Étude de Cas Complète - Statistiques et Analyse de Données

UniBank Haiti - Analyse de Risque de Crédit

Contexte

Le département des risques de UniBank Haiti vous demande d'analyser le portefeuille de prêts pour: 1. Comprendre les facteurs de défaut 2. Évaluer la qualité des segments 3. Proposer des améliorations au processus d'octroi

---

Données Disponibles

Dataset: prets_clients.csv (10,000 lignes)

```
Variables:
- pret_id: Identifiant unique
- client_id: Identifiant client
- montant: Montant du prêt (HTG)
- duree_mois: Durée du prêt en mois
- taux_interet: Taux d'intérêt annuel (%)
- revenu_mensuel: Revenu mensuel du client (HTG)
- age: Âge du client
- anciennete_emploi: Années dans l'emploi actuel
- score_credit: Score de crédit (300-850)
- dette_existante: Dettes existantes (HTG)
- nb_credits_passes: Nombre de crédits passés
- secteur: Secteur d'activité
- type_emploi: Salarié, Indépendant, Entrepreneur
- defaut: 1 si défaut, 0 sinon (variable cible)
```

PARTIE 1: Analyse Exploratoire (EDA)

Question 1.1

Effectuez une analyse univariée des variables clés. Que constatez-vous?

Solution 1.1

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats

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

```python
# 1. Vue d'ensemble
print(f"Shape: {df.shape}")
print(f"\nTypes de données:\n{df.dtypes}")
print(f"\nValeurs manquantes:\n{df.isnull().sum()}")

# 2. Statistiques descriptives - Variables numériques
desc_stats = df.describe()
print("\nStatistiques descriptives:")
print(desc_stats)

# 3. Analyse de la variable cible
print(f"\nDistribution du défaut:")
print(df['defaut'].value_counts(normalize=True))

# 4. Analyse du montant (variable continue clé)
print(f"\nMontant des prêts:")
print(f"  Moyenne: {df['montant'].mean():,.0f} HTG")
print(f"  Médiane: {df['montant'].median():,.0f} HTG")
print(f"  Écart-type: {df['montant'].std():,.0f} HTG")
print(f"  Skewness: {df['montant'].skew():.2f}")
print(f"  Kurtosis: {df['montant'].kurtosis():.2f}")

# 5. Visualisation
fig, axes = plt.subplots(2, 3, figsize=(15, 10))

# Histogramme du montant
axes[0,0].hist(df['montant'], bins=30, edgecolor='black')
axes[0,0].axvline(df['montant'].mean(), color='red', linestyle='--', label='Moyenne')
axes[0,0].axvline(df['montant'].median(), color='green', linestyle='--', label='Médiane')
axes[0,0].set_title('Distribution des Montants')
axes[0,0].legend()

# Box plot du score de crédit
axes[0,1].boxplot(df['score_credit'])
axes[0,1].set_title('Score de Crédit')

# Distribution de l'âge
axes[0,2].hist(df['age'], bins=20, edgecolor='black')
axes[0,2].set_title('Distribution de l\'Âge')

# Défaut par secteur
df.groupby('secteur')['defaut'].mean().plot(kind='bar', ax=axes[1,0])
axes[1,0].set_title('Taux de Défaut par Secteur')

# Score crédit par défaut
df.boxplot(column='score_credit', by='defaut', ax=axes[1,1])
axes[1,1].set_title('Score Crédit vs Défaut')

# Corrélation avec défaut
correlations = df.select_dtypes(include=[np.number]).corrwith(df['defaut'])
correlations.drop('defaut').sort_values().plot(kind='barh', ax=axes[1,2])
axes[1,2].set_title('Corrélation avec Défaut')

plt.tight_layout()
```

```
plt.show()
```

Observations attendues: 1. La distribution des montants est asymétrique positive (skew > 0) 2. Moyenne > Médiane indique des valeurs extrêmes à droite 3. Le taux de défaut varie selon les secteurs 4. Le score de crédit est plus bas pour les défauts 5. Corrélation négative entre score_credit et défaut

---

Question 1.2

Calculez et interprétez les mesures de tendance centrale et de dispersion pour les variables clés.

