Test Analyse Univariée et Multivariée - Test 2

Sujet: Analyses Univariées et Multivariées
Niveau: Intermédiaire
Nombre de questions: 20

Questions et Réponses

Q1. Comment choisir le nombre optimal de composantes en PCA?

R1. Critères: 1. **Variance expliquée cumulée ≥ 80-90%** 2. **Règle du coude** (scree plot) 3. **Kaiser:** Eigenvalue > 1

```
# Scree plot
pca = PCA()
pca.fit(X_scaled)

# Variance cumulée
cumsum = np.cumsum(pca.explained_variance_ratio_)
n_components = np.argmax(cumsum >= 0.90) + 1
print(f"Composantes pour 90%: {n_components}")
```

Q2. Comment interpréter les loadings (facteurs de charge) en PCA?

R2. Les **loadings** indiquent la contribution de chaque variable à chaque composante.

Interprétation: - |loading| > 0.5: Forte contribution - Même signe: Variables corrélées positivement - Signes opposés: Variables corrélées négativement

```
loadings = pd.DataFrame(
    pca.components_.T,
    columns=['PC1', 'PC2', 'PC3'],
    index=feature_names
)
# PC1 pourrait être "Rentabilité" si ROE, ROA ont des loadings élevés
```

Q3. Quelle est la différence entre le clustering hiérarchique et K-means?

R3. | K-means | Hiérarchique | |———|———–| | K fixé a priori | K déterminé après | | Clusters sphériques | Formes variées | | O(nkt) | O(n²) ou O(n³) | | Non déterministe | Déterministe | | Grands datasets | Petits/moyens datasets |

Q4. Comment utiliser le score silhouette pour évaluer le clustering?

R4.

```
from sklearn.metrics import silhouette_score, silhouette_samples

# Score global
score = silhouette_score(X, labels)

# Score par échantillon
samples = silhouette_samples(X, labels)
```

```
# Interprétation:
# -1 à 1, plus élevé = meilleur
# > 0.5: Bonne structure
# 0.25-0.5: Structure raisonnable
# < 0.25: Structure faible
```

Q5. Comment gérer les variables catégorielles dans un clustering?

R5. Options: 1. **One-hot encoding** puis K-means 2. **K-modes** (pour catégorielles) 3. **K-prototypes** (mixte) 4. **Gower distance** puis clustering hiérarchique

```
from kmodes.kmodes import KModes
km = KModes(n_clusters=4, init='Huang')
clusters = km.fit_predict(df_categorical)
```

Q6. Comment interpréter une matrice de confusion pour un modèle de scoring?

R6.

```
                Prédit
              0     1
Réel   0    TN    FP   ← Spécificité = TN/(TN+FP)
       1    FN    TP   ← Sensibilité = TP/(TP+FN)
```

Métriques: - **Accuracy:** (TP+TN) / Total - **Precision:** TP / (TP+FP) - Parmi les prédits positifs - **Recall:** TP / (TP+FN) - Parmi les vrais positifs - **F1:** 2 × (Precision × Recall) / (Precision + Recall)

Q7. Comment réaliser une analyse factorielle exploratoire (EFA)?

R7.

```
from factor_analyzer import FactorAnalyzer

# Test KMO et Bartlett
from factor_analyzer.factor_analyzer import calculate_kmo
kmo_all, kmo_model = calculate_kmo(df)
# KMO > 0.6 recommandé

# Analyse factorielle
fa = FactorAnalyzer(n_factors=3, rotation='varimax')
fa.fit(df)

# Loadings
loadings = pd.DataFrame(fa.loadings_, index=df.columns)
print(loadings)
```

Q8. Comment valider les résultats d'un clustering?

R8. 1. **Métriques internes:** Silhouette, Calinski-Harabasz 2. **Stabilité:** Bootstrap, différentes initialisations 3. **Interprétabilité:** Profils des clusters significatifs 4. **Validation métier:** Les segments ont-ils un sens?

```python
# Stabilité avec bootstrap
from sklearn.utils import resample
scores = []
for _ in range(100):
    X_boot = resample(X)
    kmeans = KMeans(n_clusters=4).fit(X_boot)
    scores.append(silhouette_score(X_boot, kmeans.labels_))
print(f"Silhouette: {np.mean(scores):.3f} ± {np.std(scores):.3f}")
```

Q9. Comment utiliser t-SNE pour la visualisation?

