Test: Types de Modèles et Machine Learning Bancaire

Niveau: Intermédiaire-Avancé | Durée: 45 minutes | 30 Questions

---

Section A: Types de Modèles (10 questions)

Question 1

Classifiez ces modèles selon leur objectif (Descriptif, Prédictif, Prescriptif):

Modèle/Analyse	Objectif
Statistiques descriptives du portefeuille	?
Prédiction du taux de défaut	?
Optimisation de l'allocation du capital	?
Segmentation RFM	?
Scoring de crédit	?

Voir la réponse

Modèle/Analyse	Objectif
Statistiques descriptives	**Descriptif** (comprendre le passé)
Prédiction du défaut	**Prédictif** (anticiper le futur)
Optimisation de l'allocation	**Prescriptif** (recommander des actions)
Segmentation RFM	**Descriptif** (comprendre les clients)
Scoring de crédit	**Prédictif** (estimer la probabilité de défaut)

Question 2

Quelle est la différence entre apprentissage supervisé et non supervisé?

Voir la réponse

Apprentissage Supervisé: - Données avec **étiquettes** (labels) connues - Le modèle apprend la relation entre X (features) et Y (target) - Exemples: Prédiction de défaut (oui/non), estimation de montant

Apprentissage Non Supervisé: - Données **sans étiquettes** - Le modèle découvre des structures/patterns - Exemples: Segmentation clients, détection d'anomalies

Tableau: | Supervisé | Non Supervisé | |———–|———–| | Classification | Clustering | | Régression | Réduction de dimension | | Avec Y | Sans Y | | Scoring crédit | Segmentation |

Question 3

Pour chaque problème, indiquez s'il s'agit de Classification ou Régression:

Problème	Type
Prédire si un client fera défaut	?
Estimer le montant de fraude potentiel	?
Déterminer le segment d'un client (VIP/Standard/Basique)	?
Prédire la LTV (valeur vie client)	?
Identifier les transactions frauduleuses	?

Voir la réponse

Problème	Type
Défaut client	**Classification binaire** (oui/non)
Montant de fraude	**Régression** (valeur continue)
Segment client	**Classification multiclasse** (3 classes)
LTV	**Régression** (valeur continue)
Transactions frauduleuses	**Classification binaire**

Question 4

Qu'est-ce que PD, LGD, EAD dans le contexte du risque de crédit?

Voir la réponse

PD - Probability of Default: - Probabilité qu'un emprunteur fasse défaut - Typiquement sur 12 mois - Output du modèle de scoring

LGD - Loss Given Default: - Pourcentage de perte en cas de défaut - Dépend des garanties, du type de prêt - Typiquement 40-60% pour prêts non garantis

EAD - Exposure at Default: - Montant exposé au moment du défaut - Pour prêts amortissables: solde restant - Pour lignes de crédit: inclut partie non tirée

Formule Perte Attendue:

```
EL = PD × LGD × EAD
```

Question 5

Pourquoi la régression logistique est-elle préférée pour le scoring bancaire?

Voir la réponse

1. **Interprétabilité:**
   - Coefficients = log-odds ratios
   - On peut expliquer chaque facteur de risque
   - Exigence réglementaire (Basel, BRH)
2. **Probabilités calibrées:**
   - Output directement en probabilité [0,1]
   - Pas besoin de calibration supplémentaire

3. **Stabilité:**
    - Modèle robuste, bien compris
    - Comportement prévisible
4. **Validation réglementaire:**
    - Historique d'approbation par les régulateurs
    - Documentation standardisée
5. **Simplicité:**
    - Facile à implémenter et maintenir
    - Faible risque d'overfitting

Question 6

Qu'est-ce que le Gini coefficient en scoring? Comment le calculer?

Voir la réponse

Définition: Le Gini mesure le pouvoir discriminant d'un modèle de scoring.

