Documentation de référence

Flux Boutiques — Prévisions d'activité boutique

Table des matières

1. Objectifs & Contexte métier

> **Résumé** Flux Boutiques est une application de prévision d'activité hebdomadaire destinée à améliorer l'aide à la décision pour le pilotage commercial de réseaux de boutiques. Elle est conçue pour fournir des analyses "prêtes à l'emploi", fiables, et exploitables même par des profils non experts.

Objectif

- Fournir une **vision prospective** sur la fréquentation des boutiques.
- Améliorer l'objectivation des prévisions sur des périodes mensuelles ou trimestrielles.
- Simplifier l'accès à la prévision pour les utilisateurs métier.

Public cible

- Utilisateurs métiers, responsables BI, exploitants opérationnels **non experts en data science**.
- Aucun prérequis technique pour utiliser l'interface Streamlit, mais une vigilance est attendue sur la **qualité des données** à intégrer.

Enjeux

- **Simplicité** de la solution : workflow réduit à l'essentiel.
- **Fiabilité** : qualité des prévisions dépendante de la donnée importée.
- **Pas d'intégration cloud ou volumétrie massive** prévue à ce stade.

2. Cycle de vie des données

Source

- **Fichier** : `Flux_brut.xlsx`
    - Unique source métier, à mettre à jour **manuellement** par l'utilisateur (bouton ou procédure "actualiser" dans Excel).
    - **Aucune transformation attendue** de la part de l'utilisateur à l'import.

Qualité & contrôle

- L'utilisateur doit **contrôler la cohérence et la complétude** des données avant chaque import :

- Exclure ou corriger les valeurs aberrantes (ex : capteurs défaillants).
- Relativiser les prévisions si les données sources sont incomplètes.

Cycle d'utilisation

```
graph TD
  A(Mise à jour du fichier Excel) --> B(Lancer l'application Streamlit)
  B --> C(Cliquer sur "Mettre à jour les données")
  C --> D(Lancer la prévision)
  D --> E{Prévision insatisfaisante ou incohérente ?}
  E -- Oui --> F(Contrôler la qualité des données, mettre à jour le modèle)
  E -- Non --> G(Utilisation des résultats pour la décision)
```

---

3. Fonctionnalités & Scénarios utilisateur

Parcours type

1. **Mettre à jour les données** dans `Flux_brut.xlsx`.
2. **Lancer l'application Streamlit**, puis cliquer sur "Mettre à jour les données".
3. **Lancer la prévision** pour la boutique/secteur choisi.
4. **Consulter et exporter** les résultats (Excel).
5. Si la prédiction est insatisfaisante ou incohérente :
    - Contrôler la qualité des données.
    - Mettre à jour le modèle de la boutique.

Interface utilisateur

- Navigation simple : page d'accueil, bouton "Mise à jour", module de prévision, export.
- Affichage graphique interactif (courbes, intervalles de confiance).
- Alertes et commentaires intégrés si besoin de correction des données.

> **Note** Les commentaires d'interface (aide, consignes, warning) sont recommandés pour guider l'utilisateur sur la validité des résultats.

---

4. Architecture technique & Modélisation

4.1 Organisation générale du projet

```
Streamlitflux pour documentation/
├── app/                    # modules applicatifs
│   ├── pages/              # pages Streamlit
│   ├── utils/              # scripts d'ETL, modèles et visualisation
│   └── database/           # base SQLite et accès
├── models/                 # modèles SARIMAX par boutique
├── config.py               # chemins et paramètres globaux
├── app.py                  # point d'entrée Streamlit
├── requirements.txt        # dépendances Python
├── Flux_brut.xlsx          # fichier source à mettre à jour manuellement
├── README.md
├── Z-documentation/        # ressources annexes (schéma BDD, script d'init)
│   └── BDD.py              # script d'initialisation de la base boutiques.db
```

- `app.py` orchestre la navigation entre les pages et charge la configuration (voir ci-dessous).
- `config.py` centralise tous les chemins (fichiers historiques, modèles, météo…) ainsi que les paramètres par défaut (latitude/longitude, variables exogènes, etc.).
- `models/` contient un sous-dossier par boutique (`<boutique>_models/`) avec :
    - `sarimax_model_<boutique>.pkl` (modèle entraîné),
    - `scaler_exog_<boutique>.pkl` / `scaler_target_<boutique>.pkl`,
    - `pca_<boutique>.pkl` pour la réduction de dimension.

