



# LIVRET D'APPRENTISSAGE

FOLLACO Ludovic

**Promotion Berners Lee 2025**

## **Certification RNCP38616**

- ➔ Bloc 03 : Appliquer des techniques d'analyse IA via des algorithmes d'apprentissage automatiques.
- ➔ Bloc 05 : Développer et mettre en production des algorithmes d'IA par apprentissage profond.

## Avant de commencer ....

Bienvenue,

Voici le *template* pour l'élaboration de votre Livret d'apprentissage.

Ne changez pas sa mise en forme, afin que les examinateurs concentrent leur attention sur vos réponses.

Respectez la taille du texte, et l'espace disponible.

Savoir être précis et concis est la clé !

*Daniel Villa Monteiro*

# CAS PRATIQUE

## Epreuve 1

 Semaine 5


### Modalités d'évaluation :

Sur la base d'un cas d'entreprise réel ou fictif, le/la candidat(e) doit identifier les enjeux/problématiques rencontrées par l'entreprise. Il/elle doit traduire les enjeux du client en objectifs réalisables. A partir de ces objectifs, il/elle doit programmer, entraîner et utiliser un modèle d'apprentissage profond.

 **Cette première épreuve du Livret d'apprentissage permet d'évaluer les compétences suivantes :**

➔ Bloc 3 : Appliquer des techniques d'analyse IA via des algorithmes d'apprentissage automatique

 C1 : **Sélectionner l'algorithme d'apprentissage le plus adapté** en comparant les performances et les caractéristiques des différentes familles d'algorithmes afin d'apporter une réponse pertinente à la problématique métier rencontrée.

 C2 : **Préparer et transformer des données** en utilisant des techniques de prétraitement (preprocessing) pour les adapter aux spécificités du modèle d'apprentissage automatique choisi.

 C3 : **Entraîner un modèle d'apprentissage automatique** en optimisant une loss function (fonction de coût) à partir des données d'entraînement afin de permettre à l'algorithme d'effectuer le moins d'erreurs possibles selon des indicateurs de succès clairement définis.

**Analyse du besoin :**

L'objectif de mon projet est simple :

- Prédire automatiquement le prix d'un service à partir de critères clairs et facilement exploitables.

L'essor des plateformes comme Fiverr, Malt ou Upwork a considérablement augmenté le volume d'offres disponibles en ligne. Les écarts sont importants en termes de :

- Prix,
- Qualité,
- Positionnement.

Ces variations sont visibles même au sein d'une même catégorie, ce qui rend difficile le positionnement tarifaire d'une nouvelle offre sur le marché.

Dans ce contexte, le projet vise à fournir une estimation :

- Rapide,
- Cohérente,
- Directement exploitable au moment de la création d'une offre.

Les critères retenus pour la prédiction sont :

- La description textuelle du service,
- Le taux de fiabilité du vendeur, calculé à partir de la note moyenne pondérée par le nombre d'évaluations

L'objectif est double :

- Proposer un prix aligné avec les tendances du marché,
- Accélérer la prise de décision, sans passer par une analyse manuelle fastidieuse.

Cette première phase s'appuie sur des modèles classiques de régression supervisée, entraînés à partir de données historiques. C'est une approche simple à mettre en œuvre, robuste, et facilement intégrable dans un outil métier.

L'enjeu est clair :

- Gagner en efficacité opérationnelle,
- Améliorer la cohérence commerciale des services proposés.

## Etat de l'art :

La prédiction de variables numériques à partir de texte et de données simples, telles que des valeurs numériques ou des catégories, est un cas courant en apprentissage automatique. Ce type de problématique peut être traité efficacement avec des modèles de régression supervisée.

Les modèles les plus utilisés sont :

- La régression linéaire : simple à mettre en œuvre, mais limitée lorsque les relations entre les variables ne sont pas linéaires,
- Les modèles ensemblistes comme les arbres de décision, le Random Forest ou le Gradient Boosting, qui offrent de meilleures performances sur des données tabulaires ou hétérogènes.

