

# Mission 2 – « Carte de Risque IA »

Le fichier **Data\_09092025.xlsx** (4 254 lignes, 16 colonnes ; feuille « Feuil1 ») et repéré la structure suivante :

**ID, Nom, Activité, Année, Liquidité générale, Liquidité immédiate, Taux de croissance du CA, Marge nette, Var Achats, Var CA&VarStock, REX/CA, IS/CA, RN/ACTIF, Ratio des immobilisations corporelles, Délai moyen de règlement des clients, Délai moyen de règlement des fournisseurs.**

Parfait 🎯 Tu veux une **mission simple et pédagogique** pour un débutant en Python & IA appliquée à la finance/comptabilité.

Voici une proposition adaptée à des stagiaires (niveau initiation) :

## Niveau débutant – Carte de Risque IA

### 🎯 Objectif

Construire une **mini-carte de risque** basée sur un petit jeu de données Excel (ou CSV) contenant quelques ratios financiers (ex. liquidité, marge nette, IS/CA).

Le but est **d'appliquer des règles simples** pour calculer un score et visualiser le résultat.

### 📁 Étapes de la mission

#### 1. Importer les données

- Charger un fichier Excel/CSV avec les colonnes suivantes :
  - Nom (société)
  - Liquidité générale
  - Marge nette
  - IS/CA

#### 2. Créer un système de règles simples

- Si **Liquidité générale** > 1 → bon (70 pts), sinon mauvais (30 pts).
- Si **Marge nette** > 5% → bon (70 pts), sinon mauvais (30 pts).
- Si **IS/CA** < 1% → alerte fiscale (20 pts), sinon correct (60 pts).

#### 3. Calculer un Score global

- Score global = moyenne des 3 scores.
- Classer en 3 catégories :

- ● Haut risque si Score global  $\leq 40$
- □ Risque moyen si  $40 < \text{Score} \leq 70$
- □ Faible risque si Score  $> 70$

#### 4. Générer un tableau résultat

Exemple attendu :

<i>Société</i>	<i>Score global</i>	<i>Risque</i>
<i>Alpha SA</i>	30	● Haut
<i>Beta SARL</i>	75	□ Faible
<i>Gamma SARL</i>	55	□ Moyen

#### 5. Visualiser (option)

- Faire un petit graphique (barres colorées ou heatmap).

### 🌀 Exemple de code starter (pour le stagiaire)

```
import pandas as pd

# Charger fichier CSV
df = pd.read_csv("donnees_risque.csv")

# Fonctions de scoring
def score_liquidite(x): return 70 if x > 1 else 30
def score_rentabilite(x): return 70 if x > 0.05 else 30
def score_fiscalite(x): return 20 if x < 0.01 else 60

# Application
df["S_liquidité"] = df["Liquidité générale"].apply(score_liquidite)
df["S_rentabilité"] = df["Marge nette"].apply(score_rentabilite)
df["S_fiscalité"] = df["IS/CA"].apply(score_fiscalite)

# Score global
df["Score global"] =
df[["S_liquidité", "S_rentabilité", "S_fiscalité"]].mean(axis=1)

# Catégorisation du risque
df["Risque"] = pd.cut(df["Score global"],
                      bins=[0,40,70,100],
                      labels=["Haut ●", "Moyen □", "Faible □"])

print(df[["Nom", "Score global", "Risque"]])
```

### ✓ Résultats attendus pour un stagiaire

1. Charger un fichier et afficher les données.
2. Appliquer des **règles de scoring simples**.

3. Obtenir un **tableau clair** des entreprises à risque.
  4. (Bonus) Générer un **graphique simple** (barres ou heatmap).
- 

☞ C'est une **mission courte**, qui permet de :

- Pratiquer la **manipulation de données** avec Pandas,
- Découvrir la logique de **scoring de risque**,
- Produire un **résultat concret et visuel**.

