



IHEC
CARTHAGE



syStème expert



PROJET: INTELLIGENCE
ARTIFICIELLE





IHEC
CARTHAGE

NOTRE EQUIPE

A world map with three women's portraits overlaid. The portraits are circular and placed over the continents of North America, Europe, and Africa. Two small airplane icons are shown flying across the map.

Rania
Ben Brahim

Kmar
Hammami

Khawla
Chaabi

2BI4



SOMMAIRE



1. Introduction du projet
2. Liens avec les concepts du cours
3. Choix du thème
4. Architecture générale du système
5. Modèle de prise de décision
6. Description détaillée des fonctions implémentées
7. demonstration
8. Conclusion

INTRODUCTION DU PROJET

Dans le cadre de ce projet, nous avons développé un système de recommandation de destinations de voyage sous forme d'une application graphique interactive en Python. Le système collecte les préférences de l'utilisateur via 10 questions oui/non, utilise un modèle Random Forest pour prédire la destination idéale, et affiche les résultats avec une interface moderne (Tkinter + effets visuels).





LIENS AVEC LES CONCEPTS DU COURS

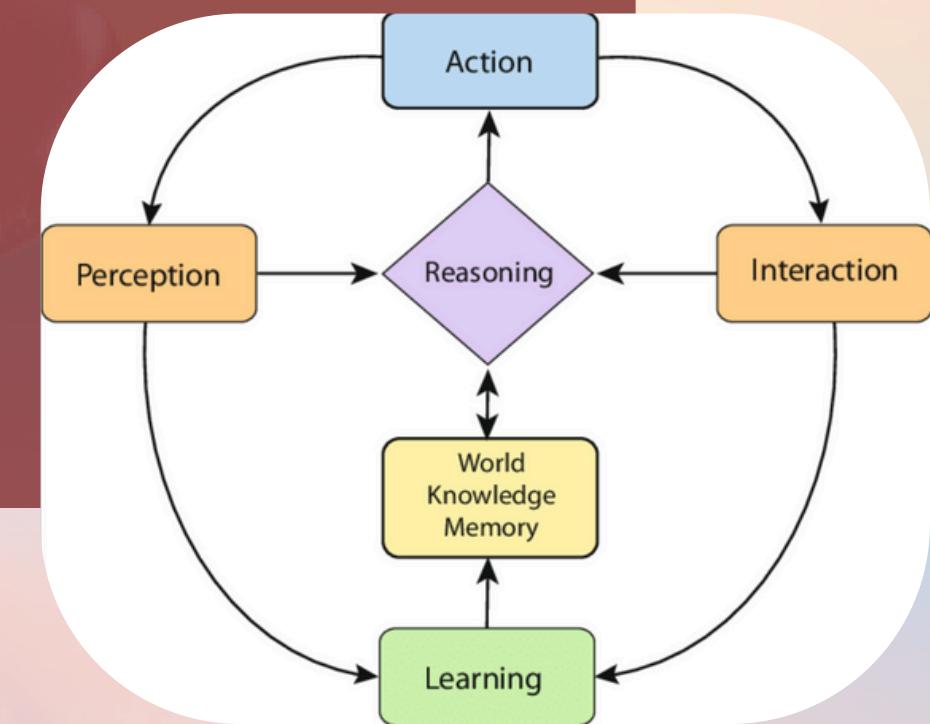


Ce projet applique directement les notions vues :

Agents intelligents: Selon la définition de Jennings et Wooldridge (1998), notre système est un agent situé dans un environnement (l'interface Tkinter), autonome (gère seul le flux) et flexible (adapte l'arbre en temps réel).

Capteurs : boutons Oui/Non pour percevoir les réponses (comme un agent logiciel avec clavier/écran).

Actionneurs : labels et images pour agir. Cela illustre le paradigme agent pour l'IA distribuée (IAD) et les SMA (un ensemble d'agents coopérants ; ici, interface et modèle coopèrent).



