

Introduction à l'analyse exploratoire

Titre du cours : Analyse exploratoire des données

Code officiel : 420-A55-BB

Professeur : Dr Komi SODOKE

Plan

I- Introduction

II- Définition de l'analyse exploratoire des données

III- Objectifs de l'analyse dans le contexte de l'apprentissage machine

- A. Comprendre la distribution des données
- B. Détection et gestion des valeurs aberrantes et des données manquantes
- C. Visualisation des relations entre les variables
- D. Détection des biais et anomalies
- E. Guider la préparation des données

IV- Importance de la qualité des données dans le développement d'une application IA

V- Conclusion

Introduction

L'analyse exploratoire des données (Exploratory Data Analysis - EDA) est une étape fondamentale dans tout projet de Machine Learning. Elle permet de mieux comprendre les données, d'identifier des tendances, de détecter des anomalies et d'orienter les choix méthodologiques avant de construire un modèle prédictif.

L'EDA repose sur l'utilisation de méthodes statistiques et de visualisations graphiques pour explorer et comprendre les données. Elle permet de répondre à des questions essentielles : Y a-t-il des valeurs aberrantes ? Les données sont-elles équilibrées ou déséquilibrées ? Existe-t-il des corrélations entre certaines variables ? La qualité des données est-elle suffisante pour entraîner un modèle efficace ? En abordant ces questions de manière systématique, l'EDA joue un rôle clé dans la réussite d'un projet d'analyse de données et de Machine Learning.

II-Définition de l'analyse exploratoire des données

L'analyse exploratoire des données (EDA) est une approche qui vise à examiner, nettoyer et transformer les données brutes afin de découvrir des schémas sous-jacents. Elle repose sur des techniques statistiques et des visualisations graphiques pour résumer les principales caractéristiques des données.

Exemple : Supposons que nous ayons un jeu de données sur les ventes d'une boutique en ligne. Une analyse exploratoire pourrait inclure l'examen de la distribution des prix des produits, l'identification des jours où les ventes sont les plus élevées, et la détection de produits qui ont des prix anormalement bas ou élevés.



Origine de l'EDA

L'analyse Exploratoire de données (EDA) a été introduite par John W. Tukey en 1977 dans son livre "Exploratory Data Analysis". Tukey, un statisticien américain, a mis l'accent sur l'importance de l'exploration visuelle et de la compréhension intuitive des données avant d'appliquer des modèles statistiques formels.

Son approche contrastait avec les méthodes statistiques traditionnelles en mettant l'accent sur :

- ✓ **La visualisation des données (histogrammes, boxplots, scatterplots, etc.).**
- ✓ **L'identification des tendances et anomalies.**
- ✓ **Une exploration interactive des données avant toute modélisation.**

❖ Rôle de l'Exploratory Data Analysis (EDA)

L'Exploratory Data Analysis (EDA), ou **Analyse Exploratoire des Données** permet d'examiner un jeu de données avant de le modéliser, afin d'en extraire des informations utiles, d'identifier d'éventuels problèmes et de préparer les données pour une meilleure exploitation.

1. Compréhension des Données

L'EDA permet aux analystes de comprendre :

- La **provenance** des données (source, mode de collecte).
- La **structure des données** (nombre de lignes et colonnes, types de variables).
- La **signification des variables** (quelles informations elles représentent et comment elles peuvent être utilisées).

Exemple : Chargement et aperçu des données

```
import pandas as pd

# Chargement des données (exemple : dataset Titanic)
df = pd.read_csv("titanic.csv")

# Afficher les 5 premières lignes
print(df.head())

# Afficher des informations générales sur le dataset
print(df.info())

# Statistiques descriptives des variables numériques
print(df.describe())
```

Interpretation

- `df.head()` permet de voir un échantillon des données.
- `df.info()` donne la structure du dataset (nombre de lignes/colonnes, types de données, valeurs manquantes).
- `df.describe()` fournit des statistiques résumées (moyenne, min, max, etc.), utiles pour repérer des anomalies.

2. Identification des Données Incomplètes ou Erronées

Les jeux de données réels contiennent souvent des valeurs manquantes, doublons ou valeurs aberrantes qui peuvent fausser les résultats.