Dans mon projet, les données comprennent une description textuelle. Pour transformer ce texte en variables exploitables par un modèle de régression, plusieurs techniques sont possibles :

- Le TF-IDF (Term Frequency – Inverse Document Frequency), une méthode simple et robuste qui pondère les mots selon leur fréquence dans un document et dans l'ensemble du corpus,
- Les Embeddings (non utilisés ici), plus puissants mais nécessitant un traitement plus complexe, davantage adaptés au Deep Learning.

Les outils utilisés sont standards en Machine Learning :

- Scikit-learn, pour les algorithmes de base (régressions, arbres, pipelines, validation croisée),
- XGBoost, reconnu pour ses performances et sa robustesse sur les petits et moyens jeux de données.

Ces méthodes permettent d'obtenir des résultats satisfaisants tout en restant simples à mettre en œuvre et à déployer dans une application.

## Choix technique :

Deux approches ont été mises en place dans le cadre du projet :

- Une régression supervisée, pour prédire un prix exact,
- Une classification supervisée, pour déterminer une tranche tarifaire (Basse, Moyenne, Haute).

Ces deux modèles partagent les mêmes données d'entrée :

- La description du service, vectorisée par TF-IDF,
- Le taux de fiabilité, intégré comme variable numérique.

### ➤ Régression

Plusieurs modèles ont été testés et comparés selon trois métriques : MAE, RMSE et  $R^2$ .

| Modèle            | MAE  | RMSE | $R^2$   |
|-------------------|------|------|---------|
| Gradient Boosting | 3.21 | 4.90 | 0.2566  |
| XGBoost           | 3.33 | 5.00 | 0.2274  |
| Random Forest     | 3.32 | 5.03 | 0.2173  |
| Ridge             | 3.82 | 5.47 | 0.0748  |
| KNN Regressor     | 3.99 | 5.80 | -0.0407 |
| Decision Tree     | 4.43 | 7.14 | -0.5770 |
| Linear Regression | 5.86 | 9.90 | -2.0353 |

Le Gradient Boosting obtenu les résultats les plus satisfaisants mais a également été retenu pour :

- Sa bonne performance globale,
- Sa capacité à gérer les non-linéarités,
- Sa stabilité sur des jeux de données de taille moyenne.

### ➤ Classification

Plusieurs modèles ont également été testés pour la classification des tranches de prix. Le Decision Tree Classifier est celui qui a obtenu les meilleurs résultats :

| Modèle              | Accuracy |
|---------------------|----------|
| Decision Tree       | 0.6175   |
| Random Forest       | 0.5339   |
| KNN Classifier      | 0.5179   |
| Logistic Regression | 0.5100   |

Il a été retenu pour sa simplicité, sa lisibilité, et sa capacité à capturer des relations non linéaires dans les données.

➤ **Pipeline technique**

Chaque modèle est intégré dans un pipeline scikit-learn composé de :

- Une vectorisation TF-IDF de la description,
- Une intégration directe des variables numériques,
- Une évaluation croisée (cross-validation),
- Une optimisation des hyperparamètres via GridSearchCV si nécessaire.

Les modèles finaux et leurs préprocesseurs sont sauvegardés avec joblib pour être utilisés dans l'application.

**Mise en oeuvre technique :**

La mise en œuvre s'est déroulée en plusieurs étapes structurées.

➤ **Chargement et préparation des données**

Le dataset utilisé provient de la plateforme Fiverr. Les données ont été nettoyées et harmonisées :

- Suppression ou traitement des valeurs manquantes,
- Homogénéisation des catégories,
- Conversion de la note moyenne et du nombre d'évaluations en un taux de fiabilité.

➤ **Vectorisation de la description**

La colonne texte ("description du service") a été vectorisée avec la méthode TF-IDF. Ce traitement a été encapsulé dans un pipeline scikit-learn, ce qui permet, d'enchaîner les étapes automatiquement et de limiter les erreurs.

➤ **Modélisation (régression et classification)**

Deux modèles ont été entraînés indépendamment :

- Un Gradient Boosting Regressor pour prédire un prix exact,
- Un Decision Tree Classifier pour prédire une tranche tarifaire.

Dans chaque cas :

- Les données ont été divisées en train/test,
- Les modèles ont été évalués par validation croisée,
- Les performances ont été mesurées à l'aide de métriques adaptées (MAE, RMSE,  $R^2$  pour la régression / Accuracy pour la classification).

