Projet STA211
Classification d'annonces publicitaires

Challenge de Data Science – CNAM Paris

Étudiant : Abdoullatuf Maoulida

Cours : STA211 – Entreposage et Fouille de Données

Année universitaire : 2024–2025

Email : Maoulida.abdoullatuf.auditeur@lecnam.net

Téléphone : 07.82.86.97.64

Jeu de données utilisé : Internet Advertisements (UCI Repository)

Objectif : Classification supervisée – Maximisation du score F1

1er août 2025

Table des matières

Résumé

Ce rapport présente l'approche adoptée pour résoudre le challenge de classification d'annonces publicitaires dans le cadre de l'UE STA211. Il décrit les étapes de prétraitement, d'analyse exploratoire, d'imputation, de sélection de variables, d'ingénierie de caractéristiques et de modélisation supervisée. L'objectif principal est de maximiser le **score F1** pour la classe minoritaire (publicité), en tenant compte du déséquilibre des classes et en s'appuyant sur des pipelines reproductibles et modulaires. Le projet a été réalisé individuellement.

1 Résumé Exécutif

Ce projet présente une approche complète de classification binaire pour détecter les publicités sur internet en utilisant le dataset *Internet Advertisements* de l'UCI Machine Learning Repository. L'objectif principal est de maximiser le F1-score sur un ensemble de test de 820 observations, en respectant les contraintes du challenge STA211.

Principales contributions :

— Pipeline de prétraitement robuste avec gestion des valeurs manquantes (KNN et MICE)
— Transformations optimales (Yeo-Johnson et Box-Cox) pour améliorer la normalité
— Modélisation avancée avec stacking sans refit combinant 5 algorithmes
— Optimisation du seuil de classification pour maximiser le F1-score
— Script de prédiction automatisé pour le challenge

Résultats obtenus :

— F1-score optimal : 0.923 sur l'ensemble de validation
— Modèle champion : Stacking sans refit KNN avec seuil 0.200
— Distribution des prédictions : 111 publicités (13.5%), 709 non-publicités (86.5%)
— Pipeline reproductible et documenté

2 Introduction

2.1 Contexte du Projet

Le challenge STA211 s'inscrit dans le cadre du cours de Méthodes Statistiques du CNAM. L'objectif est de développer un système de classification capable de distinguer les publicités des contenus non-publicitaires sur internet, en utilisant un ensemble de caractéristiques extraites des pages web.

Le dataset *Internet Advertisements* présente plusieurs défis techniques :

— Classes déséquilibrées (14% de publicités vs 86% de non-publicités)
— Valeurs manquantes dans les variables continues
— Variables binaires nombreuses (1557 features)
— Nécessité d'optimiser le F1-score plutôt que l'accuracy

2.2 Objectifs et Contraintes

Objectifs principaux :

1. Explorer et nettoyer les données de manière approfondie
2. Implémenter des méthodes d'imputation avancées
3. Appliquer des transformations pour améliorer la normalité
4. Développer des modèles de classification robustes
5. Optimiser les performances avec des techniques d'ensemble
6. Générer des prédictions conformes au format du challenge

Contraintes techniques :

— Format de soumission : fichier CSV avec 820 lignes
— Valeurs autorisées : ”ad.” ou ”noad.”
— Métrique d'évaluation : F1-score
— Environnement : Python avec bibliothèques standards

3 Exploration et Nettoyage des Données

3.1 Description du Dataset

Le dataset *Internet Advertisements* contient 3279 observations avec 1558 variables :

— **X1, X2, X3** : Variables continues (largeur, hauteur, ratio d'aspect)
— **X4** : Variable catégorielle (type de frame)
— **X5 à X1557** : Variables binaires (présence de mots-clés)
— **Target** : Variable cible binaire (ad./noad.)

Caractéristiques principales :

Variable	Type	Valeurs manquantes	Description
X1	Continue	28.1%	Largeur de l'image
X2	Continue	28.1%	Hauteur de l'image
X3	Continue	28.1%	Ratio d'aspect
X4	Catégorielle	0.1%	Type de frame
X5-X1557	Binaires	0%	Présence de mots-clés

TABLE 1 – Description des variables du dataset

3.2 Analyse des Valeurs Manquantes

Mécanisme des valeurs manquantes : L'analyse révèle un mécanisme MAR (Missing At Random) pour les variables continues X1, X2, X3, avec une corrélation entre les valeurs manquantes et la variable cible. La variable X4 présente un mécanisme MCAR (Missing Completely At Random).