Exemple : Détection des valeurs manquantes et aberrantes

```
# Vérifier le nombre de valeurs manquantes par colonne
print(df.isnull().sum())

# Vérifier la présence de doublons
print(df.duplicated().sum())

# Visualisation des valeurs aberrantes avec une boîte à moustaches
import seaborn as sns
import matplotlib.pyplot as plt

sns.boxplot(x=df['Fare'])
plt.show()
```

Interprétation :

- `isnull().sum()` permet de repérer les colonnes contenant des valeurs manquantes.
- `duplicated().sum()` détecte les lignes en double, souvent inutiles.
- **Le boxplot** permet d'identifier les valeurs aberrantes (outliers), comme des prix de billets excessivement élevés.

3. Réduction de la Dimensionnalité

Dans certains cas, toutes les variables ne sont pas pertinentes. L'EDA aide à identifier les variables les plus importantes et à supprimer celles qui sont redondantes ou peu informatives.

Exemple : Analyse de la corrélation entre variables

```
import seaborn as sns
import matplotlib.pyplot as plt

# Matrice de corrélation
plt.figure(figsize=(10,6))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm', fmt=".2f")
plt.show()
```

Interprétation :

- Une **corrélacion élevée** entre deux variables signifie qu'on pourrait en supprimer une sans perte d'information.
- Exemple : si Age et Years_of_Experience sont fortement corrélés, l'un des deux peut être supprimé.

4. Visualisation des Données

L'EDA utilise des techniques de visualisation pour rendre les données plus compréhensibles et identifier des tendances ou relations.

Exemple : Histogramme et nuage de points

```
import seaborn as sns
import matplotlib.pyplot as plt

# Matrice de corrélation
plt.figure(figsize=(10,6))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm', fmt=".2f")
plt.show()
```

Interprétation :

- L'**histogramme** montre comment se répartit une variable (ex: la majorité des passagers ont entre 20 et 40 ans).
- Le **scatterplot** permet d'analyser les relations entre variables (ex: les passagers ayant payé des billets plus chers avaient-ils plus de chances de survivre ?).

5. Préparation des Données

Une fois les anomalies détectées, il est crucial de préparer les données pour la modélisation en :

✓ Remplaçant les valeurs manquantes.

✓ Normalisant les variables.

✓ Encodant les variables catégorielles.

Exemple : Nettoyage et transformation des données :

```
# Remplacement des valeurs manquantes par la médiane
df['Age'].fillna(df['Age'].median(), inplace=True)

# Suppression des colonnes inutiles
df.drop(['Cabin'], axis=1, inplace=True)

# Encodage des variables catégorielles (ex: Sexe)
df['Sex'] = df['Sex'].map({'male': 0, 'female': 1})

# Normalisation du prix du billet
df['Fare'] = (df['Fare'] - df['Fare'].mean()) / df['Fare'].std()
```

Interprétation :

- **Remplacement des valeurs manquantes** par la médiane évite de perdre des données importantes.
- **Suppression des colonnes inutiles** (ex: Cabin a trop de valeurs manquantes).
- **Encodage des variables catégorielles** (Sex devient numérique pour être exploitable en machine learning).
- **Normalisation des valeurs** pour les mettre à la même échelle.

❖ Différence entre EDA et analyse statistique classique

L'analyse exploratoire des données (EDA) et l'analyse statistique classique partagent certaines méthodes, mais elles diffèrent dans leur approche et leurs objectifs.

-Objectif principal :

- L'EDA vise à explorer et comprendre les données sans hypothèse préalable, en mettant l'accent sur la visualisation et la détection d'anomalies.
- L'analyse statistique classique repose sur des hypothèses préétablies et cherche à tester des relations spécifiques entre les variables à l'aide de modèles mathématiques.

-Approche méthodologique :

- L'EDA est itérative et interactive, mettant l'accent sur les graphiques, les statistiques descriptives et l'observation des tendances.
- L'analyse statistique classique est plus formelle, utilisant des tests d'hypothèses, des modèles de régression et des techniques inférentielles pour tirer des conclusions sur une population à partir d'un échantillon.

-Utilisation des visualisations :

- L'EDA fait un usage intensif des visualisations (histogrammes, boxplots, heatmaps, etc.) pour identifier rapidement des schémas dans les données.
- L'analyse statistique classique utilise souvent des mesures numériques (p-valeurs, coefficients de corrélation) et se focalise moins sur les représentations graphiques.

-Flexibilité et découverte :

- L'EDA est une phase exploratoire flexible où le but est de générer des idées et d'identifier des tendances inattendues.
- L'analyse statistique classique suit une approche plus rigide avec des tests précis permettant de confirmer ou infirmer une hypothèse formulée au préalable.