Formule:

`Gini = 2 × AUC - 1`

Interprétation: - Gini = 0: Modèle aléatoire (pas de discrimination) - Gini = 1: Modèle parfait - Gini > 0.4: Acceptable pour scoring crédit - Gini > 0.5: Bon modèle

Exemple:

```
from sklearn.metrics import roc_auc_score

auc = roc_auc_score(y_test, y_proba)
gini = 2 * auc - 1

print(f"AUC: {auc:.3f}")
print(f"Gini: {gini:.3f}")
```

Question 7

Différence entre modèle PIT (Point-in-Time) et TTC (Through-the-Cycle)?

Voir la réponse

PIT (Point-in-Time): - Reflète les conditions économiques actuelles - PD varie avec le cycle économique - Utilisé pour: pricing, gestion court terme, IFRS 9 - Plus volatile

TTC (Through-the-Cycle): - Lisse les effets cycliques - PD moyenne sur un cycle complet - Utilisé pour: capital réglementaire, Basel III - Plus stable

Exemple: - En récession: PD_PIT > PD_TTC - En expansion: PD_PIT < PD_TTC - Sur le long terme: PD_PIT ≈ PD_TTC

Question 8

Quels sont les algorithmes appropriés pour chaque cas d'usage bancaire?

Cas d'Usage	Algorithme(s)
Scoring crédit réglementaire	?
Détection de fraude	?
Segmentation clients	?
Prédiction de churn	?

Voir la réponse

Cas d'Usage	Algorithme(s)
Scoring crédit	Régression logistique (interprétabilité requise)
Détection fraude	Random Forest, Isolation Forest, XGBoost
Segmentation	K-Means, Clustering hiérarchique
Prédiction churn	Gradient Boosting, Random Forest

Notes: - Scoring: Interprétabilité > Performance - Fraude: Performance > Interprétabilité (mais explicabilité nécessaire) - Segmentation: Non supervisé - Churn: Balance entre performance et explicabilité

Question 9

Qu'est-ce que la validation Champion-Challenger?

Voir la réponse

Définition: Méthode de test A/B pour modèles en production.

Fonctionnement: 1. **Champion:** Modèle actuel en production 2. **Challenger:** Nouveau modèle candidat 3. On applique les deux sur un échantillon 4. On compare les performances réelles 5. Si Challenger > Champion → promotion

Avantages: - Test en conditions réelles - Mesure l'impact business - Transition progressive - Réduction du risque

Exemple bancaire: - Champion: Scoring logistique actuel - Challenger: Nouveau modèle XGBoost - Test sur 10% des demandes pendant 6 mois - Comparer taux de défaut réel, rentabilité

Question 10

Qu'est-ce que le KS (Kolmogorov-Smirnov) en scoring?

Voir la réponse

Définition: Le KS mesure la séparation maximale entre les distributions cumulées des "bons" et des "mauvais" clients.

Interprétation: - KS = max|F_good(x) - F_bad(x)| - Varie de 0 à 1 (100%) - KS > 40%: Bon modèle de scoring - KS > 50%: Très bon modèle

Calcul:

```python
from scipy.stats import ks_2samp

good = y_proba[y_test == 0]  # Scores des non-défauts
bad = y_proba[y_test == 1]   # Scores des défauts

ks_stat, p_value = ks_2samp(good, bad)
print(f"KS: {ks_stat:.2%}")
```

Relation avec Gini: Généralement, un bon Gini implique un bon KS, mais ils mesurent des aspects différents.

Section B: Machine Learning Appliqué (12 questions)

Question 11

Comment gérer un dataset avec 1% de fraudes et 99% de transactions normales?