Parfait 🎯 Voici une **mission adaptée à un niveau moyen** (après initiation). Elle est plus riche que la version débutant, mais reste abordable :

---

## Niveau intermédiaire – Carte de Risque IA (Niveau moyen)

### 🎯 Objectif

Créer une **Carte de Risque IA** en utilisant des données financières (ex. ratios : liquidité, rentabilité, délais clients/fournisseurs, IS/CA).

Le stagiaire doit appliquer des **règles métier + techniques de détection d'anomalies** pour identifier les entreprises à risque et produire un **dashboard simplifié**.

### 📁 Étapes de la mission

#### 1. Importer et explorer les données

- Charger un fichier Excel/CSV contenant :
  - Nom, Activité, Année
  - Liquidité générale, Marge nette, IS/CA, Délai clients (DSO), Délai fournisseurs (DPO).
- Faire une **analyse exploratoire rapide** (dimensions, valeurs manquantes, statistiques de base).

#### 2. Créer un scoring multi-dimensionnel

- **Liquidité** : si  $< 1 \rightarrow$  risque élevé.
- **Rentabilité** : si Marge nette  $< 0 \rightarrow$  risque élevé.
- **Fiscalité** : IS/CA anormal ( $< 1\%$  ou  $> 50\%$ ).

- **BFR (DSO–DPO)** : si  $> 60$  jours  $\rightarrow$  alerte trésorerie.

☞ Score global = moyenne pondérée des scores thématiques.

### 3. Détection d'anomalies (bonus IA)

- Utiliser **IsolationForest** ou **Local Outlier Factor (LOF)** sur les colonnes numériques.
- Identifier les entreprises "atypiques" et les comparer avec le score métier.

### 4. Créer une carte de risque

- Générer un **DataFrame final** avec : Nom, Score global, Niveau de risque (●□□), Anomalie détectée (Oui/Non).
- Visualiser avec un **heatmap** ou un **bar chart coloré**.

### 5. Exporter les résultats

- Sauvegarder le tableau final en **Excel** (avec mise en couleur conditionnelle).

## 🌀 Exemple de code starter (simplifié)

```
import pandas as pd
from sklearn.ensemble import IsolationForest

# Charger les données
df = pd.read_excel("donnees_financieres.xlsx")

# --- Scoring métier ---
def score_liquidite(x): return 20 if x < 1 else 70
def score_rentabilite(x): return 20 if x < 0 else 70
def score_fiscalite(x): return 20 if (x < 0.01 or x > 0.5) else 60
def score_bfr(dso, dpo): return 20 if (dso - dpo) > 60 else 70

df["S_liquidité"] = df["Liquidité générale"].apply(score_liquidite)
df["S_rentabilité"] = df["Marge nette"].apply(score_rentabilite)
df["S_fiscalité"] = df["IS/CA"].apply(score_fiscalite)
df["S_BFR"] = df.apply(lambda row: score_bfr(row["Délai clients"],
row["Délai fournisseurs"]), axis=1)

df["Score global"] =
df[["S_liquidité", "S_rentabilité", "S_fiscalité", "S_BFR"]].mean(axis=1)

# --- Catégorisation ---
df["Risque"] = pd.cut(df["Score global"],
                      bins=[0,40,70,100],
                      labels=["Haut ●", "Moyen □", "Faible □"])

# --- Anomalies avec Isolation Forest ---
features = df[["Liquidité générale", "Marge nette", "IS/CA", "Délai
clients", "Délai fournisseurs"]]
model = IsolationForest(contamination=0.1, random_state=42)
df["Anomalie"] = model.fit_predict(features)
```

```
df["Anomalie"] = df["Anomalie"].map({1:"Non", -1:"Oui"})

print(df[["Nom", "Score global", "Risque", "Anomalie"]])
```

## ✓ Résultats attendus

1. Un **tableau final** indiquant Score global, Risque, Anomalie.
2. Une **visualisation** simple (heatmap ou barres colorées).
3. Un **fichier Excel exporté** avec les résultats.

📖 Cette mission pour un stagiaire **niveau moyen** :

- il doit coder les règles,
- comprendre l'IsolationForest,
- générer un dashboard simple.