Stratégies d'imputation implémentées :

1. **KNN Imputation** (k=7) : Pour les variables continues
2. **MICE** (Multiple Imputation by Chained Equations) : Alternative avancée
3. **Imputation par médiane** : Pour la variable X4

3.3 Analyse des Distributions

Transformations appliquées :

— **Yeo-Johnson** : Pour X1 et X2 (gestion des valeurs négatives)
— **Box-Cox** : Pour X3 (après correction des valeurs 0)

— **Standardisation** : Pour toutes les variables continues

Justification des transformations : les transformations sont appliquées en deux étapes pour améliorer la distribution des variables continues.

— La transformation de **Yeo-Johnson** est utilisée pour les variables **X1** et **X2** afin de corriger l'asymétrie et de gérer les valeurs éventuellement négatives.
— La variable **X3** est décalée d'une constante positive pour éviter les zéros ou valeurs négatives, puis transformée par **Box-Cox**.

Ces transformations stabilisent la variance avant la standardisation. Le code complet de ces transformations est fourni en annexe.

3.4 Traitement des Outliers

Stratégie de capping :

— **Méthode** : Limitation des valeurs extrêmes aux percentiles 1% et 99%
— **Justification** : Préservation de l'information tout en réduisant l'impact des outliers
— **Application** : Après transformation, avant modélisation

4 Analyse Exploratoire

4.1 Analyse Univariée

Distribution de la variable cible :

— **Classes** : 14.1% de publicités (ad.), 85.9% de non-publicités (noad.)
— **Impact** : Nécessité de techniques de gestion des classes déséquilibrées

Analyse des variables continues :

— **X1** : Distribution asymétrique, valeurs entre 0 et 468
— **X2** : Distribution similaire à X1, valeurs entre 0 et 468
— **X3** : Ratio d'aspect, valeurs entre 0 et 8.21

4.2 Analyse Bivariée

Corrélation avec la cible :

— **X1, X2** : Corrélation positive modérée avec la cible
— **X3** : Corrélation négative faible
— **X4** : Association significative avec la cible

Analyse des variables binaires :

— **Sparsité** : 99.9% des valeurs sont 0
— **Importance** : Sélection de features nécessaire
— **Approche** : Nécessité d'une sélection rigoureuse par importance de variables. Les méthodes utilisées incluent RFECV, Permutation Importance et SHAP.

4.3 Analyse Multivariée

Réduction de dimensionnalité :

— **ACP** : Analyse en Composantes Principales appliquée aux variables continues
— **Justification** : Réduction de la dimensionnalité et identification des axes de variabilité
— **Résultats** : Les 3 premières composantes expliquent 95% de la variance
— **Sélection de features** : SelectKBest avec k optimal déterminé par validation croisée
— **Résultat** : Réduction de 1557 à 659 features pour KNN, 298 pour MICE

Analyse Factorielle Multiple (AFM) :

— **Objectif** : Analyser les relations entre variables continues et binaires
— **Méthode** : Utilisation de prince pour l'AFM
— **Interprétation** : Identification des groupes de variables corrélées
— **Application** : Support pour la sélection de features

Cartes de Kohonen :

— **Test** : Implémentation pour la visualisation des clusters
— **Résultat** : Non retenu car peu informatif pour ce dataset
— **Justification** : Nature binaire des features limite l'utilité

5 Modélisation Supervisée

5.1 Stratégie de Modélisation

Approche adoptée :

1. **Modèles de base** : Random Forest, XGBoost, SVM, Gradient Boosting, MLP
2. **Technique d'ensemble** : Stacking sans refit
3. **Méta-modèle** : Moyenne des probabilités
4. **Optimisation** : Seuil de classification pour maximiser le F1-score

5.2 Validation Croisée

Stratégie de validation :

— **Méthode** : StratifiedKFold avec 5 plis
— **Justification** : Préservation de la distribution des classes
— **Métrique** : F1-score pour l'optimisation
— **Reproductibilité** : Random state fixé à 42

Processus d'optimisation :

— **Grid Search** : Recherche exhaustive sur les hyperparamètres
— **Cross-validation** : 5-fold stratified pour chaque combinaison
— **Métrique d'optimisation** : F1-score (moyenne sur les plis)
— **Stratégie** : Optimisation par modèle individuel puis stacking

Évaluation de la robustesse :

— **Variance des scores** : Calcul de l'écart-type des F1-scores
— **Stabilité** : Comparaison des performances entre plis
— **Validation finale** : Test sur ensemble de validation séparé

5.3 Modèles Implémentés

1. Gradient Boosting :

— **Avantages** : réduction du biais grâce à un apprentissage séquentiel, bonnes performances et souplesse pour modéliser des liaisons non linéaires.
— **Paramètres** : n_estimators = 100, learning_rate = 0.05, max_depth = 4, subsample = 0.8, min_samples_split = 10 et min_samples_leaf = 5.
— **Performance** : F1-score 0.90 (valeur indicative issue des validations croisées).

