Éthique et Gouvernance des Données - Guide pour Data Analyst Bancaire

Introduction

Dans le secteur bancaire, l'éthique et la gouvernance des données ne sont pas optionnelles - elles sont **réglementaires et essentielles**. Un Data Analyst doit comprendre les implications éthiques de son travail et les cadres de gouvernance qui régissent l'utilisation des données.

Mnémotechnique: **FAIR** - Les données doivent être **F**indable (trouvables), **A**ccessible, **I**nteroperable, **R**eusable.

Partie 1: Fondements de l'Éthique des Données

1.1 Pourquoi l'Éthique est Critique en Banque?

Risque	Conséquence	Exemple
Discrimination algorithmique	Exclusion injuste, poursuites	Scoring crédit biaisé contre certains groupes
Violation vie privée	Amendes, perte de confiance	Fuite données clients
Décisions opaques	Non-conformité réglementaire	Modèle "boîte noire" inexplicable
Biais de données	Prédictions erronées	Données historiques discriminatoires

1.2 Principes Éthiques Fondamentaux

```
1. TRANSPARENCE
   └── Expliquer comment les décisions sont prises

2. ÉQUITÉ (FAIRNESS)
   └── Traitement égal indépendamment du genre, âge, origine

3. CONFIDENTIALITÉ
   └── Protéger les données personnelles

4. RESPONSABILITÉ (ACCOUNTABILITY)
   └── Assumer les conséquences des décisions algorithmiques

5. BÉNÉFICE
   └── L'analyse doit créer de la valeur pour le client aussi
```

1.3 Cadre Légal Haïtien et International

Réglementation	Portée	Exigences clés
BRH (Banque de la République d'Haïti)	National	Reporting, confidentialité bancaire
Secret bancaire haïtien	National	Protection données financières

Réglementation	Portée	Exigences clés
FATF/GAFI	International	AML, KYC, reporting transactions suspectes
Bâle III	International	Gestion des risques, transparence
RGPD (référence)	Européen (bonnes pratiques)	Consentement, droit à l'oubli, portabilité

Partie 2: Biais Algorithmiques et Équité

2.1 Types de Biais dans les Données

Type de biais	Description	Exemple bancaire
Biais historique	Données reflètent discriminations passées	Historique de prêts favorisant certains groupes
Biais de sélection	Échantillon non représentatif	Clients digitaux surreprésentés
Biais de mesure	Variables proxy discriminatoires	Code postal corrélé à l'ethnie
Biais de confirmation	Chercher ce qu'on veut trouver	Ignorer données contredisant l'hypothèse
Biais de survivant	Ignorer les cas "disparus"	Analyser seulement clients fidèles

2.2 Détection des Biais

```python
import pandas as pd
import numpy as np
from sklearn.metrics import confusion_matrix

def analyser_equite(y_true, y_pred, groupe_protege):
    """
    Analyse l'équité d'un modèle entre groupes.

    Args:
        y_true: Labels réels
        y_pred: Prédictions du modèle
        groupe_protege: Array binaire (1 = groupe protégé)

    Returns:
        Métriques d'équité
    """
    # Taux d'approbation par groupe
    taux_groupe_0 = y_pred[groupe_protege == 0].mean()
    taux_groupe_1 = y_pred[groupe_protege == 1].mean()

    # Disparate Impact (DI) - doit être > 0.8
    disparate_impact = taux_groupe_1 / taux_groupe_0 if taux_groupe_0 > 0 else 0
```

```
    # Statistical Parity Difference (SPD) - doit être proche de 0
    spd = taux_groupe_1 - taux_groupe_0

    # Equal Opportunity - TPR par groupe
    from sklearn.metrics import recall_score
    tpr_0 = recall_score(y_true[groupe_protege == 0], y_pred[groupe_protege == 0])
    tpr_1 = recall_score(y_true[groupe_protege == 1], y_pred[groupe_protege == 1])

    return {
        'taux_approbation_majoritaire': f"{taux_groupe_0:.2%}",
        'taux_approbation_protege': f"{taux_groupe_1:.2%}",
        'disparate_impact': f"{disparate_impact:.3f}",
        'di_conforme': disparate_impact >= 0.8,  # Règle des 80%
        'statistical_parity_diff': f"{spd:.3f}",
        'equal_opportunity_diff': f"{tpr_1 - tpr_0:.3f}"
    }

# Exemple: Analyse scoring crédit par genre
# groupe_protege = (df['genre'] == 'F').astype(int)
# resultats = analyser_equite(y_true, y_pred, groupe_protege)
```