Exemple : Supposons que nous souhaitons analyser l'impact du temps d'étude sur les performances scolaires. Avec l'EDA, nous utiliserions des histogrammes et des nuages de points pour visualiser la relation entre ces deux variables. Avec l'analyse statistique classique, nous pourrions effectuer une régression linéaire et un test de signification statistique pour déterminer si cette relation est significative.

❖ Outils et bibliothèques utilisés

L'EDA repose sur plusieurs outils et bibliothèques qui facilitent la manipulation et la visualisation des données. Voici quelques-uns des plus couramment utilisés :

- **Pandas** (`pandas`) :
 - Permet de charger, manipuler et analyser des jeux de données tabulaires.
 - Exemple : `df.describe()` pour obtenir un résumé statistique des colonnes numériques.
- **NumPy** (`numpy`) :
 - Fournit des outils pour manipuler des tableaux numériques et effectuer des calculs mathématiques.
 - Exemple : `np.mean(data)` pour calculer la moyenne d'un tableau.
- **Matplotlib** (`matplotlib`) :
 - Bibliothèque de visualisation permettant de créer des graphiques statiques et interactifs.
 - Exemple : `plt.hist(df['age'])` pour afficher un histogramme des âges.

- **Seaborn** (`seaborn`) :
 - Extension de Matplotlib qui facilite la création de graphiques statistiques.
 - Exemple : `sns.boxplot(x='category', y='price', data=df)` pour visualiser la distribution des prix par catégorie.
- **Plotly** (`plotly`) :
 - Permet de créer des visualisations interactives, notamment des graphiques 3D et des tableaux de bord.
 - Exemple : `px.scatter(df, x='age', y='salary')` pour afficher un nuage de points interactif.
- **SciPy** (`scipy`) :
 - Fournit des outils avancés pour les statistiques, l'optimisation et le traitement du signal.
 - Exemple : `scipy.stats.pearsonr(x, y)` pour calculer le coefficient de corrélation entre deux variables.
- **Statsmodels** (`statsmodels`) :
 - Utilisé pour des analyses statistiques détaillées, notamment des tests d'hypothèses et des modèles de régression.
 - Exemple : `sm.OLS(y, X).fit()` pour ajuster un modèle de régression linéaire.

III- Objectifs de l'analyse dans le contexte de l'apprentissage machine

A. Comprendre la distribution des données

L'objectif est d'analyser la répartition des valeurs de chaque variable afin d'identifier d'éventuelles asymétries, tendances ou valeurs extrêmes.

Méthodes utilisées

- **Statistiques descriptives** : Moyenne, médiane, variance, écart-type, quartiles.
- **Visualisation des distributions** : Histogrammes, diagrammes en boîte (boxplots), courbes de densité (KDE - Kernel Density Estimation).
- **Analyse de la symétrie et de la dispersion** : Coefficients de skewness et kurtosis pour identifier une asymétrie ou une distribution aplatie.

Exemple

Dans une base de données contenant les salaires des employés d'une entreprise, un histogramme pourrait révéler une distribution asymétrique avec une longue queue à droite, indiquant que quelques employés gagnent beaucoup plus que la majorité.

B. Détecter les valeurs manquantes et aberrantes

1. Détection des Valeurs Manquantes

Identification : Utiliser `isnull()` et `sum()` pour compter les valeurs manquantes.

```
import pandas as pd

# Exemple de DataFrame
df = pd.DataFrame({
    'A': [1, 2, None, 4],
    'B': [None, 2, 3, 4],
    'C': [1, 2, 3, None]
})

# Compter les valeurs manquantes
missing_values = df.isnull().sum()
print(missing_values)
```

B. Détecter les valeurs manquantes et aberrantes

2. Gestion des Valeurs Manquantes

Imputation : Remplacer les valeurs manquantes par la moyenne, la médiane ou une technique avancée.

```
# Remplacer par la moyenne
df_mean_imputed = df.fillna(df.mean())

# Remplacer par la médiane
df_median_imputed = df.fillna(df.median())

# Utilisation de KNN Imputer (nécessite sklearn)
from sklearn.impute import KNNImputer

imputer = KNNImputer(n_neighbors=2)
df_knn_imputed = pd.DataFrame(imputer.fit_transform(df), columns=df.columns)

print(df_mean_imputed)
print(df_median_imputed)
print(df_knn_imputed)
```

