Test EDA / Data Wrangling - Test 1

Sujet: Exploratory Data Analysis et Data Wrangling
Niveau: Intermédiaire
Nombre de questions: 25

Questions et Réponses

Q1. Qu'est-ce que l'EDA et quel est son objectif principal?

R1. L'**Exploratory Data Analysis (EDA)** est une approche d'analyse de données visant à: - **Comprendre** la structure et les caractéristiques des données - **Détecter** les anomalies, outliers et erreurs - **Identifier** les patterns et relations - **Formuler** des hypothèses pour analyse approfondie - **Préparer** les données pour la modélisation

C'est une étape **préliminaire et itérative** avant toute analyse statistique ou ML.

Q2. Quelles sont les premières commandes Pandas à exécuter sur un nouveau dataset?

R2.

```
import pandas as pd

df = pd.read_csv('data.csv')

# 1. Dimensions
df.shape  # (lignes, colonnes)

# 2. Aperçu
df.head()
df.tail()

# 3. Types et mémoire
df.info()

# 4. Statistiques descriptives
df.describe()  # numériques
df.describe(include='object')  # catégorielles

# 5. Valeurs manquantes
df.isnull().sum()

# 6. Doublons
df.duplicated().sum()
```

Q3. Comment détecter les valeurs manquantes et quels types existe-t-il?

R3. Détection:

```
df.isnull().sum()  # Compte par colonne
df.isnull().mean() * 100  # Pourcentage
```

Types de valeurs manquantes: - **MCAR (Missing Completely At Random):** Aléatoire, pas de pattern - **MAR (Missing At Random):** Dépend d'autres variables observées - **MNAR (Missing Not At Random):** Dépend de la valeur manquante elle-même

Impact: Le type détermine la stratégie d'imputation.

Q4. Quelles sont les stratégies d'imputation des valeurs manquantes?

R4. | Stratégie | Quand utiliser | |———-|———-| | **Suppression** | Peu de manquants (<5%), MCAR | | **Moyenne** | Variable normale, MCAR | | **Médiane** | Variable asymétrique, outliers | | **Mode** | Variables catégorielles | | **Imputation par groupe** | MAR (ex: moyenne par segment) | | **Forward/Backward fill** | Séries temporelles | | **KNN Imputation** | Relations entre variables | | **Indicateur** | Créer colonne "is_missing" |

Q5. Comment détecter les outliers avec la méthode IQR?

R5.

```
def detect_outliers_iqr(df, column):
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1

    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    outliers = df[(df[column] < lower_bound) | (df[column] > upper_bound)]
    return outliers

# Exemple
outliers = detect_outliers_iqr(df, 'montant')
print(f"Nombre d'outliers: {len(outliers)}")
```

Q6. Quelle est la différence entre la méthode IQR et Z-score pour détecter les outliers?

R6. | IQR | Z-score | |—-|———| | Basé sur quartiles | Basé sur moyenne et écart-type | | **Robuste** aux outliers | Sensible aux outliers | | Seuil: 1.5×IQR | Seuil: généralement |z| > 3 | | Meilleur pour distributions asymétriques | Suppose normalité |

Recommandation: IQR pour données bancaires (souvent asymétriques).

Q7. Comment traiter les outliers?

R7. Options: 1. **Capping/Winsorization:** Remplacer par les bornes

```
df['montant'] = df['montant'].clip(lower, upper)
```

2. **Transformation log:** Réduit l'impact

```
df['log_montant'] = np.log1p(df['montant'])
```

3. **Suppression:** Si erreurs de données confirmées
4. **Segmentation:** Analyse séparée des outliers
5. **Modèles robustes:** Utiliser médiane, régression robuste

---

Q8. Comment identifier les doublons et les traiter?

R8.

```
# Identifier
df.duplicated().sum()  # Lignes complètement dupliquées
df.duplicated(subset=['client_id', 'date']).sum()  # Sur colonnes clés

# Visualiser
df[df.duplicated(keep=False)]

# Supprimer
df_clean = df.drop_duplicates()
df_clean = df.drop_duplicates(subset=['client_id', 'date'], keep='last')
```

Attention: Toujours comprendre pourquoi il y a des doublons avant de supprimer.

Q9. Comment convertir les types de données correctement?

R9.

```
# Numérique
df['montant'] = pd.to_numeric(df['montant'], errors='coerce')

# Date
df['date'] = pd.to_datetime(df['date'], format='%Y-%m-%d')

# Catégorielle (économise mémoire)
df['secteur'] = df['secteur'].astype('category')

# Boolean
df['actif'] = df['actif'].map({'Oui': True, 'Non': False})

# String
df['code'] = df['code'].astype(str)
```

Q10. Qu'est-ce que le feature engineering et donnez des exemples bancaires?

