Système Intégré de Gestion, Analyse & Scoring des Clients (SIGASC)

PROGRAMMATION PYTHON - Projet de fin d'études

École : École Nationale Supérieure de la Statistique et de l'Économie Appliquée (ENSEA)
Année Académique : 2025-2026
Enseignant : M. IPOU

Introduction

Ce projet vise à amener les étudiants de troisième année Analystes Statisticiens (AS3) à développer une application professionnelle en Python. Cette application devra intégrer une base de données SQLite, une interface Tkinter, une architecture orientée objet, ainsi que des modules d'analyse statistique, de visualisation, de scoring et de détection d'anomalies. Les étudiants devront démontrer leur capacité à manipuler des données, à concevoir une structure logicielle complète et à produire un outil fonctionnel et documenté.

Consignes Générales

Travail de groupe : Les étudiants travailleront par groupe de trois (03).

Ils devront concevoir une application complète permettant la gestion des données clients, leur analyse, la détection d'anomalies et la production d'un score. L'interface Tkinter servira à naviguer entre les différents modules : importation, audit, analyse, visualisation, scoring et exportation. Le projet doit être structuré en programmation orientée objet.

Architecture générale du système

L'application doit être construite en respectant une architecture orientée objet. Au minimum, **six classes** doivent être implémentées :

AppGUI
Gestion de l'interface Tkinter

DatabaseManager
Gestion SQLite (CRUD, requêtes, création de tables)

DataImporter
Importation CSV/XLSX vers SQLite

DataCleaner
Nettoyage, audit et cohérence des données

Analyzer
Calcul statistique et indicateurs

Visualizer
Production de graphiques Matplotlib

ScoringModel
Création d'un score simple ou via un modèle statistique

AnomalyDetector (Optionnel)
Détection d'anomalies (IsolationForest ou règles)

Structure SQLite demandée

La base **clients.db** doit contenir les tables suivantes :

1. Table clients

Champ	Type	Description
id_client	INTEGER PRIMARY KEY	Identifiant unique du client
nom	TEXT	Nom du client
prenom	TEXT	Prénom du client
age	INTEGER	Âge du client
sexe	TEXT	Sexe du client
revenu	FLOAT	Revenu du client
region	TEXT	Région du client
ancienneté	INTEGER	Ancienneté (en années)
score_initial	FLOAT	Score initial du client
date_creation	TEXT	Date de création de l'enregistrement

2. Table opérations/transactions (optionnelle mais recommandée)

Champ	Type	Description
id_operation	INTEGER PRIMARY KEY	Identifiant unique de l'opération
id_client	INTEGER	Référence au client
montant	FLOAT	Montant de l'opération
type_op	TEXT	Type d'opération
date_op	TEXT	Date de l'opération

3. Table scoring

Champ	Type	Description
id_client	INTEGER	Référence au client
score_final	FLOAT	Score final calculé
niveau_risque	TEXT	Niveau de risque (Ex: Bas, Moyen, Élevé)
date_score	TEXT	Date du calcul du score

Relations :
- Un client peut avoir plusieurs transactions
- Un client possède un enregistrement de scoring

Fonctionnalités détaillées de l'application

A – Gestion de la base SQLite
- Création automatique de la base et des tables
- Ajout, modification, suppression et affichage des clients
- Requêtes filtrées par région, âge ou niveau de risque

B – Importation & intégration des données
- Import CSV ou Excel
- Prévisualisation des données dans Tkinter
- Nettoyage automatique : types, dates, valeurs manquantes
- Insertion dans SQLite avec contrôle des doublons

C – Audit & Nettoyage
- Détection des doublons
- Détection des valeurs aberrantes (Z-score, IQR)
- Imputation des valeurs manquantes
- Normalisation et vérifications logiques
- Génération d'un rapport d'audit

D – Analyses statistiques
- Moyenne, médiane, variance
- Répartition par sexe, région
- Tableaux croisés
- Analyse d'ancienneté

E – Visualisations avancées
- Histogrammes
- Heatmaps
- Boxplots
- Séries temporelles
- Scatter plots

F – Détection d'anomalies
- Règles simples (écart-type, incohérences)
- Modèles (IsolationForest, LOF, KMeans)

G – Module de scoring
- Modèle statistique ou machine learning
- Classification des niveaux de risque
- Stockage dans SQLite

H – Exportations
- Export CSV
- Export PDF ou TXT (rapport)
- Export des anomalies et du scoring

Livrables attendus

1. **Code Python complet** avec architecture POO
2. **Base SQLite fonctionnelle**
3. **Application Tkinter exécutable**
4. **Rapport de projet** (10–15 pages)
5. **Vidéo de démonstration** (5–10 minutes)

Barème d'évaluation (sur 100 points)

Critère d'évaluation	Points	Détails
Architecture POO	20	Respect du modèle orienté objet, découpage en classes, encapsulation, héritage si pertinent
Module SQLite	15	Création et gestion de la base, requêtes efficaces, relations entre tables
Importation et intégration	10	Import CSV/Excel, prévisualisation, insertion avec contrôle qualité
Nettoyage & audit	10	Détection et traitement des anomalies, imputation, rapport d'audit
Analyses statistiques	10	Calculs statistiques pertinents, tableaux croisés, indicateurs clés
Visualisations	10	Graphiques clairs, choix adaptés aux données, lisibilité
Détection d'anomalies	10	Implémentation de méthodes de détection, interprétation des résultats
Module scoring	10	Création d'un modèle de scoring, classification des risques, stockage
Exports / Interface / Rapport	5	Fonctionnalités d'export, qualité de l'interface Tkinter, documentation
TOTAL	**100**	

Résultat attendu

Le produit final attendu est une application complète de data analysis intégrant une base SQLite, une interface Tkinter, des modules analytiques, un scoring cohérent et une détection avancée d'anomalies. L'ensemble doit refléter un niveau professionnel et constituer un travail de fin d'études solide.

Objectif pédagogique : Ce projet vise à synthétiser les compétences acquises en programmation Python, analyse de données, statistiques et développement logiciel, en produisant une application fonctionnelle et documentée.