2.3 Règle des 80% (Four-Fifths Rule)

```
Disparate Impact = Taux minorité / Taux majorité

Si DI < 0.8 (80%) → Discrimination potentielle
```

Exemple: - Taux d'approbation crédit hommes: 60% - Taux d'approbation crédit femmes: 42% - DI = 42/60 = 0.70 < 0.80 → **Alerte discrimination**

2.4 Variables Proxy Problématiques

```
# Variables qui peuvent servir de proxy discriminatoires
VARIABLES_SENSIBLES = [
    'genre',
    'age',
    'nationalite',
    'religion',
    'etat_civil',
    'handicap'
]

VARIABLES_PROXY_POTENTIELLES = [
    'code_postal',      # Proxy pour origine ethnique/revenu
    'prenom',           # Proxy pour genre/origine
    'employeur',        # Proxy pour niveau social
    'type_telephone',   # Proxy pour revenu
    'langue_preferee'   # Proxy pour origine
]

def detecter_correlation_proxy(df, var_sensible, var_candidates):
    """
    Détecte les variables corrélées avec une variable sensible.
    """
```

```
    correlations = []
    for var in var_candidates:
        if var in df.columns and var_sensible in df.columns:
            # Pour variables catégorielles, utiliser Cramér's V
            from scipy.stats import chi2_contingency

            contingency = pd.crosstab(df[var_sensible], df[var])
            chi2, p, dof, expected = chi2_contingency(contingency)
            n = contingency.sum().sum()
            cramers_v = np.sqrt(chi2 / (n * (min(contingency.shape) - 1)))

            correlations.append({
                'variable': var,
                'cramers_v': cramers_v,
                'alerte': cramers_v > 0.3
            })

    return pd.DataFrame(correlations).sort_values('cramers_v', ascending=False)
```

Partie 3: Explicabilité des Modèles (XAI)

3.1 Pourquoi l'Explicabilité?

Réglementaire: Le client a le droit de comprendre pourquoi son crédit est refusé.

Confiance: Les décideurs doivent comprendre les recommandations du modèle.

Débogage: Identifier pourquoi le modèle fait des erreurs.

3.2 Niveaux d'Explicabilité

Niveau	Description	Méthode
Global	Comprendre le modèle dans son ensemble	Feature importance, PDP
Local	Expliquer une prédiction individuelle	SHAP, LIME
Contrefactuel	"Que changer pour obtenir un autre résultat?"	What-if analysis

3.3 SHAP (SHapley Additive exPlanations)

```
import shap

def expliquer_prediction_shap(model, X, index_client):
    """
    Explique une prédiction individuelle avec SHAP.

    Args:
        model: Modèle entraîné
        X: Features
        index_client: Index du client à expliquer
```

```
    Returns:
        Explication SHAP
    """
    # Créer explainer
    explainer = shap.TreeExplainer(model)  # Pour modèles tree-based
    # Ou: explainer = shap.KernelExplainer(model.predict, X.sample(100))

    # Calculer valeurs SHAP pour ce client
    shap_values = explainer.shap_values(X.iloc[[index_client]])

    # Visualiser
    shap.initjs()
    return shap.force_plot(
        explainer.expected_value,
        shap_values[0],
        X.iloc[index_client]
    )

def generer_rapport_refus_credit(model, X, index_client, feature_names):
    """
    Génère un rapport explicatif pour un refus de crédit.
    """
    explainer = shap.TreeExplainer(model)
    shap_values = explainer.shap_values(X.iloc[[index_client]])[0]

    # Top 5 facteurs négatifs
    feature_impacts = pd.DataFrame({
        'feature': feature_names,
        'impact': shap_values
    }).sort_values('impact')

    rapport = []
    rapport.append("FACTEURS PRINCIPAUX DU REFUS:")
    rapport.append("-" * 40)

    for _, row in feature_impacts.head(5).iterrows():
        if row['impact'] < 0:
            rapport.append(f"• {row['feature']}: Impact négatif de {abs(row['impact']):.3f

    rapport.append("\nPOUR AMÉLIORER VOTRE DOSSIER:")
    rapport.append("-" * 40)
    # Suggestions basées sur les features modifiables

    return "\n".join(rapport)
```