Détail sur `app.py`

- Point d'entrée unique de l'application : gère le routage, l'état utilisateur et l'initialisation de la configuration.
- Centralise la navigation Streamlit (prévisions, gestion des boutiques, update modèles…).
- Charge et propage la configuration définie dans `config.py`.
- À adapter si de nouvelles pages/fonctions majeures sont ajoutées : toute la navigation part de ce module.

Détail sur `config.py`

- Centralise tous les chemins d'accès critiques (fichiers historiques, modèles, météo…).
- Définit les paramètres par défaut (variables exogènes, latitude/longitude, etc.).
- Gère les paramètres réseau (proxy météo) :
  - `USE_PROXY` (booléen)
  - `PROXY_URL` (adresse du proxy à utiliser pour les appels API météo)
- Permet d'adapter l'application à d'autres environnements sans toucher au cœur du code.
- **Bonne pratique :** documenter chaque constante/chemin pour fiabiliser la maintenance.

> **Note réseau** Si votre environnement nécessite un accès internet via un proxy, adaptez les variables `USE_PROXY` et `PROXY_URL` dans `config.py` pour garantir la récupération des données météo externes.

4.2 Modules Streamlit : pages et interactions

- `pages/predictions.py` : lance la prévision hebdomadaire pour la boutique sélectionnée.
  - Utilise `forecast.py` pour charger le modèle, récupérer les exogènes futures (`exo_var`) et afficher les résultats.
- `pages/update_model.py` : réentraîne le modèle SARIMAX d'une boutique après mise à jour des historiques.
- `pages/manage_boutiques.py` : ajoute ou supprime des boutiques et secteurs dans la base SQLite (`DatabaseManager`).
- `pages/update_all_models.py` : met à jour en une fois l'ensemble des modèles de toutes les boutiques détectées.
- `pages/selector.py` : tableau de bord principal : choix du secteur/boutique, accès aux prévisions ou à la mise à jour.

4.3 Pipeline de données et exogènes

Mise à jour des historiques

- `app/utils/aggregation_fichier_primaire.py` :
  - Convertit le fichier source `Flux_brut.xlsx` en format hebdomadaire (`Flux_final.xlsx`).
  - Met à jour les données météo brutes (`Météo_SUD.xlsx`) via `WeatherDataFetcher`.
  - Crée un fichier exogène complet en appelant `exo_var()` pour générer les colonnes météo, jours fériés, vacances, etc.

Chargement des données pour une boutique

- `app/utils/data_loader.py` lit `Flux_final.xlsx` pour produire :
  - la série cible (`y_hist`),
  - les historiques décalés N-1 et N-2,
  - la table calendrier (Date, Année, Semaine) alignée sur le découpage "semaine personnalisée".

Préparation des exogènes

- `app/utils/exogenous.py` :
  - Ajoute les variables externes (météo, jours fériés, vacances).
  - Agrège les données quotidiennes en semaines personnalisées.
  - Impute les semaines manquantes (régression Ridge) et applique un remplissage avant/arrière puis médiane.

4.4 Entraînement et mise à jour des modèles

- **Optimisation & entraînement** (`app/utils/model_optimiser.py`) :
  - Recherche bayésienne (`skopt.gp_minimize`) sur les paramètres (p, q, P, Q) du SARIMAX.
  - Les exogènes sont normalisées (`StandardScaler`) puis compressées en 5 composantes par Analyse en Composantes Principales (PCA).
  - La métrique de choix repose principalement sur l'AIC (objectif < 500) et la corrélation sur le jeu d'apprentissage.
- **Sauvegarde (`save_model`)**

    - Les artefacts du modèle (SARIMAX, scalers, PCA) sont sérialisés au format joblib dans le sous-dossier correspondant à la boutique.
- **Mise à jour automatique (`auto_update_model_with_latest_data`)**
    - À chaque prévision, le modèle est étendu avec les nouvelles semaines disponibles si les historiques ont évolué, sans réapprentissage complet.

4.5 Base de données SQLite

- La base `app/database/boutiques.db` contient deux tables :
    - `secteurs` (id_secteur, nom_secteur)
    - `boutiques` (id_boutique, nom_boutique, id_secteur)
- Le module `DatabaseManager` gère les accès (ajout, suppression, consultation) et garantit que la suppression d'un secteur échoue s'il possède encore des boutiques associées.

4.6 Bibliothèques et dépendances

- Les dépendances nécessaires sont listées dans `requirements.txt` :
    - streamlit
    - pandas
    - statsmodels
    - plotly
    - requests
    - joblib
    - aiohttp
    - holidays
    - scikit-learn # utilisé pour StandardScaler, PCA, Ridge…
    - skopt
- Toute nouvelle librairie doit être ajoutée à ce fichier pour que l'environnement reste reproductible.
- **Recommandation :** préciser les versions minimales recommandées, par exemple :
    - `scikit-learn>=1.1`
    - `pandas>=1.5`
    - etc.