Voir la réponse

Problème: Déséquilibre de classes extrême

Solutions:

1. **SMOTE (Synthetic Minority Over-sampling):**

```python
from imblearn.over_sampling import SMOTE
smote = SMOTE(random_state=42)
X_resampled, y_resampled = smote.fit_resample(X_train, y_train)
```

2. **Class weights:**

```python
model = RandomForestClassifier(class_weight='balanced')
```

3. **Sous-échantillonnage de la classe majoritaire:**

```python
from imblearn.under_sampling import RandomUnderSampler
```

4. **Ajuster le seuil de décision:**

```python
# Seuil < 0.5 pour privilégier le recall
y_pred = (y_proba >= 0.1).astype(int)
```

5. **Utiliser des métriques appropriées:**

- Precision, Recall, F1 (pas Accuracy!)
- AUC-PR (Precision-Recall)

Question 12

Expliquez la différence entre Precision et Recall dans le contexte de la fraude:

Voir la réponse

Precision (Précision):

```
Precision = TP / (TP + FP)
```

“Parmi les transactions signalées comme fraudes, combien sont vraiment des fraudes?”

- Haute précision = Peu de faux positifs

- Important pour: éviter de bloquer des clients légitimes

Recall (Rappel/Sensibilité):

`Recall = TP / (TP + FN)`

“Parmi toutes les vraies fraudes, combien avons-nous détectées?”

- Haut recall = Peu de fraudes manquées
- Important pour: limiter les pertes financières

Trade-off: En fraude, on privilégie souvent le **Recall** car une fraude manquée coûte plus cher qu’un faux positif.

Coût: - Faux positif: Coût de vérification (~50 HTG) - Faux négatif: Perte (~15,000 HTG)

Question 13

Qu’est-ce que l’overfitting? Comment le détecter et l’éviter?

Voir la réponse

Définition: Le modèle apprend trop bien les données d’entraînement, y compris le bruit, et ne généralise pas aux nouvelles données.

Signes: - Performance excellente sur train, médiocre sur test - Train accuracy » Test accuracy - Courbes d’apprentissage qui divergent

Comment l’éviter:

1. **Plus de données:**
   - Plus d’exemples = moins d’overfitting
2. **Régularisation:**
   - L1 (Lasso), L2 (Ridge)
   - Réduit la complexité du modèle
3. **Validation croisée:**

```
from sklearn.model_selection import cross_val_score
scores = cross_val_score(model, X, y, cv=5)
```

4. **Early stopping:**
   - Arrêter l’entraînement quand le test error augmente
5. **Simplifier le modèle:**
   - Moins de features
   - max_depth plus petit
   - min_samples_leaf plus grand

Question 14

Décrivez le pipeline complet pour construire un modèle de scoring:

Voir la réponse

Étapes:

1. **Définition du problème:**
   - Variable cible: défaut à 12 mois
   - Population: demandeurs de crédit
2. **Collecte et préparation des données:**

```
# Chargement
df = pd.read_csv('demandes_credit.csv')

# Analyse exploratoire
df.info()
df.describe()
```

3. **Feature engineering:**

```
df['ratio_dette_revenu'] = df['dette'] / df['revenu']
df['log_revenu'] = np.log1p(df['revenu'])
```

4. **Split train/test:**

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, stratify=y, random_state=42
)
```

5. **Prétraitement:**

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

6. **Entraînement:**

```
model = LogisticRegression()
model.fit(X_train_scaled, y_train)
```

7. **Évaluation:**

```
y_proba = model.predict_proba(X_test_scaled)[:, 1]
auc = roc_auc_score(y_test, y_proba)
gini = 2 * auc - 1
```

8. **Validation et documentation**
9. **Déploiement et monitoring**

Question 15

Qu'est-ce que Isolation Forest? Quand l'utiliser?

Voir la réponse

Définition: Algorithme de détection d'anomalies basé sur l'isolation.

Principe: - Les anomalies sont plus faciles à isoler - Moins de splits nécessaires pour les séparer - Score d'anomalie = nombre moyen de splits

Usage:

```
from sklearn.ensemble import IsolationForest

iso = IsolationForest(contamination=0.01, random_state=42)
df['anomalie'] = iso.fit_predict(X)
# -1 = anomalie, 1 = normal

df['score_anomalie'] = iso.decision_function(X)
# Plus négatif = plus anormal
```

Quand l'utiliser: - Détection de fraude - Détection de comportements inhabituels - Données sans labels (non supervisé) - Anomalies sont rares (< 5%)

Question 16

Comment interpréter les coefficients d'une régression logistique?

Voir la réponse

Coefficient β: - Le log des odds ratio - Impact sur le log-odds de l'événement

Odds Ratio = exp(β): - OR > 1: Augmente le risque - OR < 1: Diminue le risque - OR = 1: Pas d'effet

Exemple:

```
# Si coefficient de 'nb_retards' = 0.7
odds_ratio = np.exp(0.7)  # ≈ 2.0

# Interprétation:
# Chaque retard supplémentaire DOUBLE le risque de défaut
```

Tableau: | Variable | Coefficient | Odds Ratio | Interprétation | |———-|———-|———|———-| | nb_retards | 0.7 | 2.0 | Double le risque | | proprietaire | -0.5 | 0.6 | Réduit le risque de 40% | | ratio_dette | 1.2 | 3.3 | Triple le risque |

Question 17

Comment choisir le nombre optimal de clusters en K-Means?

Voir la réponse

Méthode du Coude (Elbow):

```
inertias = []
for k in range(2, 11):
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(X_scaled)
    inertias.append(kmeans.inertia_)

plt.plot(range(2, 11), inertias, 'bo-')
plt.xlabel('Nombre de clusters')
plt.ylabel('Inertie')
plt.title('Méthode du Coude')
```

→ Chercher le "coude" où la courbe s'aplatit

Score Silhouette:

```
from sklearn.metrics import silhouette_score

silhouettes = []
for k in range(2, 11):
    kmeans = KMeans(n_clusters=k, random_state=42)
    labels = kmeans.fit_predict(X_scaled)
    silhouettes.append(silhouette_score(X_scaled, labels))
```

```
# Choisir k qui maximise le silhouette score
```

Considérations métier: - Nombre gérable pour les équipes commerciales - Segments significativement différents - Généralement 4-7 segments en banque

---

Question 18

Quelle métrique utiliser pour évaluer un modèle de régression?

Voir la réponse

MAE (Mean Absolute Error):

```
mae = mean_absolute_error(y_true, y_pred)
```

- Interprétable dans les mêmes unités
- Robuste aux outliers

RMSE (Root Mean Square Error):

```
rmse = np.sqrt(mean_squared_error(y_true, y_pred))
```

- Pénalise plus les grandes erreurs
- Sensible aux outliers

R^2 (Coefficient de détermination):

```
r2 = r2_score(y_true, y_pred)
```

- Proportion de variance expliquée
- Entre 0 et 1 (peut être négatif si très mauvais)

MAPE (Mean Absolute Percentage Error):

```
mape = np.mean(np.abs((y_true - y_pred) / y_true)) * 100
```

- En pourcentage, facile à interpréter

Choix: - Erreur en unités → MAE/RMSE - Comparaison entre modèles → R^2 - Communication business → MAPE

Question 19

Comment valider un modèle de scoring dans le temps?

Voir la réponse

Out-of-Time Validation: 1. Entraîner sur données anciennes 2. Tester sur données plus récentes

```
# Données 2020-2022 pour entraînement
train = df[df['date'] < '2023-01-01']

# Données 2023 pour test
test = df[df['date'] >= '2023-01-01']
```

Walk-Forward Validation:

```python
from sklearn.model_selection import TimeSeriesSplit

tscv = TimeSeriesSplit(n_splits=5)
for train_idx, test_idx in tscv.split(X):
    # Entraîner et évaluer
```

Monitoring en production: - PSI (Population Stability Index) - CSI (Characteristic Stability Index) - Suivi du Gini/KS mensuel - Comparaison prévisions vs réalisé

Recalibration: Recalibrer le modèle si: - PSI > 0.25 (changement significatif) - Gini baisse de plus de 5 points - Taux de défaut prévu ≠ réalisé

Question 20

Qu'est-ce que SHAP? Comment l'utiliser pour expliquer un modèle?

Voir la réponse

SHAP (SHapley Additive exPlanations): Méthode d'explication des prédictions basée sur la théorie des jeux.

Principe: - Calcule la contribution de chaque feature à la prédiction - Basé sur les valeurs de Shapley - Fonctionne pour tout modèle (model-agnostic)

Usage:

```python
import shap

# Créer l'explainer
explainer = shap.TreeExplainer(model)
shap_values = explainer.shap_values(X_test)

# Importance globale
shap.summary_plot(shap_values, X_test)

# Explication individuelle
shap.waterfall_plot(shap.Explanation(
    values=shap_values[0],
    base_values=explainer.expected_value,
    data=X_test.iloc[0],
    feature_names=X_test.columns.tolist()
))
```

Avantages: - Explicabilité locale ET globale - Théoriquement fondé - Requis par certains régulateurs

---

Question 21

Concevoir un système de détection de fraude en temps réel:

Voir la réponse

Architecture:

```
Transaction → Feature Engine → Modèle ML → Décision → Action
                    ↓
```

```
Base de données
(historique client)
```

Features en temps réel: 1. Montant vs moyenne client 2. Localisation vs habituelle 3. Heure de la transaction 4. Fréquence récente 5. Type de merchant

Modèle: - Modèle léger pour temps réel (< 50ms) - Random Forest ou XGBoost - Pré-chargé en mémoire

Décision à 3 niveaux: 1. Score < 0.2: Approuver automatiquement 2. 0.2 ≤ Score < 0.7: Vérification supplémentaire (OTP) 3. Score ≥ 0.7: Bloquer et alerter

Contraintes: - Latence < 200ms - Disponibilité 99.99% - Scalabilité (pic de 5000 tx/s)

Question 22

Qu'est-ce que le PSI (Population Stability Index)?

Voir la réponse

Définition: Mesure le changement dans la distribution d'une variable entre deux périodes.

Formule:

```
PSI = Σ (Actual% - Expected%) × ln(Actual% / Expected%)
```

Interprétation: - PSI < 0.10: Pas de changement significatif - 0.10 ≤ PSI < 0.25: Changement modéré, à surveiller - PSI ≥ 0.25: Changement significatif, action requise

Calcul:

```python
def calculate_psi(expected, actual, bins=10):
    # Découper en bins
    breakpoints = np.percentile(expected, np.linspace(0, 100, bins+1))

    expected_counts = np.histogram(expected, breakpoints)[0]
    actual_counts = np.histogram(actual, breakpoints)[0]

    expected_pct = expected_counts / len(expected)
    actual_pct = actual_counts / len(actual)

    # Éviter division par zéro
    expected_pct = np.where(expected_pct == 0, 0.001, expected_pct)
    actual_pct = np.where(actual_pct == 0, 0.001, actual_pct)

    psi = np.sum((actual_pct - expected_pct) * np.log(actual_pct / expected_pct))
    return psi
```

Section C: Études de Cas (8 questions)

Question 23

Cas: Une banque vous demande de construire un modèle de scoring pour les PME. Quelles sont les principales différences avec un scoring particulier?

Voir la réponse

Différences clés:

Aspect	Particuliers	PME
Données	Revenus, historique crédit	États financiers, secteur
Volume	Beaucoup de données	Moins de données
Défaut	Individuel	Peut affecter plusieurs prêts
Variables	Démographiques	Ratios financiers
Cycle	Plus stable	Lié à l'activité économique

Variables spécifiques PME: - Ratio de liquidité - Ratio d'endettement - Marge opérationnelle - Ancienneté de l'entreprise - Secteur d'activité - Nombre d'employés

Défis: - Données financières parfois peu fiables - Moins d'observations - Hétérogénéité des PME - Corrélation avec l'économie locale

---

Question 24

Cas: Le taux de défaut réel est 2x plus élevé que prévu par le modèle. Que faire?

Voir la réponse

Diagnostic:

1. **Vérifier le PSI:** Changement de population?