3.4 Feature Importance Globale

```
def importance_globale(model, X, feature_names):
    """
    Calcule l'importance globale des features.
    """
    # Pour modèles tree-based
    if hasattr(model, 'feature_importances_'):
        importance = pd.DataFrame({
```

```
            'feature': feature_names,
            'importance': model.feature_importances_
        }).sort_values('importance', ascending=False)

    # Alternative avec permutation
    else:
        from sklearn.inspection import permutation_importance
        perm_importance = permutation_importance(model, X, y, n_repeats=10)
        importance = pd.DataFrame({
            'feature': feature_names,
            'importance': perm_importance.importances_mean
        }).sort_values('importance', ascending=False)

    return importance

# Vérifier que les features sensibles ne sont pas trop importantes
importance = importance_globale(model, X_test, feature_names)
for var in VARIABLES_PROXY_POTENTIELLES:
    if var in importance['feature'].values:
        imp = importance[importance['feature'] == var]['importance'].values[0]
        if imp > 0.1:  # Seuil arbitraire
            print(f"ATTENTION: {var} a une importance élevée ({imp:.2f})")
```

Partie 4: Gouvernance des Données

4.1 Cadre de Gouvernance

```
GOUVERNANCE DES DONNÉES
├── POLITIQUE
│   ├── Règles d'accès
│   ├── Classification des données
│   └── Rétention et suppression
│
├── QUALITÉ
│   ├── Exactitude
│   ├── Complétude
│   ├── Cohérence
│   └── Actualité
│
├── SÉCURITÉ
│   ├── Chiffrement
│   ├── Contrôle d'accès
│   └── Audit trail
│
└── CONFORMITÉ
    ├── Réglementaire (BRH)
    ├── Reporting
    └── Documentation
```

4.2 Classification des Données

Niveau	Description	Exemples	Contrôles
Public	Peut être divulgué	Taux affichés, brochures	Aucun
Interne	Usage interne	Procédures, organigramme	Authentification
Confidentiel	Accès restreint	Données clients agrégées	Autorisation + chiffrement
Strictement confidentiel	Très sensible	NIN, comptes individuels	Autorisation spéciale + audit

4.3 Principe du Moindre Privilège

```
# Exemple de matrice d'accès par rôle
MATRICE_ACCES = {
    'data_analyst_junior': {
        'donnees_agregees': 'lecture',
        'donnees_individuelles': 'aucun',
        'donnees_sensibles': 'aucun'
    },
    'data_analyst_senior': {
        'donnees_agregees': 'lecture',
        'donnees_individuelles': 'lecture_anonymisee',
        'donnees_sensibles': 'aucun'
    },
    'data_scientist': {
        'donnees_agregees': 'lecture_ecriture',
        'donnees_individuelles': 'lecture',
        'donnees_sensibles': 'lecture_sur_approbation'
    },
    'chef_dpo': {
        'donnees_agregees': 'lecture_ecriture',
        'donnees_individuelles': 'lecture_ecriture',
        'donnees_sensibles': 'lecture_ecriture'
    }
}

def verifier_acces(role, type_donnee, action):
    """
    Vérifie si un rôle peut effectuer une action sur un type de donnée.
    """
    if role not in MATRICE_ACCES:
        return False, "Rôle inconnu"

    permission = MATRICE_ACCES[role].get(type_donnee, 'aucun')

    if permission == 'aucun':
        return False, "Accès refusé"
    elif action == 'lecture' and 'lecture' in permission:
        return True, "Accès autorisé"
    elif action == 'ecriture' and 'ecriture' in permission:
        return True, "Accès autorisé"
    else:
        return False, "Action non autorisée"
```

4.4 Anonymisation et Pseudonymisation

```python
import hashlib
from faker import Faker

fake = Faker('fr_FR')

def anonymiser_donnees(df, colonnes_sensibles, methode='pseudonymisation'):
    """
    Anonymise ou pseudonymise les données sensibles.

    Args:
        df: DataFrame
        colonnes_sensibles: Liste des colonnes à traiter
        methode: 'anonymisation' (irréversible) ou 'pseudonymisation' (réversible avec clé

    Returns:
        DataFrame anonymisé
    """
    df_anon = df.copy()
    mapping = {}

    for col in colonnes_sensibles:
        if col not in df.columns:
            continue

        if methode == 'anonymisation':
            # Remplacement irréversible
            if 'nom' in col.lower() or 'prenom' in col.lower():
                df_anon[col] = [fake.name() for _ in range(len(df))]
            elif 'email' in col.lower():
                df_anon[col] = [fake.email() for _ in range(len(df))]
            elif 'telephone' in col.lower():
                df_anon[col] = [fake.phone_number() for _ in range(len(df))]
            elif 'adresse' in col.lower():
                df_anon[col] = [fake.address() for _ in range(len(df))]

        elif methode == 'pseudonymisation':
            # Hash réversible avec table de correspondance
            mapping[col] = {}
            def hash_value(val):
                if pd.isna(val):
                    return val
                hashed = hashlib.sha256(str(val).encode()).hexdigest()[:12]
                mapping[col][hashed] = val
                return hashed

            df_anon[col] = df[col].apply(hash_value)

    return df_anon, mapping

# Exemple d'utilisation
colonnes_sensibles = ['nom', 'email', 'telephone', 'adresse', 'nin']
df_anonymise, mapping = anonymiser_donnees(df, colonnes_sensibles)
```