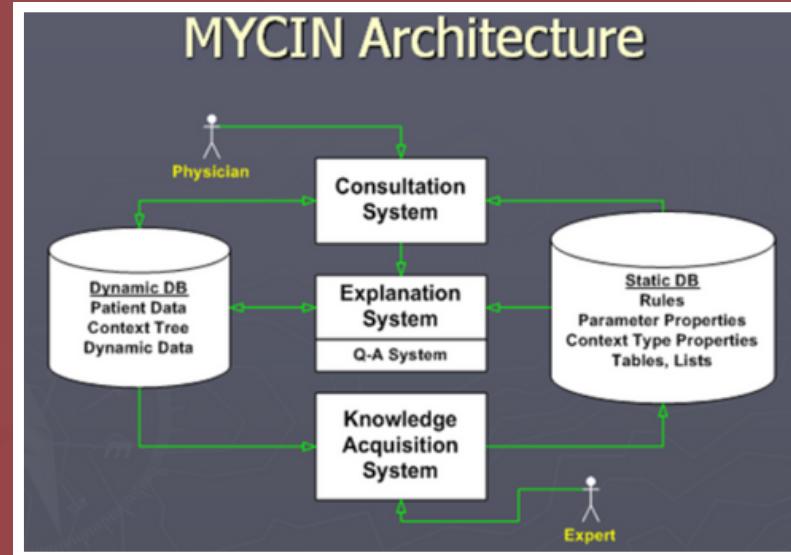


LIENS AVEC LES CONCEPTS DU COURS

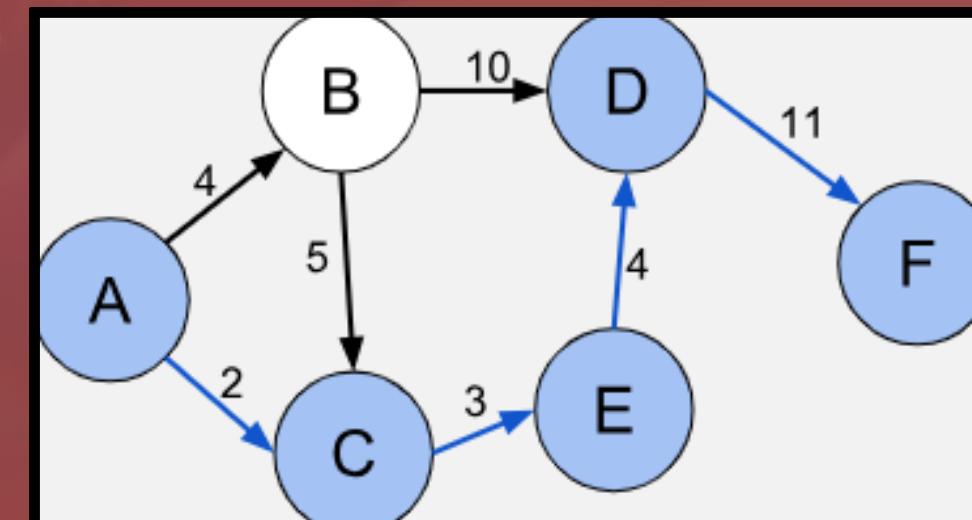


Ce projet applique directement les notions vues:

Systèmes experts : Comme dans MYCIN (programme de consultation, domaine anti-microbiens, 500 règles), notre système collecte des faits (questions comme symptômes) pour une conclusion (destination comme thérapie). Différence : inférence probabiliste vs règles IF-THEN pures.



Stratégies de recherche : Le questionnaire explore un espace d'états comme un graphe orienté (nœuds = états partiels, arcs = Oui/Non). État initial : vecteur vide ; successeur : ajout réponse ; solution optimale : max confiance (chemin minimal).



CHOIX DU THÈME



POURQUOI AVOIR CHOISI CE THÈME ?

PERTINENCE ACADEMIQUE

Ce thème permet d'explorer les piliers de l'IA : représentation des connaissances (dataset comme base de faits), inférence (modèle ML simulant un chaînage), et agents (système autonome et flexible). Il relie aux systèmes multi-agents (SMA) en cours, où des agents coopèrent ; ici, l'interface et le modèle "coopèrent" pour atteindre l'objectif de recommandation.

SIMPLICITÉ + RICHESSÉ

Le domaine des voyages est accessible, mais riche en IA : il implique une recherche dans un espace d'états (préférences → destinations), similaire aux graphes orientés vus en stratégies de recherche, où un chemin optimal mène à l'état but (destination idéale).

UTILE POUR UN RAPPORT OU SOUTENANCE

Le projet démontre une maîtrise des fondements : de la définition d'un problème (état initial : questions ; actions : réponses ; successeur : mise à jour du vecteur) à l'optimalité (confiance maximale via probabilités).

