Challenge

28 novembre 2025

Table des matières

1 Broadcasting

Comparaison Départementale

```
# Broadcasting pour  carts  d partementaux
dept_mean = df.groupby('dept')['salaire'].transform('mean')
ecarts = df['salaire'] - dept_mean  # Op ration vectoris e
```

Similarités Employés

```
# Matrice distances avec broadcasting
A = features[:, np.newaxis, :]  # (n,1,k)
B = features[np.newaxis, :, :]  # (1,n,k)
distances = np.sqrt(((A - B)**2).sum(axis=2))
```

Optimisation Budget

```
# Ajustement salarial vectoris
nouveaux_salaires = salaires * np.sqrt(performances/8)
nouveaux_salaires = np.minimum(nouveaux_salaires, salaires*1.15)
```

2 Einstein Summation

Similarités Départements

```
# Similarit  cosinus entre d partements
dot_product = np.einsum('ik,jk->ij', dept_skills, dept_skills)
norms = np.sqrt(np.einsum('ik,ik->i', dept_skills, dept_skills))
similarites = dot_product / np.outer(norms, norms)
```

Projection 2D

```
# PCA manuel avec einsum
covariance = np.einsum('ij,ik->jk', X_norm, X_norm)
valeurs_propres, vecteurs_propres = np.linalg.eigh(covariance)
projection = np.einsum('ij,jk->ik', X_norm, vecteurs_propres[:, -2:])
```

Centralité Réseau

```
# Centralit  des degr s
centralite_degres = np.einsum('ij->i', matrice_adjacence > 0)

# Centralit  de proximit
distances = 1 / (1 + matrice_adjacence)
centralite_proximite = (n-1) / np.einsum('ij->i', distances)
```

3 UFuncs Personnalisées

Score Performance

```
@np.vectorize
def score_performance(qualite, efficacite, innovation):
    base = 0.4*qualite + 0.3*efficacite + 0.3*innovation
    return 10 / (1 + np.exp(-(base-7.5)))  # Sigmo de 0-10
```

Risque Turnover

```
@np.vectorize
def risque_turnover(anciennete, satisfaction, performance):
    risque_anciennete = np.exp(-anciennete/4)
    risque_satisfaction = (10 - satisfaction) / 10
    return 0.4*risque_anciennete + 0.6*risque_satisfaction
```

Calcul Bonus

```
@np.vectorize
def calcul_bonus(performance, salaire, grade):
    multiplicateurs = {1:0.05, 2:0.08, 3:0.12, 4:0.15, 5:0.20}
    base = multiplicateurs.get(grade, 0.1)
    ajustement = np.where(performance>=9, 1.5, 1.0)
    return salaire * base * ajustement
```

4 MultiIndex

Création Hiérarchie

```
# MultiIndex    3 niveaux
index = pd.MultiIndex.from_arrays([
    df['region'], df['departement'], df['grade']
], names=['region', 'dept', 'grade'])

df_multi = pd.DataFrame({'salaire': df['salaire']}, index=index)
```

Sélection Avancée

```
# S lection cross-section
europe_tech = df_multi.xs('Europe', level='region')
seniors = df_multi.xs('Senior', level='grade', drop_level=False)

# IndexSlice pour s lections complexes
idx = pd.IndexSlice
selection = df_multi.loc[idx['Europe', 'Technologie', :], :]
```

Anomalies Hiérarchiques

```
# D tection outliers par groupe
def detect_anomalies(groupe):
    z_scores = np.abs((groupe - groupe.mean()) / groupe.std())
    return z_scores > 2.5

anomalies = df.groupby(['region', 'dept'])['salaire'].transform(
   detect_anomalies)
```

5 Agrégations Avancées

Agrégations Multiples

```
# Agr gations avec dictionnaire
agregations = {
    'salaire': ['mean', 'median', 'std'],
    'performance': ['mean', 'min', 'max'],
    'satisfaction': lambda x: np.percentile(x, 75)
}
resultats = df.groupby('departement').agg(agregations)
```

Transformation vs Agrégation

```
# Transformation: m me forme
df['salaire_normalise'] = df.groupby('grade')['salaire'].transform(
    lambda x: (x - x.mean()) / x.std()
)

# Agr gation: forme r duite
moyennes_grade = df.groupby('grade')['salaire'].mean()
```

Window Functions

```
# Rang dans le d partement
df['rang_salaire'] = df.groupby('departement')['salaire'].rank(ascending=
    False)

# Diff rence vs m diane du grade
df['ecart_median'] = df['salaire'] - df.groupby('grade')['salaire'].
    transform('median')
```

6 Time Series Avancées

Resampling

```
# Resampling mensuel -> trimestriel
trimestriel = df.resample('Q').agg({
    'performance': 'mean',
    'salaire': 'sum',
    'effectif': 'count'
})
```

Rolling Statistics

```
# Moyennes mobiles
df['ma_3m'] = df['performance'].rolling(window=3).mean()
df['ma_6m'] = df['performance'].rolling(window=6).mean()

# Volatilit  mobile
df['volatilite_3m'] = df['performance'].rolling(window=3).std()
```