➤ **Sauvegarde des modèles**

Les modèles finaux et leurs pipelines (TF-IDF + prédicteur) ont été sauvegardés avec joblib dans des fichiers \*.pkl, afin de pouvoir les réutiliser facilement dans l'interface Gradio.



### ➤ Tests de prédiction

Des tests ont été réalisés sur de nouveaux échantillons, issus de Fiverr ou simulés, afin de vérifier que les prédictions étaient cohérentes, même en dehors du jeu d'entraînement.

Cette mise en œuvre structurée garantit la reproductibilité, la portabilité et l'intégrabilité des modèles dans une application utilisateur simple.

## Epreuve 1 - 3ème Partie

### Bilan :

Le projet a permis de démontrer la faisabilité d'un système capable de prédire le prix d'un service freelance à partir d'une description et d'un taux de fiabilité.

Les résultats obtenus sont satisfaisants au regard de la simplicité du modèle et de la taille du jeu de données.

Le modèle de régression Gradient Boosting affiche un MAE de 3.21 et un RMSE de 4.90, ce qui reste cohérent avec les écarts observés dans les données d'origine. Du côté de la classification, le Decision Tree Classifier atteint une accuracy de 61,75 %, ce qui le rend utilisable dans un contexte d'aide à la décision.

Les modèles sont rapides à entraîner, simples à interpréter, et faciles à intégrer dans une interface.

Ils constituent une base fiable pour fournir une estimation tarifaire initiale, utile dans le cadre d'une veille concurrentielle ou d'une stratégie de positionnement commercial.

Ce premier résultat valide le choix de la séparation ML/DL dans le projet, avec un socle classique pour des cas simples, et une extension deep learning pour aller plus loin dans la logique de traitement du texte.

## Amélioration :

Plusieurs pistes d'amélioration ont été identifiées pour renforcer la précision et la robustesse des modèles :

### ➤ **Enrichir le jeu de données**

L'un des leviers principaux serait d'augmenter la taille du dataset, en récupérant plus d'offres réelles depuis Fiverr ou d'autres plateformes. Cela permettrait de mieux capter la diversité des services et d'améliorer la généralisation du modèle.

### ➤ **Affiner les variables utilisées**

Il serait pertinent d'exploiter d'autres caractéristiques disponibles, comme :

- La catégorie de service (ex. : design, rédaction, développement),
- Le nombre de jours de livraison,
- Ou encore le nombre de services inclus dans l'offre.

### ➤ **Améliorer le traitement du texte**

L'utilisation de techniques d'embeddings (comme Word2Vec ou BERT) permettrait d'aller au-delà du TF-IDF en capturant le sens des mots et des phrases. Cela pourrait améliorer les performances, notamment pour la classification.

### ➤ **Optimisation avancée**

Une optimisation plus fine des hyperparamètres (avec Optuna ou BayesianSearchCV) pourrait améliorer légèrement les scores, tout en évitant le surapprentissage.

### ➤ **Gestion du déséquilibre des tranches**

En classification, les classes ne sont pas parfaitement équilibrées. Il serait possible de :

- Tester des pondérations de classes,
- Ou appliquer du suréchantillonnage (SMOTE) pour homogénéiser l'apprentissage.

Ces axes montrent qu'il est possible de faire évoluer les modèles sans repartir de zéro, et d'améliorer progressivement les performances, en fonction des données disponibles et des besoins métiers.

# PROJET FINAL

## Epreuve 2

 Semaine 13

### Modalités d'évaluation :


A partir d'un cas d'entreprise réelle ou fictive, le/la candidat(e) doit développer une application exploitable par un client final intégrant des solutions IA.


 **Cette seconde épreuve du Livret d'apprentissage permet d'évaluer les compétences suivantes :**

➡ Bloc 5 : Développer et mettre en production des algorithmes d'IA par apprentissage profond (Deep Learning)

 C1 : **Préparer des données non structurées** en les convertissant en données numériques et sous forme tabulaires pour servir de données d'entraînement à un algorithme d'apprentissage profond.

 C2 : **Sélectionner l'algorithme d'apprentissage profond le plus adapté** en comparant les performances et les caractéristiques des différentes familles d'algorithmes afin d'apporter une réponse pertinente adaptée à la problématique métier rencontrée.