# Niveau avancé – Carte de Risque IA

## 🎯 Objectif

Construire une **solution complète de Carte de Risque IA** à partir de données financières et comptables, incluant :

- un **pipeline de data engineering**,
- un **moteur de scoring hybride** (règles métier + modèles IA),
- une **détection d'anomalies avancée**,
- un **dashboard interactif** (Streamlit ou Dash),
- un **export automatisé (Excel/PDF)**.

## 📁 Étapes de la mission

### 1. Préparation des données

- Charger un fichier Excel multi-années avec les colonnes :  
Nom, Activité, Année, Liquidité générale, Marge nette, IS/CA,  
RN/ACTIF, Délai clients, Délai fournisseurs, Ratio immobilisations.
- Nettoyer les données : valeurs manquantes, outliers, normalisation par secteur/année.
- Créer un **Feature Store** (DataFrame enrichi) avec :
  - Ratios standardisés (z-score/quantiles).
  - Nouveaux indicateurs : DSO–DPO (BFR), Croissance CA, Volatilité marge.

## 2. Scoring avancé

- **Scoring métier pondéré :**
  - Liquidité (25 %), Rentabilité (25 %), BFR (20 %), Fiscalité (15 %), Structure (15 %).
- **Machine Learning supervisé** (si labels disponibles) :
  - Random Forest / XGBoost pour prédire "Risque Haut vs Bas".
- Combiner **Score métier** + **Score ML** pour un **score final hybride**.

## 3. Détection d'anomalies

- Implémenter **3 méthodes** :
  - Isolation Forest,
  - Local Outlier Factor (LOF),
  - DBSCAN (clustering).
- Comparer leurs résultats et créer un **indice d'anomalie consolidé**.

## 4. Visualisation & Dashboard

- Construire une **application Streamlit** avec :
  - Filtres : Année, Activité.
  - Tableau des entreprises avec Score global, Risque (□□●), Anomalie (Oui/Non).
  - Graphiques :
    - Heatmap des scores,
    - Scatterplot (RN/ACTIF vs Marge nette),
    - Histogrammes DSO/DPO.
- Ajouter un **mode explicabilité** : SHAP values pour expliquer les scores.

## 5. Exports automatiques

- Générer :
  - **Excel** avec mise en couleur conditionnelle,
  - **PDF** avec Top-10 entreprises à risque (reportlab).

## 🌀 Exemple de code starter (extrait avancé)

```
import pandas as pd
from sklearn.ensemble import IsolationForest, RandomForestClassifier
```

```

from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
import shap

# 1. Charger données
df = pd.read_excel("donnees_financieres.xlsx")

# 2. Features
features = ["Liquidité générale", "Marge nette", "IS/CA", "RN/ACTIF",
            "Délai clients", "Délai fournisseurs", "Ratio immobilisations"]
X = df[features].fillna(0)

# 3. Scoring métier (exemple simplifié)
df["Score_metier"] = (0.25*df["Liquidité générale"] +
                     0.25*df["Marge nette"] +
                     0.20*(df["Délai clients"]-df["Délai fournisseurs"]) +
                     0.15*df["IS/CA"] +
                     0.15*df["RN/ACTIF"])

# 4. Détection d'anomalies
iso = IsolationForest(contamination=0.1, random_state=42)
df["Anomalie_IF"] = iso.fit_predict(X)

# 5. (Optionnel) Supervised ML si labels
# X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
# model = RandomForestClassifier().fit(X_train, y_train)

# 6. Explicabilité
explainer = shap.Explainer(iso, X)
shap_values = explainer(X)
shap.plots.beeswarm(shap_values)

```

## ✓ Résultats attendus

1. **Pipeline complet** de traitement (data → scoring → anomalies → export).
2. **Dashboard interactif** avec filtres & graphiques.
3. **Explicabilité des modèles** (SHAP).
4. **Livrables exportés** (Excel + PDF).

## 🚀 Niveau de difficulté

- Compétences mobilisées : Python (pandas, sklearn, shap), Machine Learning, DataViz, Streamlit, reporting automatisé.