Test de Préparation: Types de Variables - Approfondissement

Informations

- **Durée estimée:** 35 minutes
- **Nombre de questions:** 25
- **Niveau:** Intermédiaire-Avancé
- **Thèmes:** Classification avancée, encoding, feature engineering, tests statistiques

---

Section 1: Classification Avancée (5 questions)

Question 1

Un dataset bancaire contient la colonne “niveau_risque” avec les valeurs: “Faible”, “Moyen”, “Élevé”, “Critique”. Quel est le niveau de mesure?

A) Nominal
B) Ordinal
C) Intervalle
D) Ratio

Réponse

B) Ordinal

Les valeurs ont un ordre naturel: Faible < Moyen < Élevé < Critique

Caractéristiques de l’ordinal: - Catégories avec ordre - Écarts entre niveaux non quantifiables - Pas de zéro absolu

Mnémotechnique NOIR: **N**ominal - **O**rdinal - **I**ntervalle - **R**atio

---

Question 2

La variable “température en Celsius” est de quel niveau de mesure?

A) Nominal
B) Ordinal
C) Intervalle
D) Ratio

Réponse

C) Intervalle

L’intervalle a: - Ordre entre valeurs - Écarts quantifiables et égaux (10°C → 20°C = même écart que 20°C → 30°C) - PAS de zéro absolu (0°C ne signifie pas “absence de température”)

Exemples d’intervalle: - Température (Celsius, Fahrenheit) - Dates calendaires - Scores standardisés (QI)

Note: Le ratio aurait un zéro absolu (ex: température en Kelvin)

Question 3

Quelle est la différence entre une variable binaire et une variable dichotomique?

A) Il n'y a pas de différence
B) Binaire est numérique (0/1), dichotomique est catégorielle (Oui/Non)
C) Dichotomique a plus de 2 catégories
D) Binaire est pour les séries temporelles

Réponse

A) Il n'y a pas de différence

Les deux termes désignent une variable à exactement 2 modalités: - Binaire: Souvent codée 0/1 - Dichotomique: Terme statistique équivalent

Exemples: - Défaut de paiement: Oui/Non (0/1) - Genre: M/F - Client actif: True/False

Usage bancaire: Variables cibles pour classification (défaut, fraude, churn)

Question 4

Une variable "code_postal" est de quel type?

A) Quantitative continue
B) Quantitative discrète
C) Qualitative nominale
D) Qualitative ordinale

Réponse

C) Qualitative nominale

Bien que composé de chiffres, le code postal est NOMINAL car: - C'est un identifiant, pas une quantité - Les opérations mathématiques n'ont pas de sens (moyenne des codes postaux?) - Pas d'ordre naturel (6110 n'est pas "plus" que 6100)

⚠ Piège courant: Ne pas confondre "numérique" et "quantitatif"

Autres exemples de nominaux numériques: - Numéro de téléphone - Numéro de compte - Numéro de client

---

Question 5

Quelle variable nécessite un traitement spécial pour le machine learning?

A) Âge en années
B) Revenu mensuel
C) Catégorie socio-professionnelle
D) Montant du prêt

Réponse

C) Catégorie socio-professionnelle

Les variables catégorielles (qualitatives) nécessitent un encoding: - One-Hot Encoding (nominales) - Ordinal Encoding (ordinales) - Target Encoding (haute cardinalité)

```python
# One-Hot pour CSP
pd.get_dummies(df['csp'], prefix='csp')

# Résultat:
# csp_Employé, csp_Cadre, csp_Indépendant, csp_Retraité...
```

Les variables numériques (A, B, D) peuvent être utilisées directement (avec éventuelle standardisation).

Section 2: Encoding et Transformation (5 questions)

Question 6

Quand utiliser One-Hot Encoding vs Label Encoding?