Solution 1.2

```python
def comprehensive_stats(series, name):
    """Calcul complet des statistiques descriptives"""
    stats_dict = {
        'Variable': name,
        'N': len(series.dropna()),
        'Missing': series.isnull().sum(),
        'Moyenne': series.mean(),
        'Médiane': series.median(),
        'Mode': series.mode().iloc[0] if not series.mode().empty else None,
        'Écart-type': series.std(),
        'Variance': series.var(),
        'Min': series.min(),
        'Q1': series.quantile(0.25),
        'Q3': series.quantile(0.75),
        'Max': series.max(),
        'IQR': series.quantile(0.75) - series.quantile(0.25),
        'CV (%)': series.std() / series.mean() * 100,
        'Skewness': series.skew(),
        'Kurtosis': series.kurtosis()
    }
    return stats_dict

# Appliquer aux variables clés
key_vars = ['montant', 'score_credit', 'revenu_mensuel', 'age', 'anciennete_emploi']
results = []
for var in key_vars:
    results.append(comprehensive_stats(df[var], var))

stats_df = pd.DataFrame(results)
print(stats_df.to_string())
```

Interprétation exemple:

Variable	Analyse
Montant	Skewness +2.1 indique asymétrie droite. CV de 85% montre grande dispersion.
Score crédit	Distribution quasi-normale (skew proche de 0). Médiane ≈ Moyenne.

Variable	Analyse
Revenu	Très asymétrique, quelques hauts revenus. Utiliser la médiane.
Age	Distribution relativement normale, entre 25 et 65 ans.

PARTIE 2: Analyse Bivariée

Question 2.1

Analysez la relation entre le score de crédit et le défaut. Cette relation est-elle statistiquement significative?

Solution 2.1

```
from scipy.stats import ttest_ind, mannwhitneyu, pointbiserialr

# Séparer les groupes
defaut_0 = df[df['defaut'] == 0]['score_credit']
defaut_1 = df[df['defaut'] == 1]['score_credit']

# 1. Statistiques descriptives par groupe
print("Score crédit par statut de défaut:")
print(df.groupby('defaut')['score_credit'].describe())

# 2. Différence des moyennes
diff = defaut_0.mean() - defaut_1.mean()
print(f"\nDifférence des moyennes: {diff:.2f} points")

# 3. Test t (si normalité acceptable)
t_stat, p_value_t = ttest_ind(defaut_0, defaut_1)
print(f"\nTest t:")
print(f"  t-statistic: {t_stat:.3f}")
print(f"  p-value: {p_value_t:.6f}")

# 4. Test Mann-Whitney (non-paramétrique, plus robuste)
u_stat, p_value_u = mannwhitneyu(defaut_0, defaut_1, alternative='two-sided')
print(f"\nTest Mann-Whitney U:")
print(f"  U-statistic: {u_stat:.0f}")
print(f"  p-value: {p_value_u:.6f}")

# 5. Corrélation point-bisériale
r_pb, p_value_pb = pointbiserialr(df['defaut'], df['score_credit'])
print(f"\nCorrélation point-bisériale:")
print(f"  r = {r_pb:.3f}")
print(f"  p-value: {p_value_pb:.6f}")

# 6. Taille de l'effet (Cohen's d)
pooled_std = np.sqrt(((len(defaut_0)-1)*defaut_0.std()**2 + (len(defaut_1)-1)*defaut_1.std()**2) /
                     (len(defaut_0) + len(defaut_1) - 2))
cohens_d = (defaut_0.mean() - defaut_1.mean()) / pooled_std
```

```
print(f"\nTaille de l'effet (Cohen's d): {cohens_d:.3f}")

# Interprétation
alpha = 0.05
if p_value_t < alpha:
    print(f"\n La différence est statistiquement significative (p < {alpha})")
    if abs(cohens_d) < 0.2:
        print("  Mais la taille de l'effet est faible")
    elif abs(cohens_d) < 0.8:
        print("  La taille de l'effet est modérée")
    else:
        print("  La taille de l'effet est forte")
```

Question 2.2

Le taux de défaut diffère-t-il significativement selon le secteur d'activité?

Solution 2.2

```
from scipy.stats import chi2_contingency

# 1. Tableau de contingence
contingency = pd.crosstab(df['secteur'], df['defaut'])
print("Tableau de contingence:")
print(contingency)

# 2. Taux de défaut par secteur
defaut_par_secteur = df.groupby('secteur')['defaut'].agg(['sum', 'count', 'mean'])
defaut_par_secteur.columns = ['Nb_Defauts', 'Total', 'Taux_Defaut']
defaut_par_secteur = defaut_par_secteur.sort_values('Taux_Defaut', ascending=False)
print("\nTaux de défaut par secteur:")
print(defaut_par_secteur)

# 3. Test du Chi-carré
chi2, p_value, dof, expected = chi2_contingency(contingency)
print(f"\nTest Chi-carré:")
print(f"  Chi² = {chi2:.2f}")
print(f"  Degrés de liberté = {dof}")
print(f"  p-value = {p_value:.6f}")

# 4. V de Cramer (mesure d'association)
n = contingency.sum().sum()
min_dim = min(contingency.shape) - 1
cramers_v = np.sqrt(chi2 / (n * min_dim))
print(f"\nV de Cramer: {cramers_v:.3f}")

# Interprétation
if p_value < 0.05:
    print("\n Le taux de défaut diffère significativement selon le secteur")
    if cramers_v < 0.1:
        print("  Association négligeable")
    elif cramers_v < 0.3:
```

```
        print("  Association faible")
    elif cramers_v < 0.5:
        print("  Association modérée")
    else:
        print("  Association forte")
```

Question 2.3

Analysez la corrélation entre les variables numériques. Quelles variables sont les plus corrélées avec le défaut?