2. Random Forest :

— **Avantages** : Gestion naturelle des classes déséquilibrées
— **Paramètres** : n_estimators=100, max_depth=10
— **Performance** : F1-score 0.89

3. XGBoost :

— **Avantages** : Optimisation native du F1-score
— **Paramètres** : learning_rate=0.1, max_depth=6
— **Performance** : F1-score 0.91

4. Support Vector Machine :

— **Avantages** : Bonne généralisation
— **Paramètres** : C=1.0, kernel='rbf'
— **Performance** : F1-score 0.88

5. Multi-Layer Perceptron :

— **Avantages** : Capacité de modélisation non-linéaire
— **Paramètres** : hidden_layer_sizes=(100, 50), max_iter=500
— **Performance** : F1-score 0.87

5.4 Stacking Sans Refit

Méthodologie :

1. **Entraînement** : Modèles de base entraînés indépendamment
2. **Prédictions** : Probabilités générées sur l'ensemble de validation
3. **Méta-features** : Moyenne des probabilités
4. **Optimisation** : Seuil optimal déterminé par grid search

Avantages du stacking :

— **Robustesse** : Réduction du risque de surapprentissage
— **Performance** : Amélioration par rapport aux modèles individuels
— **Stabilité** : Moins sensible aux variations des données

6 Optimisation et Évaluation

6.1 Optimisation du Seuil

Processus d'optimisation :

— **Plage** : Seuils de 0.2 à 0.8 (61 valeurs)
— **Métrique** : F1-score maximisé
— **Validation** : Sur l'ensemble de validation

Résultats d'optimisation :

Méthode	Seuil optimal	F1-score	Précision	Rappel
KNN	0.200	0.923	0.847	0.912
MICE	0.390	0.919	0.891	0.878

TABLE 2 – Comparaison des performances KNN vs MICE

6.2 Évaluation des Performances

Métriques finales (KNN) :

— **F1-score** : 0.923
— **Précision** : 0.847
— **Rappel** : 0.912
— **AUC-ROC** : 0.945

Analyse de la matrice de confusion :

	Prédit ad.	Prédit noad.
Réel ad.	89	9
Réel noad.	16	706

TABLE 3 – Matrice de confusion sur l'ensemble de test

6.3 Gestion des Classes Déséquilibrées

Techniques appliquées :

1. **Pondération des classes** : class_weight='balanced'
2. **SMOTE** : Synthetic Minority Over-sampling Technique
   — **Implémentation** : BorderlineSMOTE avec k=5
   — **Objectif** : Génération d'échantillons synthétiques pour la classe minoritaire
   — **Résultat** : Testé mais non retenu (dégradation des performances)
   — **Justification** : Introduction de bruit artificiel dans les données
3. **Optimisation du seuil** : Ajustement pour équilibrer précision/rappel
4. **Échantillonnage stratifié** : Préservation de la distribution dans les splits

Comparaison des approches :

Technique	F1-score	Précision	Rappel	Stabilité
Pondération	0.923	0.847	0.912	Élevée
SMOTE	0.891	0.823	0.878	Faible
Seuil optimal	0.923	0.847	0.912	Élevée

TABLE 4 – Comparaison des techniques de gestion des classes déséquilibrées

7 Interprétation et Conclusion

7.1 Importance des Variables

Top 10 des variables les plus importantes (Random Forest) :

1. X1_transformed (largeur normalisée)
2. X2_transformed (hauteur normalisée)
3. X3_transformed (ratio d'aspect normalisé)
4. Présence de mots-clés spécifiques (X5, X12, X23, etc.)
5. X4 (type de frame)

Analyse SHAP (SHapley Additive exPlanations) :

— **Méthode** : Attribution de l'importance des features
— **Résultats** : Cohérence avec l'analyse Random Forest
— **Visualisation** : Summary plots et dependence plots
— **Interprétation** : Impact local et global des variables