 C3 : **Entraîner un modèle d'apprentissage profond** en optimisant une loss function (fonction de coût) à partir des données d'entraînement afin de permettre à l'algorithme d'effectuer le moins d'erreurs possibles selon des indicateurs de succès clairement définis.

 C4 : **Déployer efficacement un modèle d'apprentissage profond** en utilisant des outils et plateformes de production adaptés (MLOps), pour assurer une accessibilité et une performance optimale des prédictions de l'algorithme aux utilisateurs finaux.

***Compréhension du besoin client***

**Décrivez les besoins de votre client fictif, auquel votre projet tente de répondre :**

Le client souhaite disposer d'une application simple et accessible, capable de l'aider à estimer automatiquement le prix d'un service, à partir d'un formulaire.

L'objectif est de permettre à un utilisateur, de saisir les informations principales d'une offre (description, fiabilité, etc.) et d'obtenir immédiatement :

- Une estimation du prix du service (régression),
- Une tranche tarifaire indicative (classification).

L'enjeu est double :

- Gagner du temps lors de la création d'offres,
- Maintenir une cohérence tarifaire au sein de l'entreprise, en fonction des tendances observées sur le marché.

L'outil doit être facile à utiliser, rapide à exécuter, et exploitable sans connaissance en IA. Il doit également pouvoir être intégré dans un environnement de travail ou testé localement par différents profils métiers.

C'est dans ce contexte qu'a été conçue une interface Gradio, connectée à un modèle de Deep Learning, entraîné pour capter les éléments clés d'une offre et fournir une prédiction cohérente avec les données historiques.

## Epreuve 2 - 1ère Partie

### *Etat de l'art*

**Décrivez l'état de la concurrence et des recherches scientifiques quant à votre sujet d'application :**

Les modèles de Deep Learning sont particulièrement adaptés pour traiter des données non structurées, comme le texte, les images ou l'audio. Dans le cas de ce projet, la description du service est un champ textuel libre. Pour qu'un modèle puisse l'exploiter, il faut d'abord le transformer en vecteurs numériques.

#### ➤ **Embeddings textuels**

Les techniques d'embeddings permettent de représenter des phrases ou des documents sous forme de vecteurs denses, capables de capturer le sens des mots, leurs contextes, et leurs relations sémantiques.

Plusieurs approches existent :

- Word2Vec, GloVe, FastText (anciens, limités au niveau du mot),
- BERT et ses variantes, basés sur des transformers (meilleure compréhension du contexte global d'une phrase).

Pour ce projet, la méthode utilisée est issue de la librairie sentence-transformers, avec le modèle all-MiniLM-L6-v2, connu pour son bon compromis entre performance, vitesse et taille.

#### ➤ **Modèles combinés texte + données tabulaires**

Dans un contexte métier, il est fréquent que les prédictions reposent à la fois sur :

- Des données textuelles (ex. : description),
- Et des données structurées (ex. : fiabilité, niveau du vendeur).

Plusieurs architectures permettent de combiner ces deux types d'entrées dans un réseau de neurones :

- Un bloc dense dédié au texte vectorisé (embeddings),
- Un autre bloc dense pour les données numériques,
- Puis une concaténation suivie d'une ou plusieurs couches de sortie (régression ou classification).

Ce type de modèle permet une approche hybride, plus souple et souvent plus performante qu'un traitement uniquement textuel ou tabulaire.

➤ **Interface de démo avec Gradio**

Pour faciliter l'accès au modèle, une interface a été construite avec Gradio, une librairie Python permettant de générer rapidement :

- Un formulaire interactif,
- Une connexion directe au modèle,
- Un affichage clair du résultat.

Gradio est particulièrement adapté pour créer des interfaces de test, utilisables sans connaissances en programmation.

***Traduction et choix technique du projet***

**Expliquer l'ensemble de la stack technique que vous utilisez dans votre projet :**

Le modèle développé repose sur une approche combinée intégrant à la fois des embeddings textuels et des données numériques. L'architecture et les outils ont été choisis pour garantir un bon équilibre entre performance, simplicité et compatibilité avec une interface utilisateur.