Solution 2.3

```
# Matrice de corrélation
numeric_cols = ['montant', 'duree_mois', 'taux_interet', 'revenu_mensuel',
                'age', 'anciennete_emploi', 'score_credit', 'dette_existante',
                'nb_credits_passes', 'defaut']
corr_matrix = df[numeric_cols].corr()

# Visualisation
plt.figure(figsize=(12, 10))
mask = np.triu(np.ones_like(corr_matrix, dtype=bool))
sns.heatmap(corr_matrix, mask=mask, annot=True, fmt='.2f',
            cmap='RdBu_r', center=0, square=True)
plt.title('Matrice de Corrélation')
plt.tight_layout()
plt.show()

# Corrélations avec le défaut (triées)
corr_with_defaut = corr_matrix['defaut'].drop('defaut').sort_values(key=abs, ascending=False)
print("Corrélations avec le défaut (triées par importance):")
print(corr_with_defaut)

# Test de significativité pour les top corrélations
from scipy.stats import pearsonr, spearmanr

print("\nSignificativité des corrélations:")
for var in corr_with_defaut.index[:5]:
    r, p = pearsonr(df[var], df['defaut'])
    print(f"  {var}: r={r:.3f}, p={p:.6f} {'*' if p < 0.05 else ''}")
```

PARTIE 3: Tests d'Hypothèses

Question 3.1

Le département risque affirme que le taux de défaut des prêts agricoles (8%) est significativement supérieur à la moyenne de la banque (5%). Testez cette hypothèse.

Solution 3.1

```python
from statsmodels.stats.proportion import proportions_ztest

# Données
n_agriculture = len(df[df['secteur'] == 'Agriculture'])
defauts_agriculture = df[df['secteur'] == 'Agriculture']['defaut'].sum()
taux_agriculture = defauts_agriculture / n_agriculture

taux_banque = 0.05  # Hypothèse: taux banque = 5%

print(f"Prêts agricoles:")
print(f"  n = {n_agriculture}")
print(f"  Défauts = {defauts_agriculture}")
print(f"  Taux observé = {taux_agriculture:.2%}")

# Test z unilatéral (H1: taux > 5%)
z_stat, p_value = proportions_ztest(
    count=defauts_agriculture,
    nobs=n_agriculture,
    value=taux_banque,
    alternative='larger'
)

print(f"\nTest d'hypothèse:")
print(f"  H0: p = 0.05 (taux = moyenne banque)")
print(f"  H1: p > 0.05 (taux supérieur)")
print(f"\nRésultats:")
print(f"  z-statistic: {z_stat:.3f}")
print(f"  p-value: {p_value:.6f}")

# Intervalle de confiance
from statsmodels.stats.proportion import proportion_confint
ci_lower, ci_upper = proportion_confint(defauts_agriculture, n_agriculture, alpha=0.05)
print(f"\nIntervalle de confiance 95%: [{ci_lower:.2%}, {ci_upper:.2%}]")

# Conclusion
alpha = 0.05
if p_value < alpha:
    print(f"\n  Rejet de H0 (p < {alpha})")
    print("  Le taux de défaut agricole est significativement supérieur à 5%")
else:
    print(f"\n  Non-rejet de H0 (p >= {alpha})")
```

Question 3.2

Comparez les taux de défaut entre les trois types d'emploi (Salarié, Indépendant, Entrepreneur). Y a-t-il une différence significative?