```
psi = calculate_psi(train_scores, prod_scores)
```

2. **Analyser par segment:**
   - Défaut par tranche de score
   - Par type de produit
   - Par période
3. **Vérifier les données:**
   - Qualité des données en production
   - Variables manquantes ou modifiées
4. **Vérifier la calibration:**
   - Courbe de calibration
   - Brier score

Actions correctives:

1. **Recalibration simple:**

```
# Ajuster les probabilités
calibration_factor = actual_default_rate / predicted_default_rate
df['PD_calibrated'] = df['PD'] * calibration_factor
```

2. **Re-entraînement:** Si les patterns ont changé
3. **Ajuster les seuils:** Politique d'octroi plus stricte
4. **Documenter:** Rapport à la direction et au régulateur

Question 25

Cas: Concevoir une segmentation client pour une campagne de rétention:

Voir la réponse

Approche:

1. **Identifier les clients à risque:**

```
# Modèle de churn
churn_proba = model_churn.predict_proba(X)[:, 1]
df['risque_churn'] = churn_proba
```

2. **Calculer la valeur client:**

```
df['valeur_client'] = df['revenu_annuel_banque']
```

3. **Créer la matrice Risque × Valeur:**

```
df['segment_retention'] = np.where(
    (df['risque_churn'] > 0.5) & (df['valeur_client'] > df['valeur_client'].median()),
    'Priorité Haute',
    np.where(
        (df['risque_churn'] > 0.5),
        'Priorité Moyenne',
        'Pas de risque'
    )
)
```

4. **Plan d'action:**

Segment	Action	Budget
Haute valeur + Haut risque	Appel manager	50K HTG
Moyenne valeur + Haut risque	Email personnalisé	5K HTG
Faible risque	Aucune	0

5. **Mesurer le ROI:**
   - Taux de rétention
   - Coût par client sauvé
   - Valeur préservée

Question 26

Cas: Un modèle XGBoost surpasse la régression logistique (AUC 0.85 vs 0.78). Recommandez-vous de le déployer?

Voir la réponse

Analyse:

Critère	Logistique	XGBoost
AUC	0.78	0.85
Gini	0.56	0.70
Interprétabilité	Excellente	Moyenne
Validation réglementaire	Facile	Complexe

Critère	Logistique	XGBoost
Maintenance	Simple	Plus complexe

Recommandation nuancée:

Pour scoring crédit (décisions individuelles): → **Rester avec la logistique** ou utiliser XGBoost avec SHAP - Exigence réglementaire d'explication - Droit du client à comprendre le refus

Pour fraude/segmentation: → **XGBoost acceptable** - Moins de contraintes réglementaires - Performance prioritaire

Approche hybride: 1. XGBoost pour le scoring interne 2. Régression logistique simplifiée pour les explications client 3. Documenter les différences

---

Question 27

Cas: Comment mesurer l'impact business d'un modèle de détection de fraude?

Voir la réponse

Métriques techniques: - Precision, Recall, F1 - AUC-ROC

Métriques business:

1. **Pertes évitées:**

```
pertes_evitees = TP * montant_moyen_fraude
```

2. **Coût des faux positifs:**

```
cout_fp = FP * cout_verification
```

3. **Fraudes manquées:**

```
pertes_subies = FN * montant_moyen_fraude
```

4. **Ratio coût-bénéfice:**

```
benefice_net = pertes_evitees - cout_fp - pertes_subies
roi = benefice_net / cout_systeme * 100
```

Exemple: - TP: 100 fraudes détectées × 15,000 HTG = 1,500,000 HTG évités - FP: 500 fausses alertes × 50 HTG = 25,000 HTG coût - FN: 10 fraudes manquées × 15,000 HTG = 150,000 HTG perdus

Bénéfice net: 1,500,000 - 25,000 - 150,000 = **1,325,000 HTG**

Question 28

Cas: Expliquez pourquoi un client avec score 720 a été refusé:

Voir la réponse

Approche avec SHAP:

```python
# Prédiction individuelle
client_idx = X_test[X_test['score'] == 720].index[0]
client_data = X_test.iloc[client_idx]

# Explication SHAP
shap_values = explainer.shap_values(client_data)

# Facteurs qui augmentent le risque
contributions = pd.DataFrame({
    'Variable': X_test.columns,
    'Valeur': client_data.values,
    'Contribution': shap_values
}).sort_values('Contribution', ascending=False)

print("Facteurs de refus:")
print(contributions[contributions['Contribution'] > 0])
```

Exemple de réponse au client: “Votre demande a été refusée pour les raisons suivantes: 1. **Ratio dette/revenu élevé:** 45% (limite: 35%) 2. **Ancienneté emploi insuffisante:** 6 mois (minimum: 12 mois) 3. **Nombre de demandes récentes:** 5 en 3 mois

Vous pourrez soumettre une nouvelle demande dans 6 mois après amélioration de ces indicateurs.”

Question 29

Cas: Monitoring d’un modèle de scoring - que surveiller?

Voir la réponse

Dashboard de monitoring:

Indicateur	Fréquence	Seuil Alerte
Taux d’approbation	Journalier	± 5% vs historique
Score moyen	Journalier	± 10 points
PSI features	Mensuel	> 0.25
Gini/KS	Trimestriel	Baisse > 5 points
Taux défaut prévu vs réel	Trimestriel	Écart > 20%

Métriques spécifiques:

1. **Stabilité des inputs:**

```python
for col in features:
    psi = calculate_psi(train[col], prod[col])
    if psi > 0.25:
        alert(f"PSI élevé pour {col}: {psi:.2f}")
```

2. **Stabilité du score:**

```python
psi_score = calculate_psi(train_scores, prod_scores)
```

3. **Backtest:**

```python
# Défauts réels vs prévisions par décile
actual_by_decile = df.groupby('score_decile')['defaut'].mean()
predicted_by_decile = df.groupby('score_decile')['PD'].mean()
```

4. **Dérive temporelle:**

```python
# Gini par mois
monthly_gini = df.groupby('mois').apply(
    lambda x: 2 * roc_auc_score(x['defaut'], x['PD']) - 1
)
```

Question 30

Résumez les métriques clés pour chaque type de modèle bancaire:

Voir la réponse

Modèle	Métriques Principales	Seuils Acceptables
Scoring Crédit	Gini, KS, AUC	Gini > 0.40, KS > 40%
Fraude	Recall, Precision, F1	Recall > 90%, Precision > 10%
Churn	AUC, Lift	AUC > 0.70, Lift@20% > 3
Segmentation	Silhouette, Séparation	Silhouette > 0.3
LTV	MAE, MAPE, R^2	MAPE < 30%, R^2 > 0.5

Formules à retenir:

```
Gini = 2 × AUC - 1
F1 = 2 × (Precision × Recall) / (Precision + Recall)
Expected Loss = PD × LGD × EAD
Lift@k = Taux cible top k% / Taux cible global
```

Barème

Section	Points
Section A (Q1-10)	30 points
Section B (Q11-22)	40 points
Section C (Q23-30)	30 points
Total	**100 points**

Mnémotechniques

GINI = Gain In Numerical Intelligence → Plus c'est élevé, plus le modèle discrimine

EL = PLD (Perte Liée au Défaut) → EL = PD × LGD × EAD

SMOTE pour les PETITS → Suréchantillonne la classe minoritaire

Recall pour les RISQUES → En fraude, on veut tout capter

"Fit sur Train, Transform sur Test" → Éviter le data leakage