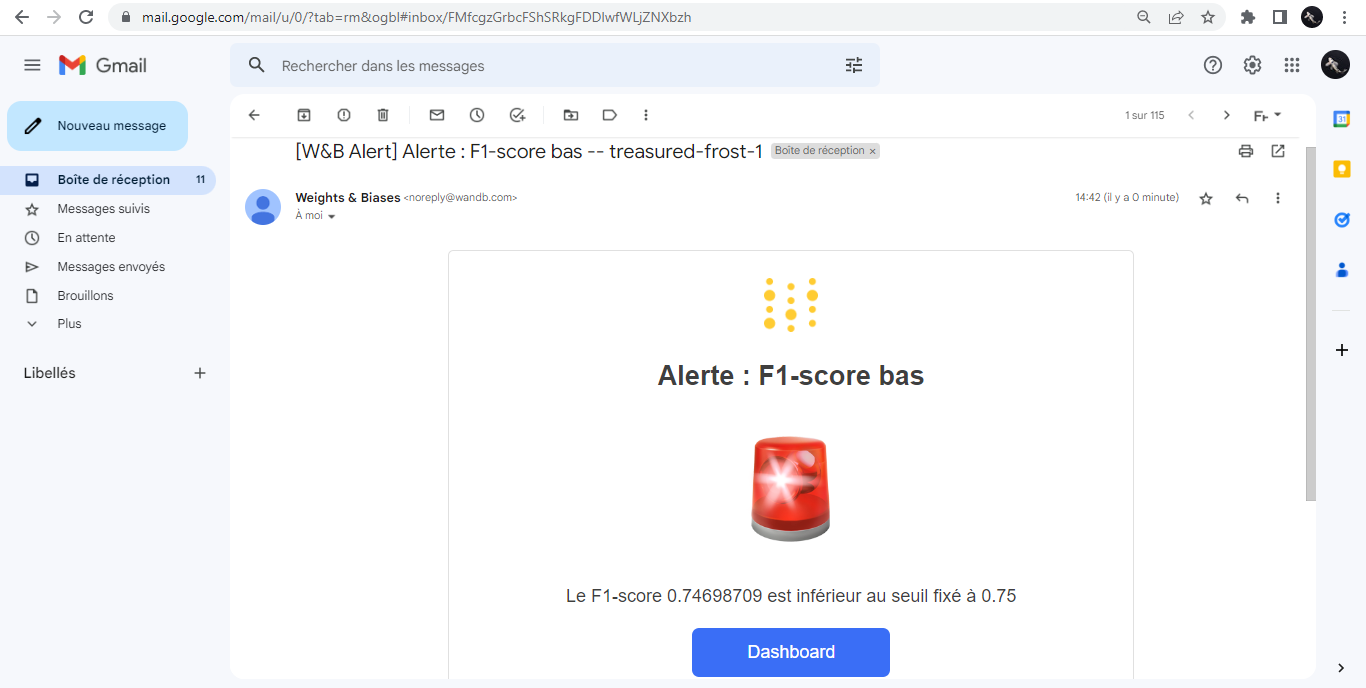
|  |
| --- |
|  |

# Monitoring du modèle avec WandB

Un suivi du modèle (ou « monitoring ») est effectué avec WandB. Un script python (Annexe XXXX) enregistre les valeurs des métriques de l’application lorsqu’il est exécuté. WandB dispose d’outils de visualisation permettant de suivre l’évolution des performances du modèle.



Lorsque ce script est exécuté, une partie du code vérifie que la valeur du f1-score n’est pas inférieure à un seuil (Annexe XXXX). En effet, si celle-ci est en-dessous de 0.75, une alerte est envoyée automatiquement par mail à l’adresse e-mail renseignée dans le code.



# Bilan

Les besoins exprimés par le client sont satisfaits. En effet, un modèle de deep learning a été créé. Celui-ci permet de détecter un sentiment positif ou négatif selon le commentaire écrit par le client. Ce modèle a été déployé dans une application web où un formulaire permet de recueillir l’avis d’un client pour effectuer une prédiction. Celle-ci est affichée de façon claire. Une fonction de l’application et l’exécution d’un script python permettent d’enregistrer les prédictions dans la base de données. De plus, le monitoring du modèle en place est effectué en exécutant un script python, avec un système d’alerting prévenant d’une révision nécessaire du modèle.

Plusieurs analyses n’ont cependant pas pu être tentées afin de tenir les délais imposés par le projet. Néanmoins, elles seraient intéressantes à explorer afin de déterminer si elles amélioreraient ou d’optimiseraient le modèle. Voici quelques possibilités :

* Tester d’autres modèles, tels que le GRU.
* Tester les performances du modèle avec d’autres méthodes d’embedding, tels que Word2Vec ou GloVe.
* Développer une fonction qui prédirait le sentiment pour chaque avis client d’un jeu de données soumis.

Ce Proof of Concept a néanmoins montré qu’il est possible de réaliser avec un modèle de TAL pour des problématiques d’analyse de sentiments d’avis d’utilisateurs. Il servira ainsi de base de discussion pour la mise en place d’un suivi des retours d’utilisateurs des chatbots des différents services de Sanofi.

# Annexes