A) One-Hot pour ordinales, Label pour nominales
B) One-Hot pour nominales, Label/Ordinal pour ordinales
C) Toujours One-Hot
D) Toujours Label

Réponse

B) One-Hot pour nominales, Label/Ordinal pour ordinales

Variable	Type	Encoding	Raison
Couleur (R,V,B)	Nominale	One-Hot	Pas d'ordre
Taille (S,M,L,XL)	Ordinale	Ordinal (0,1,2,3)	Ordre préservé
Pays	Nominale	One-Hot / Target	Pas d'ordre
Éducation	Ordinale	Ordinal	Ordre préservé

```python
# One-Hot pour nominales
from sklearn.preprocessing import OneHotEncoder
ohe = OneHotEncoder(sparse=False)
X_encoded = ohe.fit_transform(df[['couleur']])

# Ordinal pour ordinales
from sklearn.preprocessing import OrdinalEncoder
oe = OrdinalEncoder(categories=[['S', 'M', 'L', 'XL']])
df['taille_encoded'] = oe.fit_transform(df[['taille']])
```

Question 7

Une variable "agence" a 150 valeurs uniques. Quel encoding recommandez-vous?

A) One-Hot Encoding
B) Target Encoding ou Frequency Encoding
C) Label Encoding
D) Ne pas encoder

Réponse

B) Target Encoding ou Frequency Encoding

Pour les variables à haute cardinalité (beaucoup de catégories): - One-Hot créerait 150 colonnes (explosion dimensionnelle) - Label Encoding introduit un ordre artificiel

Solutions:

```
# Frequency Encoding: Remplacer par la fréquence
freq = df['agence'].value_counts() / len(df)
df['agence_freq'] = df['agence'].map(freq)

# Target Encoding: Remplacer par la moyenne de la cible
target_mean = df.groupby('agence')['defaut'].mean()
df['agence_target'] = df['agence'].map(target_mean)

# ⚠ Target encoding: Attention au data leakage! Utiliser des folds
```

Question 8

Comment standardiser une variable “revenu” qui a des outliers extrêmes?

A) StandardScaler (z-score)
B) MinMaxScaler
C) RobustScaler
D) Ne pas standardiser

Réponse

C) RobustScaler

Comparaison des scalers:

Scaler	Formule	Sensible aux outliers
StandardScaler	(x - μ) / σ	Oui
MinMaxScaler	(x - min) / (max - min)	Très
RobustScaler	(x - médiane) / IQR	Non

```
from sklearn.preprocessing import RobustScaler

scaler = RobustScaler()
df['revenu_scaled'] = scaler.fit_transform(df[['revenu']])

# RobustScaler utilise la médiane et l'IQR
# Moins affecté par les valeurs extrêmes
```

Utiliser StandardScaler si la distribution est normale sans outliers.

Question 9

Comment créer une variable “ancienneté_en_mois” à partir d’une date d’ouverture de compte?

A) Convertir la date en nombre
B) Calculer la différence avec la date actuelle en mois
C) Utiliser One-Hot sur la date
D) Supprimer la colonne date

Réponse

B) Calculer la différence avec la date actuelle en mois

```
from datetime import datetime

# Calcul de l'ancienneté
df['date_ouverture'] = pd.to_datetime(df['date_ouverture'])
date_reference = datetime.now()

df['anciennete_mois'] = (
    (date_reference - df['date_ouverture']).dt.days / 30.44
).astype(int)

# Ou avec relativedelta pour plus de précision
from dateutil.relativedelta import relativedelta

def calcul_mois(date_ouv):
    delta = relativedelta(datetime.now(), date_ouv)
    return delta.years * 12 + delta.months

df['anciennete_mois'] = df['date_ouverture'].apply(calcul_mois)
```

C'est du feature engineering: transformer une date en variable numérique exploitable.

Question 10

Quelle transformation appliquer à une variable très asymétrique (skewness > 2)?

A) StandardScaler
B) Transformation logarithmique
C) One-Hot Encoding
D) Aucune

Réponse

B) Transformation logarithmique

```
import numpy as np

# Vérifier l'asymétrie
print(f"Skewness avant: {df['montant'].skew():.2f}")

# Transformation log (ajouter 1 pour éviter log(0))
df['montant_log'] = np.log1p(df['montant'])

print(f"Skewness après: {df['montant_log'].skew():.2f}")
```

Transformations pour réduire l'asymétrie: - Log: log(x+1) pour skewness positive - Racine carrée: √x - Box-Cox: Transformation optimale automatique - Yeo-Johnson: Comme Box-Cox mais accepte les valeurs négatives

Importance: Beaucoup de modèles (régression) performent mieux avec des distributions symétriques.