B. Détecter les valeurs manquantes et aberrantes

3. Détection des Valeurs Aberrantes

Identification : Utiliser la méthode des quartiles (IQR) et le Z-score.

```
import numpy as np

# Méthode des quartiles (IQR)
Q1 = df.quantile(0.25)
Q3 = df.quantile(0.75)
IQR = Q3 - Q1

outliers_iqr = ((df < (Q1 - 1.5 * IQR)) | (df > (Q3 + 1.5 * IQR)))
print(outliers_iqr)

# Z-score
from scipy import stats

z_scores = np.abs(stats.zscore(df, nan_policy='omit'))
outliers_z = (z_scores > 3)
print(outliers_z)
```

B. Détecter les valeurs manquantes et aberrantes

4. Gestion des Valeurs Aberrantes

Suppression, Transformation, Imputation, et Segmentation

Suppression

```
df_no_outliers = df[~((df < (Q1 - 1.5 * IQR)) | (df > (Q3 + 1.5 * IQR))).any(axis=1)]
```

Transformation

```
df_log_transformed = np.log(df.replace(0, np.nan)) # Remplacer les 0 par NaN pour éviter des erreurs de log
```

Remplacer les valeurs aberrantes par la moyenne

```
df_cleaned = df.copy()
df_cleaned[outliers_iqr] = df.mean()
```

```
print(df_no_outliers)
```

```
print(df_log_transformed)
```

```
print(df_cleaned)
```

C. Visualisation des Relations entre les Variables

Pour analyser les corrélations et les interactions entre différentes variables, nous utilisons plusieurs méthodes permettant d'identifier des schémas qui influencent la variable cible.

Méthodes Utilisées :

1. Matrice de Corrélation :

- Calcul des coefficients de corrélation (Pearson, Spearman, Kendall) pour mesurer l'intensité des relations entre les variables.

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Exemple de DataFrame
df = pd.DataFrame({
    'Publicité': [100, 200, 300, 400],
    'Ventes': [20, 40, 60, 80],
    'Prix': [10, 20, 30, 40]
})

# Calcul de la matrice de corrélation
correlation_matrix = df.corr(method='pearson')
print(correlation_matrix)

# Visualisation de la matrice de corrélation
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.show()
```

C. Visualisation des Relations entre les Variables

2. Représentations Graphiques :

Utilisation de nuages de points (scatter plots) et de heatmaps de corrélation pour visualiser les relations entre les variables.

```
# Nuage de points entre la Publicité et les Ventes
sns.scatterplot(x='Publicité', y='Ventes', data=df)
plt.title('Relation entre la Publicité et les Ventes')
plt.show()
```

3. Pair Plots :

- Visualisation des relations entre plusieurs variables simultanément.

```
# Pair plot des variables
sns.pairplot(df)
plt.show()
```

C. Visualisation des Relations entre les Variables

4. Analyse en Composantes Principales (ACP) :

Réduction de dimensionnalité pour comprendre les relations entre les variables.

```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

# Standardisation des données
df_standardized = StandardScaler().fit_transform(df)

# ACP
pca = PCA(n_components=2)
principalComponents = pca.fit_transform(df_standardized)

# Visualisation des composantes principales
plt.figure()
plt.scatter(principalComponents[:, 0], principalComponents[:, 1])
plt.title('Analyse en Composantes Principales (ACP)')
plt.xlabel('Composante Principale 1')
plt.ylabel('Composante Principale 2')
plt.show()
```

D. Détection des biais et anomalies

S'assurer que les données sont représentatives et équilibrées pour éviter tout biais pendant l'entraînement des modèles prédictifs.

Méthodes Utilisées :

1. Analyse de l'Équilibre des Classes :

- Calculer les fréquences relatives des différentes classes dans la variable cible pour mesurer leur proportion.

```
import pandas as pd

# Exemple de DataFrame avec une variable cible
df = pd.DataFrame({
    'Classe': ['A', 'A', 'B', 'A', 'B', 'B', 'A', 'B', 'A', 'B', 'B']
})

# Calcul des fréquences relatives
class_frequencies = df['Classe'].value_counts(normalize=True)
print(class_frequencies)
```

D. Détection des biais et anomalies

2. Techniques de Rééquilibrage des Classes :

- Oversampling : Augmenter la proportion des classes minoritaires avec des techniques comme SMOTE (Synthetic Minority Over-sampling Technique).

```
from imblearn.over_sampling import SMOTE

# Exemple de DataFrame avec des caractéristiques et une variable cible
X = df[['caractéristique1', 'caractéristique2']]
y = df['Classe']

smote = SMOTE(random_state=42)
X_resampled, y_resampled = smote.fit_resample(X, y)

# Visualisation des classes rééchantillonnées
print(y_resampled.value_counts())
```

D. Détection des biais et anomalies

2. Techniques de Rééquilibrage des Classes :

- **Undersampling** : Réduire la proportion des classes majoritaires.