R10. Le **feature engineering** est la création de nouvelles variables à partir des données existantes.

Exemples bancaires:

```
# Ratio dette/revenu
df['DTI'] = df['dette'] / df['revenu']

# Âge du compte
df['anciennete'] = (pd.Timestamp.now() - df['date_ouverture']).dt.days

# Indicateur de risque
df['high_risk'] = (df['score_credit'] < 500).astype(int)

# Agrégations
df['nb_tx_30j'] = df.groupby('client_id')['tx_id'].transform('count')

# Extraction date
```

```
df['mois'] = df['date'].dt.month
df['jour_semaine'] = df['date'].dt.dayofweek
```

Q11. Comment analyser les corrélations entre variables?

R11.

```
# Matrice de corrélation
corr_matrix = df.select_dtypes(include=[np.number]).corr()

# Visualisation
import seaborn as sns
plt.figure(figsize=(12, 10))
sns.heatmap(corr_matrix, annot=True, cmap='RdBu_r', center=0)

# Corrélations avec la cible
corr_with_target = corr_matrix['defaut'].sort_values(ascending=False)
```

Interprétation: - |r| > 0.7: Forte corrélation - 0.3 < |r| < 0.7: Corrélation modérée - |r| < 0.3: Faible corrélation

---

Q12. Comment standardiser et normaliser les données?

R12.

```
from sklearn.preprocessing import StandardScaler, MinMaxScaler

# Standardisation (Z-score): moyenne=0, std=1
scaler = StandardScaler()
df_scaled = scaler.fit_transform(df[['var1', 'var2']])

# Normalisation (Min-Max): [0, 1]
scaler = MinMaxScaler()
df_normalized = scaler.fit_transform(df[['var1', 'var2']])
```

Quand utiliser: - StandardScaler: Algorithmes sensibles à l'échelle (régression, PCA) - MinMaxScaler: Réseaux de neurones, visualisations

---

Q13. Comment gérer les variables catégorielles?

R13.

```
# Label Encoding (ordinal)
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['secteur_encoded'] = le.fit_transform(df['secteur'])

# One-Hot Encoding (nominal)
df_dummies = pd.get_dummies(df, columns=['secteur'], drop_first=True)

# Target Encoding (avec cible)
mean_by_cat = df.groupby('secteur')['defaut'].mean()
df['secteur_target'] = df['secteur'].map(mean_by_cat)
```

Q14. Comment identifier les variables importantes pour prédire le défaut?

R14. Méthodes: 1. **Corrélation:** avec la variable cible 2. **Chi-carré:** pour variables catégorielles 3. **Mutual Information:** capture relations non-linéaires 4. **Feature Importance:** avec Random Forest

```
from sklearn.feature_selection import mutual_info_classif
mi_scores = mutual_info_classif(X, y)
mi_df = pd.DataFrame({'feature': X.columns, 'MI': mi_scores})
mi_df.sort_values('MI', ascending=False)
```

Q15. Qu'est-ce que le data profiling automatique?

R15. Le **data profiling** génère automatiquement un rapport complet sur les données.

```
# Avec ydata-profiling (ex pandas-profiling)
from ydata_profiling import ProfileReport
profile = ProfileReport(df, title='EDA Report')
profile.to_file('report.html')
```

Contenu: - Statistiques par variable - Distributions - Corrélations - Valeurs manquantes - Alertes sur qualité des données

---

Q16. Comment fusionner plusieurs tables de données?

R16.

```
# INNER JOIN
df_merged = pd.merge(df1, df2, on='client_id', how='inner')

# LEFT JOIN
df_merged = pd.merge(df1, df2, on='client_id', how='left')

# Multiple keys
df_merged = pd.merge(df1, df2, on=['client_id', 'date'], how='left')

# Concaténation verticale
df_concat = pd.concat([df1, df2], axis=0, ignore_index=True)
```

Q17. Comment créer des agrégations par groupe?

R17.

```
# Agrégation simple
df.groupby('agence')['montant'].mean()

# Agrégations multiples
df.groupby('agence').agg({
    'montant': ['sum', 'mean', 'count'],
    'defaut': 'mean'
})

# Transform (garde la taille originale)
df['montant_agence'] = df.groupby('agence')['montant'].transform('sum')
```

```python
# Pivot table
pd.pivot_table(df, values='montant', index='agence', columns='produit', aggfunc='sum')
```

Q18. Comment gérer les données de séries temporelles?

R18.

```python
# Index temporel
df = df.set_index('date')

# Resampling
df.resample('M').sum()  # Mensuel
df.resample('W').mean()  # Hebdomadaire

# Rolling window
df['MA_7'] = df['montant'].rolling(window=7).mean()

# Lag features
df['montant_lag1'] = df['montant'].shift(1)

# Différence
df['diff'] = df['montant'].diff()
```

Q19. Comment détecter les anomalies dans les données bancaires?