➤ **Embedding de la description (texte)**

Le texte est transformé en vecteurs numériques grâce au modèle all-MiniLM-L6-v2\*\*, fourni par la librairie sentence-transformers. Ce modèle est pré-entraîné, léger et suffisamment performant pour des cas d'usage comme le nôtre.

Le vecteur résultant contient 384 dimensions, représentant le contenu sémantique complet de la description.

➤ **Données tabulaires**

Les autres entrées du modèle sont :

- Le taux de fiabilité, exprimé sous forme de score (note moyenne × volume d'évaluations),
- Le niveau du vendeur, encodé et intégré comme variable catégorielle (Note : dans l'application actuelle, cette variable est fixée à "Confirmé").

➤ **Architecture du modèle deep learning**

Le modèle est construit avec Keras (TensorFlow). Il comprend :

- Un bloc dense pour les embeddings textuels,
- Un second bloc dense pour les variables numériques,
- Une concaténation des deux,
- Une ou plusieurs couches cachées,
- Deux têtes de sortie (une pour la régression, l'autre pour la classification).

Cela permet d'obtenir :

- Un prix estimé (sortie régression),
- Une tranche de prix (Basse, Moyenne, Haute) (sortie classification).

➤ **Environnement technique**

- Python 3
- TensorFlow / Keras pour le modèle
- Sentence-transformers pour l'encodage du texte
- Joblib pour la gestion des scalers
- Gradio pour l'interface
- FastAPI pour le déploiement en API REST

Ce choix technique permet d'intégrer facilement le modèle dans une application de démonstration, tout en restant compatible avec une future mise en production (serveur, Docker, etc.).



## Epreuve 2 - 2ème Partie

### *Mise en oeuvre du projet*

**Expliquez les principales étapes de la mise en oeuvre de votre projet :**

Le projet a été mis en place en plusieurs étapes, de la préparation des données à l'intégration finale du modèle dans une interface interactive.

➤ **Préparation des données**

Le fichier source `fiverr_cleaned_transformed.csv` contient les descriptions, les niveaux de vendeurs, les notes moyennes et le nombre d'évaluations.

Le taux de fiabilité est calculé pour chaque ligne, puis les descriptions sont extraites et converties en vecteurs denses à l'aide du modèle `all-MiniLM-L6-v2`.

Les données tabulaires (fiabilité, niveau) sont encodées et normalisées via un scaler sauvegardé pour garantir la cohérence à l'inférence.

➤ **Construction du modèle Deep Learning**

Le modèle est construit avec TensorFlow / Keras, en deux parties :

- Un bloc dense appliqué aux Embeddings de description,
- Un autre bloc dense pour les données numériques (fiabilité + niveau),
- Une concaténation, suivie de couches intermédiaires,
- Deux sorties : une pour la prédiction du prix, l'autre pour la prédiction de la tranche.

Le modèle est entraîné en multi-sortie (`loss=[mae, categorical_crossentropy]`) sur un ensemble d'apprentissage issu du fichier préparé. Les performances sont surveillées à l'aide d'un set de validation.

➤ **Enregistrement du modèle et des composants**

Une fois entraîné, le modèle est sauvegardé au format `.h5`.

Le Scaler des données tabulaires est sauvegardé séparément avec Joblib.

L'ensemble peut être chargé au moment de la prédiction, sans nécessiter de retraitement manuel.

➤ **Intégration dans une interface utilisateur**

Une application Gradio a été développée pour tester le modèle dans un environnement simple et accessible :

- Champ libre pour la description,
- Curseur pour le taux de fiabilité,
- Bouton de prédiction (le niveau du vendeur est actuellement fixé à "Confirmé" dans le code).

L'utilisateur obtient instantanément :

- Un prix estimé en dollars,
- Une tranche de prix prévisible (Basse, Moyenne, Haute).

➤ **Déploiement API (optionnel)**

Une API REST a été construite avec FastAPI pour permettre une intégration dans d'autres outils ou interfaces.

Elle expose une route POST recevant les entrées et renvoyant les prédictions du modèle en JSON. Cette API est prête pour un futur déploiement avec Docker ou en cloud.