Section 3: Tests Statistiques selon le Type (5 questions)

Question 11

Quel test utiliser pour comparer la moyenne d'une variable continue entre deux groupes?

A) Test du Chi-carré
B) t-test indépendant
C) Corrélation de Pearson
D) ANOVA

Réponse

B) t-test indépendant

Guide des tests selon les variables:

Variable dépendante	Variable indépendante	Test
Continue	Catégorielle (2 groupes)	t-test
Continue	Catégorielle (3+ groupes)	ANOVA
Continue	Continue	Corrélation/Régression
Catégorielle	Catégorielle	Chi-carré

```
from scipy import stats

# t-test: Comparer le revenu entre défaillants et non-défaillants
groupe_defaut = df[df['defaut'] == 1]['revenu']
groupe_ok = df[df['defaut'] == 0]['revenu']

stat, pvalue = stats.ttest_ind(groupe_defaut, groupe_ok)
print(f"p-value: {pvalue:.4f}")
```

Question 12

Pour analyser l'association entre "segment client" (3 catégories) et "type de produit" (4 catégories), quel test utiliser?

A) t-test
B) ANOVA
C) Test du Chi-carré
D) Régression linéaire

Réponse

C) Test du Chi-carré

Le Chi-carré teste l'association entre deux variables catégorielles.

```
from scipy.stats import chi2_contingency

# Créer la table de contingence
contingency = pd.crosstab(df['segment'], df['type_produit'])
print(contingency)

# Test du Chi-carré
chi2, pvalue, dof, expected = chi2_contingency(contingency)
```

```
print(f"Chi² = {chi2:.2f}, p-value = {pvalue:.4f}")

# Si p < 0.05: Association significative
```

V de Cramér pour mesurer la force de l'association:

```
n = contingency.sum().sum()
v_cramer = np.sqrt(chi2 / (n * (min(contingency.shape) - 1)))
```

Question 13

Quelle corrélation utiliser entre une variable ordinale et une variable continue?

A) Pearson
B) Spearman
C) Point-bisérial
D) Chi-carré

Réponse

B) Spearman

Corrélation de Spearman: - Basée sur les rangs - Appropriée pour les ordinales - Robuste aux outliers - Détecte les relations monotones (pas forcément linéaires)

```
from scipy.stats import spearmanr

# Niveau d'éducation (ordinal) vs Revenu (continue)
# Encoder l'ordinal en numérique d'abord
education_map = {'Primaire': 1, 'Secondaire': 2, 'Universitaire': 3, 'Master+': 4}
df['education_num'] = df['education'].map(education_map)

rho, pvalue = spearmanr(df['education_num'], df['revenu'])
print(f"Spearman ρ = {rho:.3f}, p = {pvalue:.4f}")
```

Pearson est pour deux variables continues avec relation linéaire.

Question 14

Comment tester si une variable continue suit une distribution normale?

A) Test du Chi-carré
B) Test de Shapiro-Wilk
C) t-test
D) Corrélation de Pearson

Réponse

B) Test de Shapiro-Wilk

```
from scipy.stats import shapiro, normaltest

# Shapiro-Wilk (recommandé pour n < 5000)
stat, pvalue = shapiro(df['revenu'])
print(f"Shapiro-Wilk: p = {pvalue:.4f}")

# D'Agostino-Pearson (pour n plus grand)
```

```
stat, pvalue = normaltest(df['revenu'])
print(f"D'Agostino: p = {pvalue:.4f}")

# Interprétation:
# p > 0.05: Distribution normale
# p < 0.05: Distribution NON normale
```

Importance: - Détermine le choix entre tests paramétriques et non-paramétriques - Vérifie les hypothèses de la régression

---

Question 15

Une variable binaire (défaut oui/non) et une variable continue (revenu): quelle corrélation?

A) Pearson
B) Spearman
C) Point-biserial
D) Chi-carré

Réponse

C) Point-biserial

La corrélation point-bisériale est une forme de Pearson pour une variable binaire et une continue.

```
from scipy.stats import pointbiserialr

# Corrélation entre défaut (0/1) et revenu
rpb, pvalue = pointbiserialr(df['defaut'], df['revenu'])
print(f"Corrélation point-bisériale: {rpb:.3f}, p = {pvalue:.4f}")

# rpb négatif: Plus le revenu est élevé, moins de défauts
# rpb positif: Plus le revenu est élevé, plus de défauts
```

Note: Mathématiquement équivalent à Pearson, mais l'interprétation est spécifique aux binaires.