R9.

```python
from sklearn.manifold import TSNE

# t-SNE en 2D
tsne = TSNE(n_components=2, perplexity=30, random_state=42)
X_tsne = tsne.fit_transform(X_scaled)

# Visualisation
plt.scatter(X_tsne[:, 0], X_tsne[:, 1], c=labels, cmap='viridis')
plt.title('t-SNE Visualization')
plt.show()
```

Note: Perplexity (5-50) affecte le résultat. t-SNE ne préserve pas les distances globales.

Q10. Comment réaliser une régression logistique avec interprétation?

R10.

```python
import statsmodels.api as sm

X = df[['score_credit', 'dti', 'anciennete']]
X = sm.add_constant(X)
y = df['defaut']

model = sm.Logit(y, X).fit()
print(model.summary())

# Odds Ratios
odds_ratios = np.exp(model.params)
print("\nOdds Ratios:")
print(odds_ratios)
```

Interprétation OR: - OR = 1.5 pour score_credit: Augmentation de 1 point du score → 50% plus de risque de défaut (si coefficient positif)

Q11. Qu'est-ce que la régularisation et quand l'utiliser?

R11. Régularisation: Pénalité sur les coefficients pour éviter le surapprentissage.

Type	Pénalité	Effet
L1 (Lasso)	$\Sigma\|\beta\|$	Certains $\beta \to 0$
L2 (Ridge)	$\Sigma\beta^2$	β réduits

Type	Pénalité	Effet
Elastic Net	L1 + L2	Combinaison

```
from sklearn.linear_model import LogisticRegression

# Avec régularisation L1
model = LogisticRegression(penalty='l1', solver='saga', C=0.1)
model.fit(X_train, y_train)
```

Q12. Comment interpréter une courbe ROC?

R12. - **AUC = 0.5:** Modèle aléatoire - **AUC = 0.7-0.8:** Acceptable - **AUC = 0.8-0.9:** Bon - **AUC > 0.9:** Excellent

```
from sklearn.metrics import roc_curve, auc

fpr, tpr, thresholds = roc_curve(y_test, y_proba)
roc_auc = auc(fpr, tpr)

plt.plot(fpr, tpr, label=f'ROC (AUC = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend()
```

Q13. Comment calculer le coefficient de Gini pour un modèle de scoring?

R13.

```
# Gini = 2 × AUC - 1
gini = 2 * roc_auc - 1

# Ou directement
from sklearn.metrics import roc_auc_score
gini = 2 * roc_auc_score(y_test, y_proba) - 1
```

Interprétation: - Gini = 0: Pas de discrimination - Gini = 0.4: Bon modèle - Gini = 0.6+: Excellent modèle

---

Q14. Comment réaliser une analyse de sensibilité des résultats?

R14.

```
# 1. Variation des paramètres
for k in range(2, 10):
    kmeans = KMeans(n_clusters=k)
    score = silhouette_score(X, kmeans.fit_predict(X))
    print(f"k={k}: silhouette={score:.3f}")

# 2. Cross-validation
from sklearn.model_selection import cross_val_score
scores = cross_val_score(model, X, y, cv=5, scoring='roc_auc')
print(f"AUC: {scores.mean():.3f} ± {scores.std():.3f}")
```

```python
# 3. Bootstrap confidence interval
bootstrap_scores = [...]  # Voir Q8
```

Q15. Comment créer des features d'interaction?

R15.

```python
# Manuel
df['score_x_revenu'] = df['score'] * df['revenu']

# Avec PolynomialFeatures
from sklearn.preprocessing import PolynomialFeatures
poly = PolynomialFeatures(degree=2, interaction_only=True, include_bias=False)
X_interactions = poly.fit_transform(X)
feature_names = poly.get_feature_names_out(X.columns)
```

Q16-20. [Questions additionnelles sur les applications bancaires de ces techniques...]

Scoring

Score	Niveau
0-8	À améliorer
9-13	Intermédiaire
14-17	Avancé
18-20	Expert