4.5 Audit Trail et Logging

```python
import logging
from datetime import datetime
from functools import wraps

# Configuration du logger
logging.basicConfig(
    filename='data_access_audit.log',
    level=logging.INFO,
    format='%(asctime)s | %(levelname)s | %(message)s'
)

def audit_data_access(func):
    """
    Décorateur pour auditer les accès aux données.
    """
    @wraps(func)
    def wrapper(*args, **kwargs):
        # Informations à logger
        user = kwargs.get('user', 'UNKNOWN')
        action = func.__name__
        timestamp = datetime.now().isoformat()

        # Avant exécution
        logging.info(f"USER: {user} | ACTION: {action} | STATUS: STARTED")

        try:
            result = func(*args, **kwargs)
            logging.info(f"USER: {user} | ACTION: {action} | STATUS: SUCCESS")
            return result
        except Exception as e:
            logging.error(f"USER: {user} | ACTION: {action} | STATUS: FAILED | ERROR: {str
            raise

    return wrapper

@audit_data_access
def lire_donnees_clients(query, user=None):
    """
    Lit les données clients avec audit.
    """
    # Exécution de la requête
    return pd.read_sql(query, connection)

# Utilisation
df = lire_donnees_clients("SELECT * FROM clients WHERE segment='VIP'", user="analyst_jdoe"
```

Partie 5: Protection des Données Personnelles

5.1 Droits des Personnes (Inspiré RGPD)

Droit	Description	Implémentation
Accès	Voir ses données	Export sur demande
Rectification	Corriger ses données	Formulaire de modification
Effacement	“Droit à l’oubli”	Suppression sécurisée
Portabilité	Récupérer ses données	Export format standard
Opposition	Refuser traitement	Opt-out marketing
Explication	Comprendre les décisions	Explicabilité modèles

5.2 Consentement

```
TYPES_CONSENTEMENT = {
    'marketing_email': {
        'description': "Recevoir des offres par email",
        'obligatoire': False,
        'par_defaut': False
    },
    'marketing_sms': {
        'description': "Recevoir des offres par SMS",
        'obligatoire': False,
        'par_defaut': False
    },
    'analyse_comportement': {
        'description': "Analyse de comportement pour personnalisation",
        'obligatoire': False,
        'par_defaut': False
    },
    'partage_partenaires': {
        'description': "Partage avec partenaires",
        'obligatoire': False,
        'par_defaut': False
    },
    'scoring_credit': {
        'description': "Évaluation de solvabilité",
        'obligatoire': True,  # Nécessaire pour le service
        'par_defaut': True
    }
}

def verifier_consentement(client_id, type_traitement):
    """
    Vérifie si le client a donné son consentement.
    """
    # Récupérer les consentements du client
    consentements = get_client_consents(client_id)

    if type_traitement not in TYPES_CONSENTEMENT:
        raise ValueError(f"Type de traitement inconnu: {type_traitement}")

    info = TYPES_CONSENTEMENT[type_traitement]

    if info['obligatoire']:
        return True  # Service essentiel
```

```
    return consentements.get(type_traitement, info['par_defaut'])
```