Section 4: Feature Engineering Bancaire (5 questions)

Question 16

Comment créer des features à partir d'une date de transaction?

A) Utiliser la date telle quelle
B) Extraire jour, mois, année, jour de semaine, fin de mois
C) Supprimer la colonne
D) Convertir en texte

Réponse

B) Extraire jour, mois, année, jour de semaine, fin de mois

```
df['date'] = pd.to_datetime(df['date_transaction'])

# Extractions temporelles
```

```python
df['annee'] = df['date'].dt.year
df['mois'] = df['date'].dt.month
df['jour'] = df['date'].dt.day
df['jour_semaine'] = df['date'].dt.dayofweek  # 0=Lundi, 6=Dimanche
df['heure'] = df['date'].dt.hour
df['trimestre'] = df['date'].dt.quarter

# Features dérivées
df['weekend'] = (df['jour_semaine'] >= 5).astype(int)
df['fin_mois'] = (df['jour'] >= 25).astype(int)
df['debut_mois'] = (df['jour'] <= 5).astype(int)

# Cyclique (pour saisonnalité)
df['mois_sin'] = np.sin(2 * np.pi * df['mois'] / 12)
df['mois_cos'] = np.cos(2 * np.pi * df['mois'] / 12)
```

Ces features capturent des patterns temporels importants (salaires fin de mois, moins d'activité le weekend, etc.)

Question 17

Comment créer une variable "ratio_utilisation_crédit"?

A) ratio = crédit_utilisé
B) ratio = crédit_utilisé / limite_crédit
C) ratio = limite_crédit - crédit_utilisé
D) ratio = crédit_utilisé + limite_crédit

Réponse

B) ratio = crédit_utilisé / limite_crédit

```python
# Ratio d'utilisation du crédit
df['ratio_utilisation'] = df['credit_utilise'] / df['limite_credit']

# Gérer la division par zéro
df['ratio_utilisation'] = df.apply(
    lambda row: row['credit_utilise'] / row['limite_credit']
                if row['limite_credit'] > 0 else 0,
    axis=1
)

# Ou avec numpy
df['ratio_utilisation'] = np.where(
    df['limite_credit'] > 0,
    df['credit_utilise'] / df['limite_credit'],
    0
)
```

Ce ratio est crucial pour le scoring crédit: - Ratio > 0.7 (70%): Risque élevé - Ratio < 0.3 (30%): Client prudent

---

Question 18

Quelle variable créer pour mesurer la stabilité des revenus?

A) Moyenne des revenus
B) Coefficient de variation des revenus (CV = σ/μ)
C) Maximum des revenus
D) Médiane des revenus

Réponse

B) Coefficient de variation des revenus (CV = σ/μ)

```
# Calculer la stabilité sur les 12 derniers mois
revenus_mensuels = df.groupby('client_id')['revenu_mensuel'].agg(['mean', 'std'])
revenus_mensuels['cv_revenu'] = revenus_mensuels['std'] / revenus_mensuels['mean']

# Interprétation:
# CV faible (< 0.1): Revenus très stables (salarié)
# CV moyen (0.1 - 0.3): Revenus modérément stables
# CV élevé (> 0.3): Revenus instables (indépendant, saisonnier)
```

Le CV (coefficient de variation) normalise l'écart-type par la moyenne, permettant de comparer la variabilité entre clients avec des niveaux de revenus différents.

Question 19

Comment créer un indicateur de "client à risque" combinant plusieurs variables?

A) Utiliser une seule variable
B) Créer un score composite avec pondération
C) Moyenner toutes les variables
D) Choisir aléatoirement

Réponse

B) Créer un score composite avec pondération

```
# Score de risque composite
def calculer_score_risque(row):
    score = 0

    # Facteurs de risque (plus de points = plus de risque)
    if row['ratio_endettement'] > 0.4:
        score += 2
    if row['nb_retards_12m'] > 0:
        score += row['nb_retards_12m']  # 1 point par retard
    if row['anciennete_mois'] < 12:
        score += 1
    if row['ratio_utilisation_credit'] > 0.7:
        score += 2
    if row['revenus_stables'] == 0:  # CV > 0.3
        score += 1

    return score

df['score_risque'] = df.apply(calculer_score_risque, axis=1)
```

```python
# Classification
df['niveau_risque'] = pd.cut(
    df['score_risque'],
    bins=[-1, 2, 4, 100],
    labels=['Faible', 'Moyen', 'Élevé']
)
```

Question 20

Comment traiter une variable “montant_transaction” avec beaucoup de zéros?