4.7 Initialisation de la base SQLite

Un script d'exemple est fourni dans `Z-documentation/BDD.py`. Il crée la base `app/database/boutiques.db` et y insère les secteurs/boutiques par défaut.

```
python Z-documentation/BDD.py
```

À exécuter **une seule fois** lors de la mise en place de l'application ou en cas de réinitialisation de la base.

---

5. Gestion des variables exogènes

Variables intégrées (voir `config.py`)

- Température max/min
- Précipitations
- Jours fériés
- Vacances scolaires
- Nombre de jours dans la semaine

Gestion des valeurs manquantes

- Imputation par régression Ridge
- Remplissage avant/arrière (ffill/bfill), puis médiane si besoin

Note sur l'ajout de nouvelles variables exogènes

> **Attention** L'ajout de nouvelles données exogènes dans l'application n'est **pas trivial** et nécessite :
>
> - La modification de la liste des exogènes dans `config.py`.
> - L'adaptation des scripts d'import et de prétraitement (`exogenous.py`, `weather_fetcher.py`), pour intégrer et transformer correctement la nouvelle variable.
> - Parfois, le re-entraîner la totalité des modèles SARIMAX pour garantir la cohérence des prédictions.

6. Restitution & Reporting

Exports disponibles

- Format **Excel uniquement** (`to_excel`)
- Pas de CSV, PDF, ou export direct vers SI tiers
- Tous les graphiques sont intégrés dans l'interface Streamlit (Plotly)

> **Note** Si des formats d'export spécifiques sont requis, il faut les ajouter dans le code (ajout d'un module d'export CSV facile à mettre en place).

7. Sécurité, Robustesse & Pratiques dev

Sécurité

- **Aucun contrôle d'accès** ni gestion d'utilisateur
- Application strictement **locale**

Robustesse

- SQLite locale (pas de backup auto, ni synchronisation)
- **Attention** : la base peut être supprimée par le script BDD si lancé à la main

Pratiques dev

- Pas de pipeline de tests ou d'intégration continue (CI/CD)
- Dépendances listées dans `requirements.txt` (à compléter si besoin)

8. Maintenance, Extension & Passage à l'échelle

Maintenance courante

- Ajout/Suppression de boutiques possible via l'interface
- Nettoyage des chemins d'accès recommandé pour chaque nouvel environnement

Limites et évolutivité

- Application **locale uniquement**
- **Pas conçue pour le cloud** ou le multi-utilisateur
- Toute augmentation de volume nécessitera une refonte ou une migration vers une architecture adaptée (API, serveur, etc.)
- Il est possible de mettre en place une base de données commune sur un cloud pouvant héberger des appels SQL, ce qui est déjà **partiellement mis en œuvre**

9. FAQ, Exemples, Lexique

FAQ

Q : Pourquoi ma prévision échoue-t-elle ? *Vérifiez la complétude et la qualité de votre fichier `Flux_brut.xlsx` (valeurs manquantes, colonnes absentes, etc.).*

Q : Que signifie un AIC élevé (> 500) ? *La dynamique n'est pas parfaitement capturée par le modèle. Essayez de recalculer complètement le modèle, d'enrichir les données ou de corriger les données problématiques.*

Q : Que faire si l'application plante à l'import ou la mise à jour ? *Assurez-vous qu'aucun fichier Excel n'est ouvert. Refermez et relancez.*

Q : Comment ajouter une variable exogène ? *Modifier directement le code source dans `config.py` et les scripts d'import exogènes.*

Lexique

- **AIC** : Critère d'Information d'Akaike, indicateur de performance des modèles statistiques (plus bas = meilleur).
- **SARIMAX** : Modèle de prévision de séries temporelles avec prise en compte des effets saisonniers et variables exogènes.
- **PCA** : Analyse en Composantes Principales, réduction de dimension pour résumer l'information.
- **Exogène** : Variable explicative externe (ex : météo, jours fériés, etc.).

10. Précautions & Bonnes pratiques

> **Toujours** mettre à jour manuellement le fichier source avant chaque prévision, et **fermer tous les fichiers Excel** avant de lancer l'application.
>
> Sauvegarder régulièrement les données, et ne jamais supprimer la base sans en avoir fait une copie.

Fin de la documentation de référence
