Rapport TP6.

🧠 1. Le sujet du projet — En termes simples

Le sujet traite de la **prise de décision dans un véhicule autonome** à partir de plusieurs sources d'information incertaines :

- **Capteurs physiques** :
    - **Caméra** (vision)
    - **Radar** (détection distance/objet)
- **Contexte environnemental** :
    - **Météo** (Soleil, Pluie, Brouillard)
    - **Luminosité** (Jour, Nuit)

Le problème est :

> *Comment décider si un obstacle est réellement présent lorsque les capteurs peuvent se tromper et que l'environnement influence leur fiabilité ?*

Pour cela, on utilise un **réseau bayésien**, un modèle probabiliste de raisonnement sous incertitude.

📋 2. Ce que le TP demande exactement

Le TP demande de :

1. Construire un **réseau bayésien** (graphe + probabilités)
2. Justifier théoriquement :
    - pourquoi c'est un **DAG**

- comment on fait de la **fusion de capteurs**
- comment interpréter les **probabilités conditionnelles**

3. Comparer cette approche à une approche **heuristique (si-alors)**

🧩 3. Le problème à résoudre

Les capteurs peuvent être en conflit :

Situation	Caméra	Radar
Nuit + Brouillard	Peu fiable	Très fiable
Soleil + Jour	Très fiable	Fiable

Donc on ne peut pas faire :

> *"Si caméra voit obstacle → obstacle réel"*

On a besoin d'un système qui :

- **combine toutes les informations**
- gère **l'incertitude**
- produit une **décision probabiliste**

🛠️ 4. Comment on a résolu le problème

On a construit un **réseau bayésien** avec :

🎯 Variables

Variable	Rôle
Meteo	Condition météo
Luminosite	Jour / Nuit

Obstacle_Reel	Vérité du monde
Camera	Détection caméra
Radar	Détection radar

🧭 Dépendances

- Meteo → Camera
- Luminosite → Camera
- Obstacle_Reel → Camera
- Obstacle_Reel → Radar

Ce graphe est un **DAG** :
aucune boucle, uniquement des causes vers des effets.

📊 5. Fonctionnement du projet (pas à pas)

🧱 Étape 1 — Construction du modèle

Dans `model_builder.py` :

- Création du graphe bayésien
- Définition des **tables de probabilités conditionnelles (CPD)**

Exemple :
La CPD de la caméra a **12 colonnes** car :

3 Meteo×2 Luminosite´×2 Obstacle=123 \text{ Meteo} \times 2 \text{ Luminosité} \times 2 \text{ Obstacle} = 123 Meteo×2 Luminosite´×2 Obstacle=12

Chaque colonne représente une situation réelle différente.

🧱 Étape 2 — Inférence probabiliste

Dans `inference.py` :

On calcule :

$P(Obstacle\_Reel \mid Camera, Radar, Meteo, Luminosite)$

avec l'algorithme **Variable Elimination**.

🧪 Étape 3 — Simulation de scénarios

On lit plusieurs situations depuis `example_scenarios.csv` :

```
Meteo | Luminosite | Camera | Radar
```

Pour chaque scénario :

- on applique l'inférence
- on obtient la **probabilité finale d'obstacle réel**

📈 Étape 4 — Analyse & visualisation

Le programme génère :

- `results.csv` → résultats numériques
- `network.png` → graphe du réseau bayésien
- `probabilities.png` → graphique des probabilités

🧾 6. Ce que montrent les résultats

Exemples :

Situation	Interprétation
Brouillard + Nuit, Caméra=Présent, Radar=Absent → **16%**	La caméra est peu fiable, donc on ne fait pas confiance
Pluie + Nuit, Caméra=Absent, Radar=Présent → **82%**	Le radar domine la décision
Soleil + Jour, les deux = Présent → **98%**	Confiance très élevée

👉 Le système **résout automatiquement les conflits**.

———————————————————————————

🧩1️⃣`model_builder.py` — Construction du modèle

🎯 Rôle du fichier

Construire le **réseau bayésien** et définir toutes les **probabilités**.

🧠 Partie 1 — Création du graphe

```
model = DiscreteBayesianNetwork([
    ('Meteo', 'Camera'),
    ('Luminosite', 'Camera'),
    ('Obstacle_Reel', 'Camera'),
    ('Obstacle_Reel', 'Radar')
])
```

Explication :

- Chaque flèche signifie *“influence”*.
- Exemple : la caméra dépend de la météo, de la luminosité et de la réalité de l’obstacle.
- Le graphe est **orienté** et **sans cycle** ⇒ **DAG** ✔️

📊 Partie 2 — Tables de probabilités (CPDs)

Exemple :

```
cpd_meteo = TabularCPD('Meteo', 3, [[0.5], [0.3], [0.2]])
```

Signification :

> 50% Soleil, 30% Pluie, 20% Brouillard

🎥 CPD la plus importante : Camera

```
cpd_camera = TabularCPD(
    'Camera', 2,
    values=[ [... 12 valeurs ...], [... 12 valeurs ...] ],
    evidence=['Meteo', 'Luminosite', 'Obstacle_Reel'],
    evidence_card=[3, 2, 2]
)
```

Explication orale :

> La caméra dépend de **3 variables**.
> 3 × 2 × 2 = **12 situations différentes**.
> Chaque colonne décrit la fiabilité de la caméra dans une situation précise.

C'est là qu'on modélise **l'incertitude visuelle**.

🧪 Vérification

```
assert model.check_model()
```

> On vérifie mathématiquement que le réseau est cohérent.

🧮2️⃣ `inference.py` — Raisonnement

```
infer = VariableElimination(model)
result = infer.query(variables=['Obstacle_Reel'], evidence=evidence)
```

À dire :

> Ici on applique le raisonnement bayésien.
> Le système calcule :
> **P(Obstacle réel | toutes les observations)**

C'est le **cerveau décisionnel** du système.

🧪3️⃣`main.py` — Exécution du système

```
model = build_model()
df_results = compute_from_csv(model, 'data/example_scenarios.csv')
```

Rôle :

- Charger le modèle
- Lire des scénarios
- Calculer automatiquement les probabilités
- Sauvegarder les résultats
- Générer les graphiques

📈4️⃣`utils.py` — Visualisation

```
plot_bayesian_network(model)
plot_results(df_results)
```

Explication :

> On transforme le raisonnement abstrait en **images compréhensibles** :
>
> - le graphe du réseau
> - les probabilités par scénario