Saisonnalité

```
# D tection pattern saisonnier
from statsmodels.tsa.seasonal import seasonal_decompose

result = seasonal_decompose(df['performance'], model='additive', period
    =12)
saisonnalite = result.seasonal
tendance = result.trend
```

7 Optimisation

Optimisation Mémoire

```
# Downcast des types num riques
df['salaire'] = pd.to_numeric(df['salaire'], downcast='float')
df['age'] = pd.to_numeric(df['age'], downcast='integer')

# Cat gories pour strings r p titifs
df['departement'] = df['departement'].astype('category')
```

Vectorisation vs Apply

```
# VECTORIS  (rapide)
df['score'] = df['performance'] * 10 + df['satisfaction'] * 5

# APPLY (lent)
df['score'] = df.apply(lambda row: row['performance']*10 + row['
    satisfaction']*5, axis=1)
```

Parallel Processing

```
from multiprocessing import Pool

def process_department(data):
    return data['salaire'].mean()

with Pool(4) as pool:
    results = pool.map(process_department, [group for _, group in df.
        groupby('departement')])
```

8 Domaines d'Application

Ressources Humaines

- **Broadcasting** : Comparaisons salariales, benchmarks
- **MultiIndex** : Structure organisationnelle hiérarchique
- **UFuncs** : Scoring performance, risque turnover
- **Agrégations** : Métriques par département/grade

Analyse Financière

- **Einsum** : Corrélations, calculs de covariance
- **Time Series** : Tendances, saisonnalité revenus
- **Rolling Stats** : Volatilité, moyennes mobiles
- **Optimisation** : Calculs sur grands volumes

Analyse Réseaux

- **Broadcasting** : Matrices de similarités
- **Einsum** : Centralité, mesures réseau
- **MultiIndex** : Structures complexes
- **Agrégations** : Métriques de groupe

Data Engineering

- **Optimisation** : Gestion mémoire, performance
- **Vectorisation** : Transformation données
- **UFuncs** : Nettoyage, feature engineering
- **Time Series** : Agrégation temporelle

Instructions

Répondez aux 30 questions en utilisant tous les concepts avancés vus dans la documentation. Optimisez pour la performance et la mémoire.

9 Dataset

- `advanced_employees.csv` : 100,000 employés
- `employee_timeseries.csv` : Données mensuelles 2024
- Structure hiérarchique : Région → Département → Sous-département → Team

10 Questions

10.1 MultiIndex & Indexation Avancée

Q1 : Structure Hiérarchique Complexe

- Créez un MultiIndex : Région → Département → Grade → Année d'embauche
- Calculez les statistiques salariales à chaque niveau
- Utilisez `xs()` pour extraire les Managers d'Europe embauchés après 2020

Q2 : Sélection Multi-Niveaux

- Sélectionnez avec `IndexSlice` tous les employés :
  - Département "Technologie" OU "Finance"
  - Grades "Senior" ou "Lead"
  - Performance > 7.5
- Mesurez le temps de sélection

Q3 : Agrégation Hiérarchique Dynamique

- Pour chaque niveau hiérarchique, calculez automatiquement :
  - Effectifs et turnover
  - Écart-type des salaires
  - Corrélation performance-salaire

10.2 Broadcasting Avancé

Q4 : Matrice de Similarités

- Créez une matrice 100k×100k de similarités entre employés
- Utilisez broadcasting pour calculer les distances euclidiennes
- Optimisez la mémoire avec des calculs en chunks

Q5 : Normalisation Multi-Dimensionnelle

- Normalisez simultanément salaire, performance et satisfaction
- Utilisez broadcasting pour appliquer des weights différents

$$\text{Score} = 0.5 \times Z_{\text{salaire}} + 0.3 \times Z_{\text{performance}} + 0.2 \times Z_{\text{satisfaction}}$$

Q6 : Benchmarks Departmentaux

- Pour chaque employé, calculez son écart aux benchmarks de son département
- Utilisez broadcasting pour comparer aux moyennes départementales
- Identifiez les over/under performers

10.3 Einstein Summation

Q7 : Contraction de Tensors

- Créez un tensor département × grade × compétence
- Utilisez einsum pour calculer les similarités entre départements
- Identifiez les départements avec profils similaires

Q8 : Projections Multi-Dimensionnelles

- Projetez les données employés dans un espace 2D
- Utilisez einsum pour le produit matriciel de projection

$$\text{Projection} = \text{Données} \times \text{Matrice de Projection}$$

Q9 : Calculs de Covariance

- Calculez la matrice de covariance entre toutes les features
- Utilisez einsum pour une implémentation optimisée
- Comparez avec `np.cov()` en termes de performance

10.4 UFuncs & Vectorisation

Q10 : UFunc ROI Formation Avancée

- Créez une ufunc qui calcule :

$$\text{ROI} = \frac{\Delta\text{Performance} \times \text{Salaire}}{\text{Coût Formation} \times \text{Durée}}$$

- Supportez le broadcasting pour multiples scénarios
- Mesurez l'accélération vs Python pur

