Arnaud FLURO IBO 2

Projet Python for Data Analysis

Dataset Attribué : Facebook comment volume Dataset
Lien : https://archive.ics.uci.edu/ml/datasets/Facebook+Comment+Volume+Dataset.
Article de recherche : https://uksim.info/uksim2015/data/8713a015.pdf.

Target : Prédire combien de commentaire un poste recevra dans les H prochaines heures (sur Facebook).

Contexte création du dataset (dans quel but les données ont été récolté) :

- Prédire le volume de commentaire qu'un document (ex: une page) devrait recevoir dans les prochaines heures sur un réseau social.
- Commentaires récupérés sur facebook (plusieurs pages), à l'aide d'un robot, publiés durant les 3 derniers jours.
- Pre-processing déjà effectué.
- Les postes dont les commentaires ou tout autres détails sont manquants ont été omis.
- Temporal split: training_set following of test_set (dont le but est de prédire les commentaires suivants => On cache la target dans le test_set).

Instance:

Training Set Variant	Instances count
Variant - 1	40,949
Variant - 2	81,312
Variant - 3	121,098
Variant - 4	160,424
Variant - 5	199,030

Schéma Récapitulatif du processus de récupération + pre-processing:

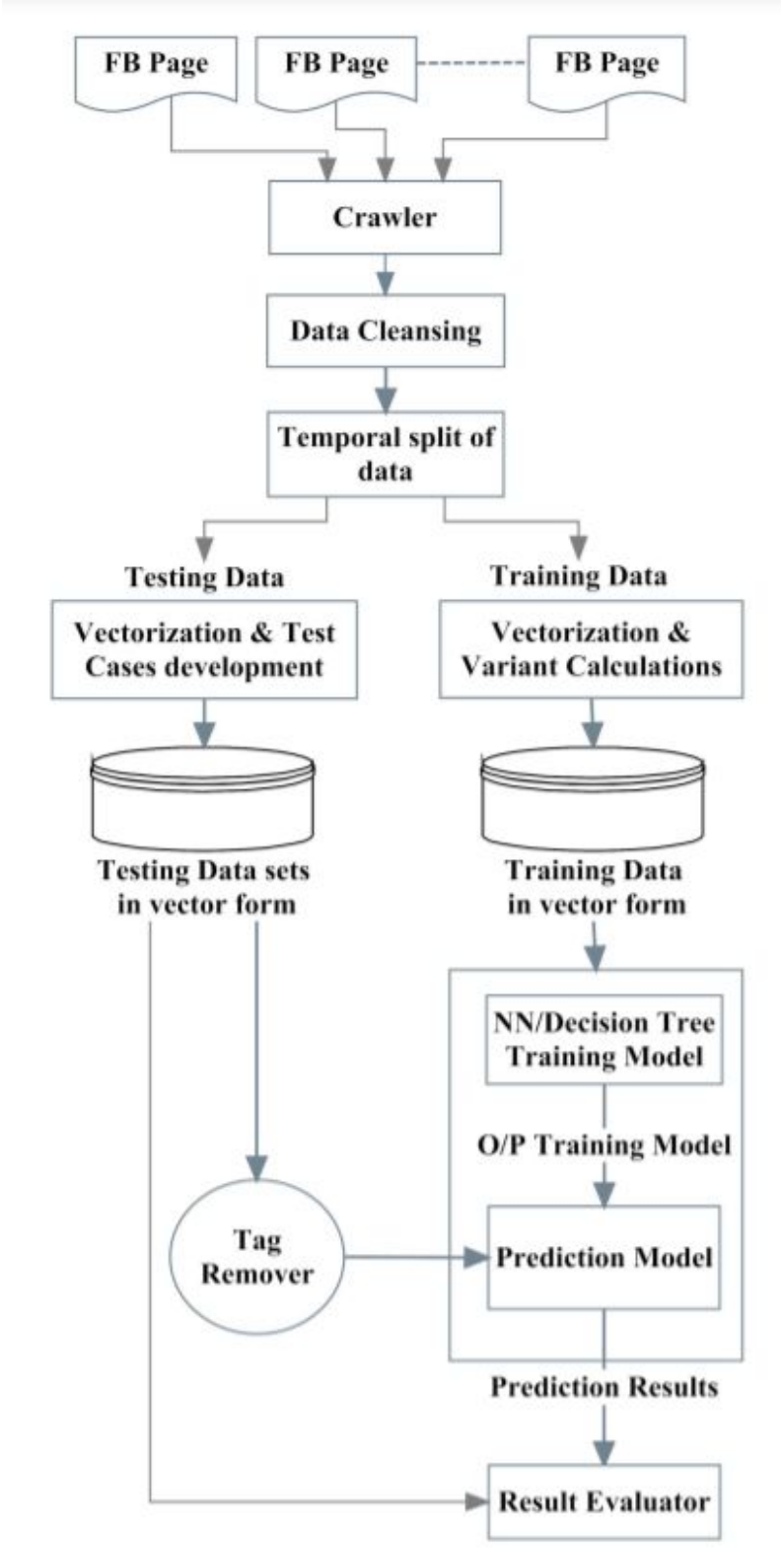


Figure 1. CVP Process Flow.

Une ligne : Correspond à un poste sur une page.