5.3 Rétention et Suppression

```
POLITIQUE_RETENTION = {
    'transactions': {
        'duree_active': '10 ans',  # Obligation légale
        'duree_archive': '5 ans',
        'base_legale': 'Obligation légale (anti-blanchiment)'
    },
    'logs_connexion': {
        'duree_active': '1 an',
        'duree_archive': '2 ans',
        'base_legale': 'Intérêt légitime (sécurité)'
    },
    'marketing': {
        'duree_active': '3 ans après dernière interaction',
        'duree_archive': '0',
        'base_legale': 'Consentement'
    },
    'candidatures_emploi': {
        'duree_active': '2 ans',
        'duree_archive': '0',
        'base_legale': 'Intérêt légitime (recrutement)'
    }
}

def verifier_retention(type_donnee, date_creation):
    """
    Vérifie si les données doivent être supprimées.
    """
    from dateutil.relativedelta import relativedelta

    politique = POLITIQUE_RETENTION.get(type_donnee)
    if not politique:
        return False, "Politique non définie"

    # Parser la durée
    duree_str = politique['duree_active']
    # Simplification: extraire le nombre d'années
    import re
    match = re.search(r'(\d+)\s*an', duree_str)
    if match:
        annees = int(match.group(1))
        date_limite = date_creation + relativedelta(years=annees)

        if datetime.now() > date_limite:
            return True, f"À supprimer (créé le {date_creation}, limite: {date_limite})"

    return False, "Rétention valide"
```

Partie 6: Cas Pratiques Bancaires

6.1 Cas 1: Scoring Crédit Éthique

Problème: Le modèle de scoring refuse plus de demandes de femmes.

```python
# Audit d'équité
def audit_scoring_credit(model, X_test, y_test, df_clients):
    """
    Audit complet du modèle de scoring.
    """
    y_pred = model.predict(X_test)

    # Par genre
    genre_equite = analyser_equite(
        y_test, y_pred,
        (df_clients.loc[X_test.index, 'genre'] == 'F').astype(int)
    )

    # Par âge (jeunes vs autres)
    age_equite = analyser_equite(
        y_test, y_pred,
        (df_clients.loc[X_test.index, 'age'] < 25).astype(int)
    )

    # Par localisation
    loc_equite = analyser_equite(
        y_test, y_pred,
        (df_clients.loc[X_test.index, 'region'] == 'rural').astype(int)
    )

    rapport = {
        'genre': genre_equite,
        'age_jeunes': age_equite,
        'localisation_rurale': loc_equite
    }

    # Alertes
    alertes = []
    if not genre_equite['di_conforme']:
        alertes.append("ALERTE: Disparate Impact genre non conforme")
    if not age_equite['di_conforme']:
        alertes.append("ALERTE: Disparate Impact âge non conforme")
    if not loc_equite['di_conforme']:
        alertes.append("ALERTE: Disparate Impact localisation non conforme")

    return rapport, alertes
```

6.2 Cas 2: Demande d'Effacement

Scénario: Un client demande la suppression de toutes ses données.

```python
def traiter_demande_effacement(client_id, raison):
    """
    Traite une demande de droit à l'oubli.
```

```
    """
    # 1. Vérifier l'identité
    if not verifier_identite(client_id):
        return {'status': 'error', 'message': 'Identité non vérifiée'}

    # 2. Vérifier les obligations légales
    obligations = verifier_obligations_legales(client_id)
    if obligations['pret_en_cours']:
        return {
            'status': 'partial',
            'message': 'Effacement partiel possible (prêt en cours)',
            'donnees_conservees': ['historique_prets', 'garanties']
        }

    if obligations['investigation_aml']:
        return {
            'status': 'refused',
            'message': 'Effacement impossible (investigation en cours)',
            'base_legale': 'Obligation légale AML'
        }

    # 3. Effacement progressif
    plan_effacement = {
        'immediat': ['preferences_marketing', 'historique_navigation'],
        'sous_30_jours': ['donnees_contact', 'historique_transactions'],
        'archive_legale': ['documents_contractuels']  # Conservés selon loi
    }

    # 4. Exécuter l'effacement
    for categorie, donnees in plan_effacement.items():
        if categorie != 'archive_legale':
            supprimer_donnees(client_id, donnees)

    # 5. Confirmer
    return {
        'status': 'success',
        'message': 'Données effacées selon le plan',
        'confirmation_id': generer_confirmation_id()
    }
```