Solution 3.2

```python
from scipy.stats import chi2_contingency, f_oneway, kruskal
```

```python
# 1. Tableau de contingence
ct = pd.crosstab(df['type_emploi'], df['defaut'])
print("Tableau de contingence:")
print(ct)
print("\nTaux de défaut par type d'emploi:")
print(df.groupby('type_emploi')['defaut'].mean().sort_values(ascending=False))

# 2. Test Chi-carré (variables catégorielles)
chi2, p_value, dof, expected = chi2_contingency(ct)
print(f"\nTest Chi-carré:")
print(f"  Chi² = {chi2:.2f}")
print(f"  p-value = {p_value:.6f}")

# 3. ANOVA sur le score de crédit par type d'emploi (exemple additionnel)
groups = [df[df['type_emploi'] == t]['score_credit'].values for t in df['type_emploi'].unique()]
f_stat, p_anova = f_oneway(*groups)
print(f"\nANOVA - Score crédit par type emploi:")
print(f"  F = {f_stat:.2f}")
print(f"  p-value = {p_anova:.6f}")

# 4. Post-hoc si ANOVA significative
if p_anova < 0.05:
    from statsmodels.stats.multicomp import pairwise_tukeyhsd
    tukey = pairwise_tukeyhsd(df['score_credit'], df['type_emploi'], alpha=0.05)
    print("\nPost-hoc Tukey HSD:")
    print(tukey.summary())

# Conclusion
if p_value < 0.05:
    print("\n  Les taux de défaut diffèrent significativement selon le type d'emploi")
else:
    print("\n  Pas de différence significative")
```

PARTIE 4: Modélisation du Risque

Question 4.1

Créez un score de risque simplifié basé sur les variables les plus importantes.

Solution 4.1

```python
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, roc_auc_score

# 1. Préparation des données
features = ['score_credit', 'revenu_mensuel', 'dette_existante', 'anciennete_emploi', 'age']
X = df[features].copy()
y = df['defaut']

# Gérer les valeurs manquantes
```

```
X = X.fillna(X.median())

# 2. Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42, stratify=y)

# 3. Standardisation
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# 4. Régression logistique
model = LogisticRegression(random_state=42)
model.fit(X_train_scaled, y_train)

# 5. Coefficients
coef_df = pd.DataFrame({
    'Variable': features,
    'Coefficient': model.coef_[0],
    'Odds_Ratio': np.exp(model.coef_[0])
}).sort_values('Coefficient', key=abs, ascending=False)

print("Coefficients du modèle:")
print(coef_df)

# 6. Évaluation
y_pred = model.predict(X_test_scaled)
y_prob = model.predict_proba(X_test_scaled)[:, 1]

print("\nRapport de classification:")
print(classification_report(y_test, y_pred))

print(f"\nAUC-ROC: {roc_auc_score(y_test, y_prob):.3f}")

# 7. Score de risque simplifié
def risk_score(row):
    """Score de risque sur 100 points"""
    score = 50  # Score de base

    # Ajustements basés sur les coefficients
    if row['score_credit'] < 500:
        score += 20
    elif row['score_credit'] < 600:
        score += 10
    elif row['score_credit'] > 750:
        score -= 15

    if row['anciennete_emploi'] < 2:
        score += 15
    elif row['anciennete_emploi'] > 5:
        score -= 10

    dti = row['dette_existante'] / max(row['revenu_mensuel'], 1)
    if dti > 0.5:
        score += 20
```

```
    elif dti > 0.3:
        score += 10

    return max(0, min(100, score))  # Borner entre 0 et 100

df['risk_score'] = df.apply(risk_score, axis=1)
print("\nDistribution du score de risque:")
print(df.groupby(pd.cut(df['risk_score'], bins=[0, 30, 50, 70, 100]))['defaut'].mean())
```

PARTIE 5: Recommandations

Question 5.1

Basé sur votre analyse, quelles sont vos recommandations pour améliorer le processus d'octroi de crédit?

Solution 5.1

Recommandations:

1. **Renforcer les critères pour le secteur Agriculture**
   - Taux de défaut = 8% vs 5% moyenne
   - Exiger plus de garanties
   - Réduire les montants ou durées maximales
2. **Utiliser le score de crédit comme critère principal**
   - Corrélation forte avec le défaut (r = -0.45)
   - Seuil minimum recommandé: 550
   - Conditions différenciées par tranche de score
3. **Évaluer le DTI (Debt-to-Income)**
   - Créer une variable DTI = dette_existante / revenu_mensuel
   - Seuil d'alerte > 40%
   - Refus si DTI > 50%
4. **Considérer l'ancienneté d'emploi**
   - Risque accru si < 2 ans
   - Prime de risque ou garantie supplémentaire
5. **Mettre en place un scoring automatique**
   - Score de risque 0-100
   - Décision automatique pour scores extrêmes
   - Analyse manuelle pour scores intermédiaires

Livrables

1. **Rapport EDA** avec visualisations
2. **Résultats des tests statistiques** avec interprétations
3. **Matrice de corrélation** annotée
4. **Modèle de scoring** simplifié
5. **Note de recommandations** pour le comité de crédit

Critères d’Évaluation

Critère	Points
Qualité de l’EDA	25%
Maîtrise des tests statistiques	25%
Interprétation des résultats	20%
Pertinence des recommandations	20%
Code propre et documenté	10%