```
from imblearn.under_sampling import RandomUnderSampler

rus = RandomUnderSampler(random_state=42)
X_resampled, y_resampled = rus.fit_resample(X, y)

# Visualisation des classes rééchantillonnées
print(y_resampled.value_counts())
```

D. Détection des biais et anomalies

3. Identification des Biais :

Visualisation des Distributions Conditionnelles : Utiliser des graphiques pour détecter d'éventuelles disparités.

```
import seaborn as sns
import matplotlib.pyplot as plt

# Exemple de DataFrame avec des caractéristiques et une variable cible
df = pd.DataFrame({
    'caractéristique1': [1, 2, 3, 4, 5, 1, 2, 3, 4, 5],
    'caractéristique2': [5, 4, 3, 2, 1, 5, 4, 3, 2, 1],
    'Classe': ['A', 'A', 'B', 'B', 'A', 'B', 'B', 'A', 'A', 'B']
})

# Visualisation des distributions conditionnelles
sns.boxplot(x='Classe', y='caractéristique1', data=df)
plt.title('Distribution conditionnelle de la caractéristique 1 par classe')
plt.show()

sns.boxplot(x='Classe', y='caractéristique2', data=df)
plt.title('Distribution conditionnelle de la caractéristique 2 par classe')
plt.show()
```

Exemple : Supposons que nous analysons les classes dans un jeu de données sur les ventes. Si une classe "VIP" représente seulement 5% des données tandis que la classe "Regular" représente 95%, cela peut engendrer un biais lors de l'entraînement. En appliquant SMOTE, nous pouvons augmenter la proportion de la classe "VIP" pour qu'elle soit mieux représentée et équilibrer le jeu de données.

E. Guider la préparation des données

S'assurer que les données sont transformées et mises en forme de manière optimale pour l'entraînement des modèles.

Méthodes utilisées

- **Encodage des variables catégorielles** : One-hot encoding, Label Encoding.
- **Transformation des variables numériques** : Normalisation (MinMaxScaler), standardisation (StandardScaler).
- **Création de nouvelles variables** : Feature engineering basé sur des combinaisons de variables existantes.
- **Réduction de la dimensionnalité** : ACP, sélection de variables basées sur l'importance des features.

Importance de la qualité des données dans le développement d'une application IA

Garantir que les données utilisées pour l'entraînement des modèles sont fiables, cohérentes et représentatives afin d'éviter les erreurs et les biais dans les prédictions.

Principaux enjeux liés à la qualité des données

- **Exactitude** : Les données doivent refléter la réalité et ne pas contenir d'erreurs.
- **Complétude** : L'absence de valeurs critiques peut compromettre la performance du modèle.
- **Cohérence** : Les données doivent être homogènes à travers les différentes sources.
- **Représentativité** : L'échantillon utilisé pour l'apprentissage doit être fidèle à la population cible.
- **Absence de biais** : Un biais dans les données peut entraîner des prédictions injustes ou erronées.

Méthodes pour améliorer la qualité des données

- **Nettoyage des données** : Suppression des doublons, correction des incohérences.
- **Imputation des valeurs manquantes** : Remplacement par la moyenne, médiane, ou modèles prédictifs.
- **Détection et correction des valeurs aberrantes** : Utilisation de méthodes statistiques et algorithmiques.
- **Uniformisation des formats** : Standardisation des unités, normalisation des formats de date, harmonisation des valeurs catégorielles.
- **Échantillonnage équilibré** : Techniques de rééquilibrage pour éviter des biais dans la distribution des classes.

Exemple

Dans une application de reconnaissance faciale, si les données d'entraînement sont majoritairement issues d'un seul groupe ethnique, le modèle risque d'être moins performant pour d'autres groupes. Une analyse approfondie et un rééquilibrage des données sont donc essentiels pour garantir une IA équitable et performante.

Conclusion

L'analyse exploratoire des données (EDA) est une phase cruciale dans le développement des modèles d'apprentissage automatique. Elle aide à détecter et corriger les anomalies, à comprendre la structure des données et à garantir leur qualité avant l'entraînement des modèles. Maîtriser les techniques d'EDA permet d'améliorer les performances des modèles tout en réduisant les risques de biais et d'erreurs. En appliquant ces méthodes, les analystes et data scientists peuvent prendre des décisions informées et optimiser leurs résultats en apprentissage automatique.