A) Supprimer les zéros
B) Créer deux variables: indicateur_transaction (0/1) et montant_si_non_zero
C) Remplacer les zéros par la moyenne
D) Ignorer le problème

Réponse

B) Créer deux variables: indicateur_transaction (0/1) et montant_si_non_zero

C’est le “zero-inflated” pattern, courant en bancaire:

```python
# Variable indicatrice: A-t-il fait une transaction?
df['a_transige'] = (df['montant_transaction'] > 0).astype(int)

# Montant conditionnel (si transaction)
df['montant_si_transige'] = df['montant_transaction'].replace(0, np.nan)

# Pour le ML, deux approches:
# 1. Two-part model: Prédire si transaction, puis prédire montant
# 2. Log-transform: log(1 + montant)
df['log_montant'] = np.log1p(df['montant_transaction'])
```

Cette approche capture deux informations distinctes: 1. Le comportement (transigeant vs inactif) 2. L’intensité (combien quand il transige)

Section 5: Cas Pratiques (5 questions)

Question 21

Un modèle de scoring crédit reçoit une erreur car “segment” contient des valeurs inconnues en production. Comment prévenir ce problème?

A) Ignorer les erreurs
B) Utiliser handle_unknown=‘ignore’ dans OneHotEncoder
C) Supprimer les lignes
D) Convertir en numérique

Réponse

B) Utiliser handle_unknown=‘ignore’ dans OneHotEncoder

```python
from sklearn.preprocessing import OneHotEncoder

# Entraînement
```

```
ohe = OneHotEncoder(
    handle_unknown='ignore',  # Ignore les catégories inconnues
    sparse=False
)
ohe.fit(X_train[['segment']])

# Production - nouvelle catégorie 'PLATINE' non vue
X_prod = pd.DataFrame({'segment': ['RETAIL', 'PREMIUM', 'PLATINE']})
X_encoded = ohe.transform(X_prod)  # 'PLATINE' aura tous les 0s
```

Alternative: `handle_unknown='infrequent_if_exist'` pour regrouper les rares.

Toujours tester le pipeline avec des données réalistes avant la mise en production!

Question 22

Comment valider qu'un encoding ordinal respecte l'ordre correct?

A) Vérifier visuellement
B) S'assurer que les categories sont explicitement définies dans le bon ordre
C) Utiliser LabelEncoder
D) L'ordre n'a pas d'importance

Réponse

B) S'assurer que les categories sont explicitement définies dans le bon ordre

```
from sklearn.preprocessing import OrdinalEncoder

# MAUVAIS: LabelEncoder ne respecte pas l'ordre sémantique
# Il utilise l'ordre alphabétique

# BON: OrdinalEncoder avec catégories explicites
categories_education = [['Primaire', 'Secondaire', 'Universitaire', 'Master', 'Doctorat']]

oe = OrdinalEncoder(categories=categories_education)
df['education_encoded'] = oe.fit_transform(df[['education']])

# Vérification
print(dict(zip(categories_education[0], range(5))))
# {'Primaire': 0, 'Secondaire': 1, 'Universitaire': 2, 'Master': 3, 'Doctorat': 4}
```

L'ordre est crucial pour les modèles qui exploitent les relations numériques (régression, arbres)!

Question 23

Un dataset a une variable "ville" avec 500 villes différentes. Comment la traiter efficacement?