R19. Méthodes: 1. **Règles métier:** Montant > seuil, transactions nocturnes 2. **Statistique:** Z-score, IQR sur historique client 3. **Machine Learning:**

```python
from sklearn.ensemble import IsolationForest
iso = IsolationForest(contamination=0.01)
df['anomaly'] = iso.fit_predict(df[['montant', 'nb_tx']])
```

Q20. Comment valider la qualité des données?

R20. Checklist de validation:

```python
def data_quality_report(df):
    report = {
        'rows': len(df),
        'columns': len(df.columns),
        'missing_pct': df.isnull().mean().mean() * 100,
        'duplicates': df.duplicated().sum(),
        'memory_mb': df.memory_usage(deep=True).sum() / 1e6
    }
    return report

# Règles de validation
assert df['montant'].min() >= 0, "Montants négatifs détectés"
assert df['date'].max() <= pd.Timestamp.now(), "Dates futures détectées"
assert df['taux'].between(0, 100).all(), "Taux hors limites"
```

Q21. Comment documenter les transformations de données?

R21. Bonnes pratiques: 1. **Notebook structuré:** Sections claires (chargement, nettoyage, transformation) 2. **Commentaires:** Expliquer le "pourquoi" 3. **Log des modifications:**

```
transformations_log = []
transformations_log.append({
    'step': 'Remove outliers',
    'column': 'montant',
    'method': 'IQR',
    'rows_removed': 150
})
```

4. **Versioning:** Git pour le code, DVC pour les données

Q22. Comment optimiser le chargement de grands fichiers?

R22.

```
# Lire par chunks
chunks = pd.read_csv('big_file.csv', chunksize=100000)
for chunk in chunks:
    process(chunk)

# Spécifier les types (économise mémoire)
dtypes = {'id': 'int32', 'montant': 'float32', 'categorie': 'category'}
df = pd.read_csv('file.csv', dtype=dtypes)

# Sélectionner les colonnes nécessaires
df = pd.read_csv('file.csv', usecols=['col1', 'col2'])

# Utiliser Parquet (plus efficace)
df.to_parquet('data.parquet')
df = pd.read_parquet('data.parquet')
```

Q23. Comment créer un pipeline de préparation des données?

R23.

```
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler, OneHotEncoder

numeric_features = ['montant', 'age']
categorical_features = ['secteur', 'type']

numeric_transformer = Pipeline([
    ('imputer', SimpleImputer(strategy='median')),
    ('scaler', StandardScaler())
])

categorical_transformer = Pipeline([
    ('imputer', SimpleImputer(strategy='constant', fill_value='missing')),
    ('onehot', OneHotEncoder(handle_unknown='ignore'))
])
```

```python
preprocessor = ColumnTransformer([
    ('num', numeric_transformer, numeric_features),
    ('cat', categorical_transformer, categorical_features)
])

# Utilisation
X_processed = preprocessor.fit_transform(df)
```

Q24. Quels sont les principaux pièges de l'EDA?

R24. 1. **Data leakage:** Utiliser des infos futures pour prédire le passé 2. **Overfitting de l'EDA:** Trop ajuster les transformations au dataset 3. **Ignorer le contexte métier:** Traiter les données sans comprendre le domaine 4. **Confirmation bias:** Chercher uniquement ce qu'on veut trouver 5. **Correlation ≠ Causation:** Conclusions hâtives sur les relations 6. **Survivorship bias:** N'analyser que les clients actifs

---

Q25. Proposez un workflow EDA complet pour un dataset de prêts bancaires.

R25.

```
1. CHARGEMENT ET APERÇU
      shape, head, info, describe

2. QUALITÉ DES DONNÉES
      Valeurs manquantes → Stratégie d'imputation
      Doublons → Suppression justifiée
      Types → Conversion appropriée
      Incohérences → Règles métier

3. ANALYSE UNIVARIÉE
      Variables numériques → Histogrammes, stats
      Variables catégorielles → Fréquences, bar charts

4. ANALYSE BIVARIÉE
      Num vs Num → Scatter, corrélation
      Num vs Cat → Box plots, ANOVA
      Cat vs Cat → Contingence, Chi-carré

5. FEATURE ENGINEERING
      Ratios (DTI, LTV)
      Agrégations (comportement client)
      Indicateurs dérivés

6. PRÉPARATION MODÉLISATION
      Encoding
      Scaling
      Split train/test
```

Scoring

Score	Niveau
0-10	À améliorer
11-17	Intermédiaire
18-22	Avancé
23-25	Expert