6.3 Cas 3: Explicabilité du Refus de Crédit

```
def generer_lettre_refus_explicable(client_id, model, X_client):
    """
    Génère une lettre de refus avec explications.
    """
    # Prédiction et explication
    prediction = model.predict_proba(X_client)[0]
    explainer = shap.TreeExplainer(model)
    shap_values = explainer.shap_values(X_client)[0]

    # Top 3 facteurs négatifs (modifiables)
    features_modifiables = ['revenus', 'anciennete_emploi', 'ratio_endettement', 'epargne'
```

```
impacts = pd.DataFrame({
    'feature': X_client.columns,
    'impact': shap_values,
    'valeur': X_client.values[0]
})

facteurs_negatifs = impacts[
    (impacts['feature'].isin(features_modifiables)) &
    (impacts['impact'] < 0)
].nsmallest(3, 'impact')

lettre = f"""
RÉPONSE À VOTRE DEMANDE DE CRÉDIT
================================

Cher(e) Client(e),

Nous avons examiné attentivement votre demande de crédit et,
après analyse de votre dossier, nous ne pouvons malheureusement
pas y donner une suite favorable à ce stade.

PRINCIPAUX ÉLÉMENTS CONSIDÉRÉS:

"""

for _, row in facteurs_negatifs.iterrows():
    if row['feature'] == 'ratio_endettement':
        lettre += f"• Votre ratio d'endettement actuel ({row['valeur']:.1%}) "
        lettre += "dépasse notre seuil de confort.\n"
    elif row['feature'] == 'ancienneté_emploi':
        lettre += f"• Votre ancienneté professionnelle ({row['valeur']:.0f} mois) "
        lettre += "est inférieure à notre critère minimum.\n"
    elif row['feature'] == 'epargne':
        lettre += f"• Le niveau d'épargne constaté est insuffisant "
        lettre += "par rapport au montant demandé.\n"

lettre += """

POUR AMÉLIORER VOS CHANCES:

• Réduire vos crédits en cours
• Constituer une épargne régulière
• Attendre d'avoir plus d'ancienneté

Vous pouvez renouveler votre demande après 6 mois.

Cordialement,
UniBank Haiti
"""

return lettre
```

Partie 7: Checklist Éthique pour Data Analyst

7.1 Avant Chaque Projet

```
## Checklist Éthique - Début de Projet

### Données
- [ ] Les données sont-elles collectées légalement?
- [ ] Le consentement approprié existe-t-il?
- [ ] Les données sont-elles nécessaires (minimisation)?
- [ ] La qualité des données est-elle vérifiée?

### Modèle
- [ ] Les variables proxy sont-elles identifiées?
- [ ] Le modèle sera-t-il explicable?
- [ ] Les biais potentiels sont-ils documentés?

### Impact
- [ ] Qui est affecté par ce projet?
- [ ] Y a-t-il des groupes vulnérables concernés?
- [ ] Quels sont les risques si le modèle se trompe?

### Conformité
- [ ] Les exigences réglementaires sont-elles identifiées?
- [ ] La documentation sera-t-elle maintenue?
- [ ] Un audit est-il prévu?
```

7.2 Avant Déploiement

```
## Checklist Éthique - Pré-Déploiement

### Tests d'Équité
- [ ] Disparate Impact vérifié (≥ 0.8)
- [ ] Statistical Parity analysée
- [ ] Equal Opportunity testée

### Explicabilité
- [ ] Feature importance documentée
- [ ] SHAP values disponibles
- [ ] Rapport de refus explicable

### Sécurité
- [ ] Données anonymisées/pseudonymisées
- [ ] Accès restreints configurés
- [ ] Audit trail activé

### Documentation
- [ ] Data lineage documenté
- [ ] Modèle versioned
- [ ] Décisions de design justifiées
```

Résumé et Mnémotechniques

Principes Clés

Principe	Question à se poser
Nécessité	Ai-je vraiment besoin de cette donnée?
Proportionnalité	L'analyse est-elle proportionnée à l'objectif?
Transparence	Puis-je expliquer ce que je fais?
Équité	Tous les groupes sont-ils traités équitablement?
Sécurité	Les données sont-elles protégées?

Mnémotechniques

Concept	Mnémotechnique
Principes éthiques	**TERB** - Transparence, Équité, Responsabilité, Bénéfice
Qualité données	**ECCAT** - Exactitude, Complétude, Cohérence, Actualité, Traçabilité
Droits clients	**AREPO** - Accès, Rectification, Effacement, Portabilité, Opposition
Biais	**SHSCC** - Selection, Historique, Survivant, Confirmation, Confondant
Équité	**80%** - Règle des quatre cinquièmes (Disparate Impact)

Seuils à Retenir

```
Disparate Impact ≥ 0.8 → Conforme
SHAP importance variable sensible < 0.1 → Acceptable
Rétention transactions = 10 ans (légal)
Consentement marketing = opt-in explicite
```

Objectif: Construire des modèles non seulement performants, mais aussi justes, transparents et respectueux des droits des personnes.