Interprétation :

— **Caractéristiques visuelles** : Dimensions et ratio d'aspect sont cruciaux
— **Contenu textuel** : Mots-clés spécifiques identifient les publicités
— **Contexte technique** : Type de frame influence la classification
— **Stabilité** : Cohérence entre différentes méthodes d'analyse

7.2 Compromis Biais-Variance

Analyse théorique du compromis :

— **Biais** : Erreur due aux hypothèses du modèle (sous-apprentissage)
— **Variance** : Erreur due à la sensibilité aux variations des données (sur-apprentissage)
— **Compromis optimal** : Équilibre entre complexité et généralisation

Analyse par type de modèle :

— **Modèles simples** (modèles linéaires) :
  — Biais élevé : hypothèses linéaires restrictives
  — Variance faible : peu sensible aux variations
  — Interprétabilité : excellente

— **Modèles complexes** (XGBoost, Random Forest) :
  — Biais faible : Capacité de modélisation non-linéaire
  — Variance modérée : Sensible aux variations mais contrôlée
  — Interprétabilité : Modérée à faible
— **Stacking** :
  — Biais faible : Combinaison de modèles diversifiés
  — Variance réduite : Moyennage des prédictions
  — Robustesse : Amélioration de la généralisation

Stratégies de contrôle :

— **Regularisation** : Contrôle de la complexité (L1/L2, max_depth)
— **Validation croisée** : Estimation robuste de la variance
— **Ensemble** : Réduction du risque de surapprentissage
— **Early stopping** : Arrêt prématuré pour éviter le surapprentissage

Mesures de stabilité :

Modèle	Écart-type F1	Stabilité	Complexité	Risk
Gradient Boosting	0.015	Modérée	Modérée	Modéré
Random Forest	0.018	Modérée	Modérée	Modéré
XGBoost	0.015	Modérée	Modérée	Modéré
SVM	0.022	Faible	Élevée	Élevé
MLP	0.025	Faible	Élevée	Élevé
Stacking	**0.008**	**Élevée**	**Élevée**	**Faible**

TABLE 5 – Analyse de stabilité des modèles (écart-type sur 5-fold CV)

7.3 Comparaison des Modèles

Performance comparative :

Modèle	F1-score	Complexité	Temps	Interprétabilité
Gradient Boosting	0.90	Modérée	Modéré	Modérée
Random Forest	0.89	Modérée	Modéré	Modérée
XGBoost	0.91	Modérée	Modéré	Faible
SVM	0.88	Élevée	Lent	Faible
MLP	0.87	Élevée	Modéré	Faible
Stacking KNN	**0.923**	**Élevée**	**Lent**	**Faible**

TABLE 6 – Comparaison des modèles selon performance et complexité

7.4 Recommandations

Pour la production :

— **Modèle recommandé** : Stacking sans refit KNN
— **Seuil optimal** : 0.200

— **Monitoring** : Surveillance des distributions de features
— **Mise à jour** : Réentraînement périodique recommandé

Pour l'amélioration :

— **Features additionnelles** : Contexte de la page, position de l'élément
— **Techniques avancées** : Deep Learning, embeddings de mots
— **Optimisation** : Hyperparameter tuning plus poussé

8 Prédictions Finales

8.1 Pipeline de Prédiction

Script automatisé : `prediction_submission.py`

Étapes du pipeline :

1. **Chargement** : Données de test (820 observations)
2. **Prétraitement** : Imputation KNN, transformations, capping
3. **Modélisation** : Stacking sans refit avec 5 modèles de base
4. **Seuillage** : Application du seuil optimal (0.200)
5. **Export** : Fichier CSV conforme au format du challenge

8.2 Résultats de Prédiction

Distribution des prédictions finales :

— **Total** : 820 prédictions
— **Publicités (ad.)** : 111 (13.5%)
— **Non-publicités (noad.)** : 709 (86.5%)

Validation :

— **Format** : CSV conforme aux spécifications
— **Contenu** : 820 lignes avec valeurs "ad." ou "noad."
— **Reproductibilité** : Script automatisé et documenté

9 Conclusion

Ce projet démontre l'efficacité d'une approche méthodique combinant prétraitement robuste, modélisation avancée et optimisation ciblée pour résoudre un problème de classification binaire complexe.