ARCHITECTURE GÉNÉRALE DU SYSTÈME

En IA, un système expert classique a une base de connaissances, un moteur d'inférence et une interface. Ici, la base est implicite dans le dataset, l'inférence via Random Forest (qui explore des arbres comme une recherche en graphe), et l'interface perçoit/agit comme un agent. La classe principale initialise ces composants, reliant à la définition d'un agent situé, autonome et flexible

```
class VoyageExpertSystem:  
    """Système expert pour recommander des destinations de voyage avec interface moderne."""  
  
    def __init__(self, root):  
        self.root = root  
        self.root.title("🌐 Système Expert de Voyage")  
        self.root.geometry("1200x750")  
        self.root.resizable(False, False)  
        self.root.configure(bg='black')  
  
        # Données  
        self.init_data()  
  
        # Variables d'état  
        self.reponses = []  
        self.index_question = 0  
        self.historique = []  
        self.images_cache = {}  
  
        # Entrainer le modèle  
        self.train_model()  
  
        # Charger les images  
        self.load_images()  
  
        # Construire l'interface  
        self.build_modern_ui()  
  
        # Démarrer  
        self.afficher_question()
```



ARCHITECTURE GÉNÉRALE DU SYSTÈME

Base de connaissances (dataset d'entraînement)

Les connaissances sont représentées comme un graphe implicite : critères booléens (faits) mènent à des destinations (conclusions). Cela s'inspire des règles de production en systèmes experts, mais encodées dans des données pour un apprentissage.

Exemple de données initiales :

```
def init_data(self):
    """Initialise les données du système expert."""
    self.X = [
        # [chaud, Nature, PasCher, HorsEurope, Culture, Fête, Plage, Montagne, Aventure, Luxe]
        [1, 1, 1, 1, 0, 0, 1, 0, 1, 0], # Bali
        [0, 0, 0, 1, 1, 0, 0, 0, 0], # Paris
        [0, 1, 0, 0, 0, 0, 1, 1, 0], # Islande
        [1, 0, 1, 1, 1, 0, 0, 0, 0], # Marrakech
        [1, 0, 0, 1, 1, 0, 0, 0, 1], # Tokyo
        [0, 1, 0, 0, 0, 0, 1, 1, 0], # Les Alpes
        [1, 1, 1, 0, 1, 1, 0, 0, 0], # Thaïlande
        [0, 0, 0, 1, 0, 0, 0, 0, 1], # Londres
        [1, 1, 0, 1, 1, 0, 0, 1, 0], # Pérou
        [1, 1, 1, 0, 0, 1, 0, 1, 0], # Maldives
        [0, 1, 1, 0, 1, 0, 0, 1, 1, 0], # Suisse
        [1, 0, 0, 1, 1, 1, 0, 0, 1, 1], # Dubai
    ]
    self.Y = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
```

Définition des règles

Def init_data(self):

C'est ici que se trouvent toutes les "règles" du système expert.

<code>self.X = [[1,1,0,1,0,0,1,0,1,0], # Bali</code>	# Bali  Tableau des règles
<code>[0,1,1,0,1,1,0,0,0,0], # Tokyo</code>	 → Matrice X

...

Ce sont les SI conditions
 Chaque ligne représente une destination.
 Chaque colonne une caractéristique.

<code>self.Y = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10,</code>	Conclusions → Labels Y
--	------------------------

C'est ici que tu associes chaque règle à une destination.

ARCHITECTURE GÉNÉRALE DU SYSTÈME

Interface utilisateur



Le Random Forest simule un moteur d'inférence en chaînage avant : part des faits (réponses) pour déduire la conclusion. En fondements d'IA, cela évoque une recherche dans un espace d'états, où chaque arbre explore des chemins optimaux.

```
def train_model(self):
    """Entraîne le modèle de machine learning."""
    self.clf = RandomForestClassifier(n_estimators=50, random_state=42, max_depth=5)
    self.clf.fit(self.X, self.Y)
```



ARCHITECTURE GÉNÉRALE DU SYSTÈME

Interface utilisateur

L'interface rend le système interactif, comme un agent percevant l'environnement utilisateur. Elle visualise l'arbre pour l'explicabilité, un aspect clé en IA symbolique.



The screenshot displays two windows side-by-side. On the left, the "Expert Voyage" application window shows a question: "Préfères-tu un climat chaud ?" with "OUI" and "NON" buttons. On the right, a "Decision Tree in Real Time" window visualizes the classification logic. The root node is "Culture <= 0.5". It branches into "True" (Mont <= 0.5) and "False" (Chaud <= 0.5). The "Mont" node leads to "Luxe <= 0.5" (value = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]), which further branches into "Luxe" (value = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]) and "Avent" (value = [0, 0, 0, 1, 0, 0, 0, 0, 0, 0]). The "Avent" node leads to "Nature <= 0.5" (value = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]), which leads to a final node "Pérou" (value = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]). Other nodes include "Islande" (value = [0, 0, 2, 1, 0, 0, 0, 2, 2, 1]), "Londres" (value = [0, 0, 0, 1, 0, 0, 2, 2, 0, 2, 1]), and "Pérou" (value = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]). The "Arbre de Décision en Temps Réel" window also includes text: "L'arbre se met à jour avec vos choix" and "Arbre complet - Commencez le questionnaire".