A) One-Hot Encoding (500 colonnes)
B) Grouper par région/département puis One-Hot
C) Supprimer la variable
D) Utiliser Label Encoding

Réponse

B) Grouper par région/département puis One-Hot

```
# Approche 1: Groupement hiérarchique
# Mapper les villes aux régions (10-20 régions au lieu de 500 villes)
ville_to_region = {
    'Port-au-Prince': 'Ouest',
    'Pétionville': 'Ouest',
    'Cap-Haïtien': 'Nord',
    # ...
}
df['region'] = df['ville'].map(ville_to_region)
df_encoded = pd.get_dummies(df['region'], prefix='region')

# Approche 2: Target Encoding
target_mean = df.groupby('ville')['defaut'].mean()
df['ville_risk'] = df['ville'].map(target_mean)

# Approche 3: Frequency Encoding
freq = df['ville'].value_counts(normalize=True)
df['ville_freq'] = df['ville'].map(freq)

# Approche 4: Garder les top N villes, grouper le reste en "Autre"
top_villes = df['ville'].value_counts().head(20).index
df['ville_grouped'] = df['ville'].apply(
    lambda x: x if x in top_villes else 'Autre'
)
```

Question 24

Comment détecter si une variable numérique devrait être traitée comme catégorielle?

A) Vérifier si elle a moins de 10 valeurs uniques
B) Vérifier si les valeurs sont des identifiants (codes, numéros)
C) Les deux (A et B)
D) Toutes les numériques restent numériques

Réponse

C) Les deux (A et B)

Indices qu'une variable numérique est en fait catégorielle:

```
def analyser_type_variable(df, col):
    n_unique = df[col].nunique()
    n_total = len(df)
    ratio = n_unique / n_total

    print(f"Colonne: {col}")
    print(f"Valeurs uniques: {n_unique}")
    print(f"Ratio: {ratio:.2%}")

    # Heuristiques
    if n_unique <= 10:
        print("→ Probablement catégorielle (peu de valeurs)")
```

```
    elif ratio < 0.01:
        print("→ Probablement catégorielle (faible cardinalité)")
    else:
        print("→ Probablement numérique continue")

    # Vérifier si ce sont des codes
    sample = df[col].head()
    print(f"Échantillon: {sample.tolist()}")

# Exemples:
# - "nombre_enfants": 0-10 valeurs → Catégorielle ou discrète
# - "code_agence": 150 codes → Catégorielle nominale
# - "age": 18-90 → Numérique
```

Question 25

Quel pré-traitement appliquer avant un algorithme basé sur les distances (KNN, K-Means)?

A) Aucun
B) Standardisation de toutes les variables numériques
C) One-Hot uniquement
D) Log transformation uniquement

Réponse

B) Standardisation de toutes les variables numériques

Les algorithmes basés sur les distances sont sensibles à l'échelle des variables:

```
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans

# Sans standardisation: le revenu (millions) dominerait l'âge (dizaines)
# Avec standardisation: toutes les variables ont le même poids

# Pipeline complet
numeric_cols = ['age', 'revenu', 'anciennete', 'nb_produits']

scaler = StandardScaler()
X_scaled = scaler.fit_transform(df[numeric_cols])

# K-Means sur données standardisées
kmeans = KMeans(n_clusters=4, random_state=42)
df['cluster'] = kmeans.fit_predict(X_scaled)
```

Pour les catégorielles: One-Hot puis standardisation, ou utiliser des distances appropriées (Gower).

Résumé et Mnémotechniques

"NOIR" - Niveaux de Mesure

```
N - Nominal: Catégories sans ordre (couleur, ville)
O - Ordinal: Catégories avec ordre (S<M<L, risque)
```

```
I - Intervalle: Ordre + écarts égaux, pas de zéro absolu (°C)
R - Ratio: Ordre + écarts égaux + zéro absolu (revenus, âge)
```

Encoding par Type

```
Nominale → One-Hot (ou Target si haute cardinalité)
Ordinale → OrdinalEncoder avec ordre explicite
Binaire → 0/1 directement
Continue asymétrique → Log transform
```

Tests par Combinaison

```
Continue vs Continue → Pearson/Spearman
Continue vs Catégorielle (2) → t-test / Point-biserial
Continue vs Catégorielle (3+) → ANOVA
Catégorielle vs Catégorielle → Chi-carré
```

Score et Auto-évaluation

Score	Niveau	Recommandation
23-25	Expert	Prêt pour l'examen
18-22	Avancé	Réviser les points faibles
13-17	Intermédiaire	Revoir le document complet
< 13	Débutant	Étude approfondie nécessaire

Test préparé pour l'examen Data Analyst - UniBank Haiti Thème: Types de Variables - Maîtriser la nature des données