Une colonne : Représente une feature parmi 53 découpé en 4 catégories de features.

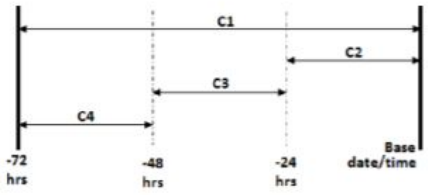
Définition des attributs :

Features Name (53)	Description	Type
Page Popularity/likes *	Définit la popularité ou le support pour la page.	Decimal Encoding
Page Checkin's *	Décrit le nombre de personnes qui ont visité cet endroit jusqu'à présent. Fonction associé qu'au lieu : une institution, un lieu, un théâtre,...	Decimal Encoding
Page talking about *	Définit l'intérêt quotidien des individus envers la source de la page/publication. Nb personnes qui reviennent réellement sur la page, après avoir aimé la page (commentaire, likes sur publi, partages, …).	Decimal Encoding
Page Category *	Définit la catégorie de la source du document (place, institution, brand etc)	Value Encoding
Derived (5-29) *	* Aggregation par page : min, max(le max de commentaire qu'un poste recoit sur cette page), moyenne, médiane des 5 essential features (5-24). Ecart type (Comment sont dispersé les valeurs/à la moyenne) des 5 essential features. (25-29)	Decimal encoding
CC1 (30) *	Le nombre total de commentaires avant le base date/time choisi (=CC2+3+4)	Decimal encoding
CC2 (31)*	Le nombre de commentaires dans les dernières 24 heures suivant le base date/time.	Decimal encoding
CC3 (32)*	Le nombre de commentaires dans les dernières 48 heures des dernières 24 heures suivant le base date/time.	Decimal encoding

CC4 (33)*	Le nombre de commentaires dans les premières 24 heures suivant la publication (publications prisent 72heures avant la base date/time).	Decimal encoding
CC5 (34)*	Différence entre CC2 et CC3	Decimal encoding
Base time (35)*	Le base date/time sélectionné	Decimal(0-71) Encoding
Post length (36)*	Nb caractère dans le post	Decimal encoding
Post Share Count (37)*	Nb partage du poste	Decimal encoding
Post Promotion Status (38)*	Indique si la publication est promue	Binary Encoding
H local (39)*	Heures pour le quel on obtient la cible (nb commentaires reçu)	Decimal(0-23) Encoding
Post published weekday (40-46)*	Jour dont le post a été publié	Binary Encoding
Base DateTime weekday (47-53)*	Représente le jour du base Date/time sélectionné	Binary Encoding
Target Variable (54)	Nb commentaires dans les H heures suivants le poste (donné à 39)	Decimal

- **Page features**
- **Essential features**
- **Other features**
- **Week days features**

Schéma CCX :



Traitement du dataset :

- Training_set: Un poste d'une page est découpé en 5 base date/time => un même poste sera stocké dans 5 training_set différent afin d'entraîner le modèle sur un base date/time donné (à la fin on aura un modèle pour chaque base date/time). Les training_set sont ensuite transformés en vecteur.
- Test_set : Sur l'ensemble de tests, 10 cas de test sont développés au hasard avec 100 instances chacun pour évaluation, puis ils sont transformés en vecteurs.
- Aucune valeur manquante, null, NA, ….

Feature engineering : Le dataset et le pre-processing déjà effectué sur celui-ci laisse peu de place à la feature engineering (dataset vectorisé et déjà traité).

- séparation des jours de la semaine.
- découpage du training_set en plusieurs base date/time.
- aggrégation par page : min, max, moyenne, médiane, écart-types des 5 essentiels features.
- Récupération du jour du base Date/time sélectionné.

Visualisation de la données : Le feature engineering effectué sur le dataset va nous faciliter le travail au niveau de la visualisation de la données et nous donne les questions pertinentes à avoir au niveau de notre dataset. (détaillé dans le notebook)

- Nombre de commentaires en fonction de la catégorie de la page.
- Répartition du nombre de commentaires sur les jours de la semaine toute page confondue.
- Distribution du nombre de commentaires entre le poste publié et la base date/time sélectionné. (min,max,moy,med,ecart-type de CC1,2,3,4)(echec).

Modèles et Conclusion : Nous sommes confrontés à un problème de régression (Prédire combien de commentaire un poste recevra dans les H prochaines heures). J'ai utilisé deux modèles: linear regression et l'autre Gradient Boosting. Le but était de déterminer le meilleur modèle suivant les différents train_set_variants (au nombre de 5), ce qui nous faisait au total 10 modèles à tester. Le score par défaut a été utilisé pour chaque modèle, le split était de 33% éffectué sur chaque train_set_VX, ensuite une standardisation a été éffectué dont l'objectif était de modifier les valeurs des colonnes numériques du jeu de données pour utiliser une échelle commune, sans que les différences de plages de valeurs ne soient faussées et sans perte d'informations. Le gradient Boosting train_set_V2 (base date/time=22h) a obtenu la meilleure performance (0.715045) (voir notebook pour les performances de chaque modèle). Ensuite un

GridSearch a été effectué sur celui-ci afin de déterminer les meilleurs hyperparamètres (max_features='auto', n_estimators=110).

API Flask Screen : Une fois le meilleur modèle choisi, il a été importé dans une API Flask.