Q11 : Vectorisation de Calculs Complexes

- Calculez pour chaque employé son "market value" basé sur :
  - Salaire actuel
  - Performance historique
  - Compétences rares
  - Ancienneté
- Utilisez uniquement des opérations vectorisées

Q12 : Simulations Monte Carlo

- Simulez 10,000 scénarios de croissance salariale
- Utilisez vectorisation pour tous les scénarios simultanément
- Calculez les percentiles de croissance

10.5 Array Structurés & Mémoire

Q13 : Optimisation Mémoire avec Structured Arrays

- Convertissez le DataFrame en structured array NumPy
- Optimisez les types de données
- Mesurez le gain de mémoire et performance

Q14 : Calculs sur Structured Arrays

- Implémentez les agrégations directement sur le structured array
- Utilisez les opérations NumPy natives
- Comparez les performances avec Pandas

Q15 : Memory Mapping pour Gros Datasets

- Utilisez `np.memmap` pour charger les données
- Implémentez des calculs partiels sans tout charger en RAM
- Mesurez l'impact sur l'utilisation mémoire

10.6 Agrégations Avancées

Q16 : Agrégations Conditionnelles

- Calculez par département :
  - Salaire moyen des hauts performeurs (performance > 8)
  - Turnover risk des bas salaires (salaire $<$ département médian)
  - Satisfaction des employés récents ($<$2 ans)

Q17 : Window Functions Multi-Niveaux

- Pour chaque employé, calculez :
  - Rang salarial dans son département ET grade
  - Performance relative à sa cohorte d'embauche
  - Growth rate vs année précédente

Q18 : Agrégations Recursives

- Calculez des métriques hiérarchiques :
  - Somme des salaires par branche hiérarchique
  - Performance moyenne des équipes (incluant les sous-équipes)
  - Depth de la structure organisationnelle

10.7 Time Series Avancées

Q19 : Resampling Multi-Fréquentiel

- Agrégez les données à multiple fréquences :
  - Quotidien, hebdomadaire, mensuel, trimestriel
  - Calculez les métriques à chaque fréquence
  - Analysez la perte d'information

Q20 : Rolling Windows Adaptatives

— Implémentez des fenêtres variables :
  — Basées sur les événements (promotions, formations)
  — Basées sur la volatilité des performances
  — Fenêtres exponentielles pondérées

Q21 : Analyse de Saisonnalité Multi-Période

— Détectez les cycles :
  — Saisonnier (annuel)
  — Mensuel (fin de mois)
  — Hebdomadaire (weekends)
  — Quotidien (heures de travail)

10.8 Performance & Optimisation

Q22 : Benchmark Complet des Méthodes

— Comparez pour 10 opérations différentes :
  — Pandas method chains
  — NumPy vectorization
  — apply() avec fonctions custom
  — np.vectorize()
  — Cython/Numba

Q23 : Optimisation Mémoire Automatique

— Créez une fonction qui optimise automatiquement un DataFrame :
  — Downcast des numériques
  — Conversion en categories
  — Suppression des colonnes inutiles
— Mesurez le gain sur le dataset complet

Q24 : Parallel Processing Avancé

— Parallélisez le calcul des corrélations entre toutes paires de features
— Utilisez `joblib` ou `dask`
— Optimisez le nombre de workers

10.9 Intégration & Applications

Q25 : Système de Recommandation Interne

— Recommandez pour chaque employé :
  — Formations basées sur les compétences similaires
  — Mentorat basé sur le parcours similaire
  — Projets basés sur les intérêts et compétences
— Utilisez broadcasting + einsum pour les similarités

Q26 : Detection d'Anomalies Multi-Dimensionnelle

— Identifiez les employés atypiques sur multiple dimensions :

- Performance vs salaire
- Ancienneté vs grade
- Compétences vs département

— Utilisez l'algorithme Isolation Forest vectorisé

Q27 : Forecasting de Masse Salariale

— Prévoyez la masse salariale pour les 3 prochaines années :

- Croissance organique
- Turnover estimé
- Promotions planifiées

— Utilisez des time series models vectorisés

10.10 Questions Bonus

Q28 : Visualisation de Données Hiérarchiques

— Créez un sunburst plot interactif de l'organisation
— Utilisez Plotly avec données agrégées efficacement
— Optimisez le temps de rendu pour 100k+ points

Q29 : API de Requêtes Avancées

— Implémentez un système de requêtes type-SQL sur le structured array
— Supportez WHERE, GROUP BY, HAVING
— Benchmarkez vs pandas.DataFrame.query()

Q30 : Compression et Archivage

— Compressez le dataset avec différentes méthodes :

- Parquet avec compression
- HDF5 avec blosc
- Feather format

— Mesurez ratio compression / vitesse accès

Grille d'Évaluation

Catégorie	Poids	Critères
Performance	30%	Temps d'exécution, usage mémoire
Exactitude	25%	Résultats corrects, edge cases
Optimisation	20%	Utilisation des concepts avancés
Qualité du code	15%	Lisibilité, commentaires, structure
Innovation	10%	Solutions créatives, approches originales