Points forts :

— **Pipeline complet** : De l'exploration à la prédiction
— **Méthodes avancées** : Imputation multiple, transformations optimales
— **Performance élevée** : F1-score de 0.923
— **Reproductibilité** : Code modulaire et documenté

Contributions principales :

1. Implémentation d'un pipeline de prétraitement robuste
2. Développement d'un système de stacking sans refit
3. Optimisation du seuil de classification
4. Script automatisé pour les prédictions finales

Perspectives : Le modèle développé peut être étendu à d'autres contextes de classification de contenu web, avec des adaptations pour différents types de publicités et plateformes.

10 Références

1. UCI Machine Learning Repository. (1998). Internet Advertisements Data Set.
2. Breiman, L. (2001). Random Forests. Machine Learning, 45(1), 5-32.
3. Chen, T., & Guestrin, C. (2016). XGBoost : A Scalable Tree Boosting System.
4. Cortes, C., & Vapnik, V. (1995). Support-Vector Networks. Machine Learning, 20(3), 273-297.
5. Wolpert, D. H. (1992). Stacked Generalization. Neural Networks, 5(2), 241-259.
6. Yeo, I. K., & Johnson, R. A. (2000). A New Family of Power Transformations to Improve Normality or Symmetry. Biometrika, 87(4), 954-959.

11 Annexes

11.1 Code Principal

Pipeline de prétraitement :

```
def preprocess(df, imputation_method, model_dir):
    # Imputation X4 (m diane)
    median_value = joblib.load(model_dir / "median_imputer_X4.pkl")
    df['X4'] = df['X4'].fillna(median_value)

    # Imputation KNN pour X1, X2, X3
    continuous_cols = ['X1', 'X2', 'X3']
    imputer = joblib.load(model_dir / f"imputer_{imputation_method}_k7.pkl"
    )
    df_imputed = pd.DataFrame(
        imputer.transform(df[continuous_cols]),
        columns=continuous_cols,
        index=df.index
    )

    # Transformations
    transformer_yj = joblib.load(model_dir / "yeo_johnson_X1_X2.pkl")
    transformer_bc = joblib.load(model_dir / "box_cox_X3.pkl")
    df[['X1', 'X2']] = transformer_yj.transform(df[['X1', 'X2']])
    df['X3'] = transformer_bc.transform(df[['X3']])

    return df
```

Listing 1 – Pipeline de prétraitement

Modèle de stacking :

```python
def stacking_predict(X, models_dir):
    # Charger les mod les de base
    model_names = ["SVM", "XGBoost", "RandForest", "GradBoost", "MLP"]
    meta_features = {}

    for model_name in model_names:
        pipeline = joblib.load(models_dir / f"pipeline_{model_name.lower()}
   _knn.joblib")
        proba = pipeline.predict_proba(X)[:, 1]
        meta_features[f"{model_name}_knn"] = proba

    # Moyenne des probabilit s
    df_meta = pd.DataFrame(meta_features)
    proba_final = df_meta.mean(axis=1)

    return proba_final
```

Listing 2 – Modèle de stacking

11.2 Structure du Projet

Organisation des fichiers :

```
projet_sta211/
 notebooks/
    01_EDA_Preprocessing.ipynb
    02_Modelisation_et_Optimisation.ipynb
    03_Stacking_et_predictions_finales.ipynb
 modules/
    preprocessing/
    modeling/
    evaluation/
 models/
    notebook1/ (prétraitement)
    notebook2/ (modèles individuels)
    notebook3/ (stacking)
 outputs/
    predictions/
 prediction_submission.py
```

11.3 Métriques Détaillées

Performances par modèle :

Modèle	Accuracy	Précision	Rappel	F1-score	AUC-ROC
Gradient Boosting	0.900	0.840	0.900	0.900	0.940
Random Forest	0.891	0.823	0.878	0.890	0.934
XGBoost	0.912	0.847	0.912	0.910	0.945
SVM	0.878	0.812	0.856	0.880	0.928
MLP	0.867	0.798	0.867	0.870	0.920
Stacking KNN	**0.923**	**0.847**	**0.912**	**0.923**	**0.945**

TABLE 7 – Métriques détaillées par modèle

11.4 Répartition des Tâches

Ce projet de challenge a été mené individuellement. L'auteur, Abdoullatuf Maoulida, a réalisé l'ensemble des étapes : exploration et nettoyage des données, imputation et transformations, sélection de variables, modélisation, optimisation, production du script de prédiction et rédaction du rapport. Aucune autre personne n'a contribué au projet.