## Epreuve 2 - 3ème Partie

### *Bilan du projet*

#### Quel bilan tirez-vous de votre projet ? :

Le modèle de Deep Learning développé remplit pleinement les objectifs fixés en début de projet :

- Il est capable de prédire automatiquement le prix d'un service freelance,
- Il fournit également une estimation de la tranche tarifaire,
- Et il est intégré dans une interface simple, accessible sans compétences techniques.

Les résultats sont cohérents avec ceux obtenus en Machine Learning, tout en offrant une meilleure adaptabilité à des cas plus complexes, grâce à l'utilisation d'Embeddings textuels.

Le modèle fonctionne bien en local, avec un temps de réponse rapide, et permet de tester plusieurs scénarios de description et de fiabilité. L'approche combinée (texte + données tabulaires) apporte une meilleure robustesse face à la diversité des services.

Le fait d'avoir séparé les logiques ML (modèles classiques) et DL (modèle avancé) permet également de montrer la complémentarité des méthodes selon les cas d'usage. Ce découpage rend l'ensemble du projet modulable et pédagogique.

Enfin, la construction d'une API REST en parallèle ouvre la voie à un déploiement en conditions réelles, dans un environnement plus professionnel.

## Epreuve 2 - 3ème Partie

### *Axes d'améliorations*

**Si vous aviez 1 mois pour finaliser votre projet, quelles sont les fonctionnalités que vous ajouteriez ? :**

Plusieurs axes d'amélioration ont été identifiés pour renforcer la performance, la fiabilité et l'accessibilité du projet.

➤ **Déploiement distant**

Actuellement, l'application tourne en local. Elle pourrait être déployée sur un serveur distant avec :

- Un conteneur Docker pour l'environnement Python,
- Un hébergement cloud (Render, Heroku, OVH, etc.),
- Une supervision basique de l'API REST et de l'application Gradio.

➤ **Rendre le modèle plus interactif**

Le niveau du vendeur est aujourd'hui fixé en dur dans le code. Il pourrait être exposé dans le formulaire Gradio via une liste déroulante, pour permettre à l'utilisateur de tester différents profils.

➤ **Ajout d'une base de données**

Une base de données SQLite ou PostgreSQL pourrait être utilisée pour :

- Enregistrer les prédictions effectuées par les utilisateurs,
- Suivre l'usage de l'outil,
- Améliorer le modèle à terme via un mécanisme de feedback.

➤ **Affinage du modèle**

Des optimisations possibles incluent :

- Une meilleure normalisation des variables,
- Une augmentation du volume de données d'entraînement,
- Des tests croisés entre plusieurs architectures pour ajuster la complexité du modèle au volume disponible.

➤ **Intégration dans un processus métier réel**

Enfin, l'outil pourrait être intégré dans un véritable workflow commercial, par exemple pour assister un chargé de mission dans l'élaboration d'un devis. Cela nécessiterait un cahier des charges fonctionnel plus précis.

Ces améliorations sont toutes réalistes à court ou moyen terme, et permettraient de faire évoluer ce projet prototype vers une solution plus aboutie en environnement professionnel.

## **Epreuve 2 - Conclusion**

Ce projet m'a permis de mettre en pratique l'ensemble des compétences abordées pendant la formation, du traitement des données à la mise en production d'un modèle d'intelligence artificielle.

J'ai pu expérimenter :

- L'utilisation de modèles classiques de machine learning pour des cas simples,
- La construction d'un modèle de deep learning combinant texte et données tabulaires,
- L'intégration de ces modèles dans une interface utilisateur fonctionnelle,
- Ainsi que la création d'une API REST pour préparer un déploiement plus professionnel.

Ce travail m'a permis de mieux comprendre les enjeux liés à la mise en œuvre concrète d'un projet IA, aussi bien sur le plan technique que fonctionnel.

J'ai particulièrement apprécié la partie deep learning, qui m'a poussé à sortir de ma zone de confort et à structurer le code de manière propre et réutilisable.

Aujourd'hui, je me sens capable de proposer une solution IA complète, de l'analyse des besoins jusqu'à la livraison d'une interface testable.

Je n'ai aucun regret sur ce projet, seulement l'envie de continuer à progresser, notamment sur les sujets liés à l'industrialisation des modèles (MLOps, conteneurisation, monitoring...).