MODÈLE DE PRISE DE DÉCISION UTILISÉ

Représentation des connaissances

Les faits booléens (oui/non) et destinations forment une abstraction du problème, comme définir un état initial et des opérateurs en stratégies de recherche

Inférence avec Random Forest

L'entraînement et la prédiction appliquent une "recherche" probabiliste, optimale comme une solution minimisant le coût (ici, maximisant la confiance).

Code d'entraînement :



MODÈLE DE PRISE DE DÉCISION UTILISÉ



```
def train_model(self):  
    """Entraîne le modèle de machine learning."""  
    self.clf = RandomForestClassifier(n_estimators=50, random_state=42, max_depth=5)  
    self.clf.fit(self.X, self.Y)
```

Visualisation de l'arbre de décision

Visualise le raisonnement, reliant aux graphes en IA : noeuds = états, arcs = décisions.



DESCRIPTION DÉTAILLÉE DE CHAQUE FONCTION

init_data()

Initialise la base de faits, fondement de tout système expert.

train_model()

Entraîne l'inféreur, simulant l'acquisition de connaissances expertes.

load_images() et fonctions associées

Gère les visuels, améliorant l'interaction homme-machine en IA.

```
def add_coins_corners(img, radius=5):
    """Ajoute des coins arrondis à une image."""
    mask = Image.new('L', img.size, 0)
    draw = ImageDraw.Draw(mask)
    draw.rounded_rectangle([(0, 0), img.size], radius, fill=255)

    result = Image.new('RGBA', img.size)
    result.paste(img, (0, 0))
    result.putalpha(mask)
    return result
```

build_left_panel() et build_right_panel()

Construit l'interface, où le panneau droit visualise la "recherche" en temps réel.



DESCRIPTION DÉTAILLÉE DE CHAQUE FONCTION

afficher_question() et repondre(valeur)

Collecte les faits itérativement, comme une perception progressive en agent IA.

```
def repondre(self, valeur):
    """Enregistre la réponse et passe à la question suivante."""
    self.reponses.append(valeur)
    self.index_question += 1

    # Mettre à jour l'arbre en temps réel
    self.update_decision_tree()

    self.afficher_question()
```

faire_prediction()

Infère la conclusion avec probabilités, comme une résolution optimale

recommencer()

Réinitialise l'état, illustrant l'autonomie de l'agent.

afficher_historique() et effacer_historique()

Gère la mémoire, un aspect avancé en SMA.



DEMONSTARTION



 Système Expert de Voyage

Expert Voyage

Votre destination idéale en 10 questions

Question 1/10

Préfères-tu un climat chaud ?

OUI NON



 Arbre de Décision en Temps Réel

L'arbre se met à jour avec vos choix

Arbre complet - Commencez le questionnaire

```

graph TD
    Root["Climat: 0.5  
gpa: 0.847  
samples: 7  
value = [0, 0, 2, 1, 0, 0, 2, 2, 2, 1]  
class = Islande { }"] --> Mont["Mont: 0 = 0.5  
gpa: 0.5  
samples: 2  
value = [0, 0, 2, 0, 0, 0, 0]  
class = Islande { }"]
    Root --> Chaud["Chaud: 0 = 0.5  
gpa: 0.703  
samples: 5  
value = [0, 0, 0, 1, 0, 0, 0, 2, 0, 1]  
class = London { }"]
    Mont --> Ljung["Ljung: 0 = 0.5  
gpa: 0.625  
samples: 3  
value = [0, 0, 0, 0, 0, 0, 0, 0, 0]  
class = Ljung { }"]
    Mont --> Pern["Pern: 0 = 0.5  
gpa: 0.625  
samples: 3  
value = [0, 0, 0, 1, 0, 0, 0, 0, 0]  
class = Pern { }"]
    Chaud --> Nature["Nature: 0 = 0.5  
gpa: 0.664  
samples: 2  
value = [0, 0, 0, 0, 0, 0, 2, 0, 0, 1]  
class = Pern { }"]
    Nature --> Pern2["Pern: 0 = 0.5  
gpa: 0.664  
samples: 1  
value = [0, 0, 0, 0, 0, 0, 2, 0, 0, 0]  
class = Pern { }"]
  
```





DEMONSTARTION



Système Expert de Voyage

Expert Voyage

Votre destination idéale en 10 questions

✓ Terminé



Budget 1600€
Période Mai-Sept
Vol 14h
Temp 20°C

Historique Recommencer

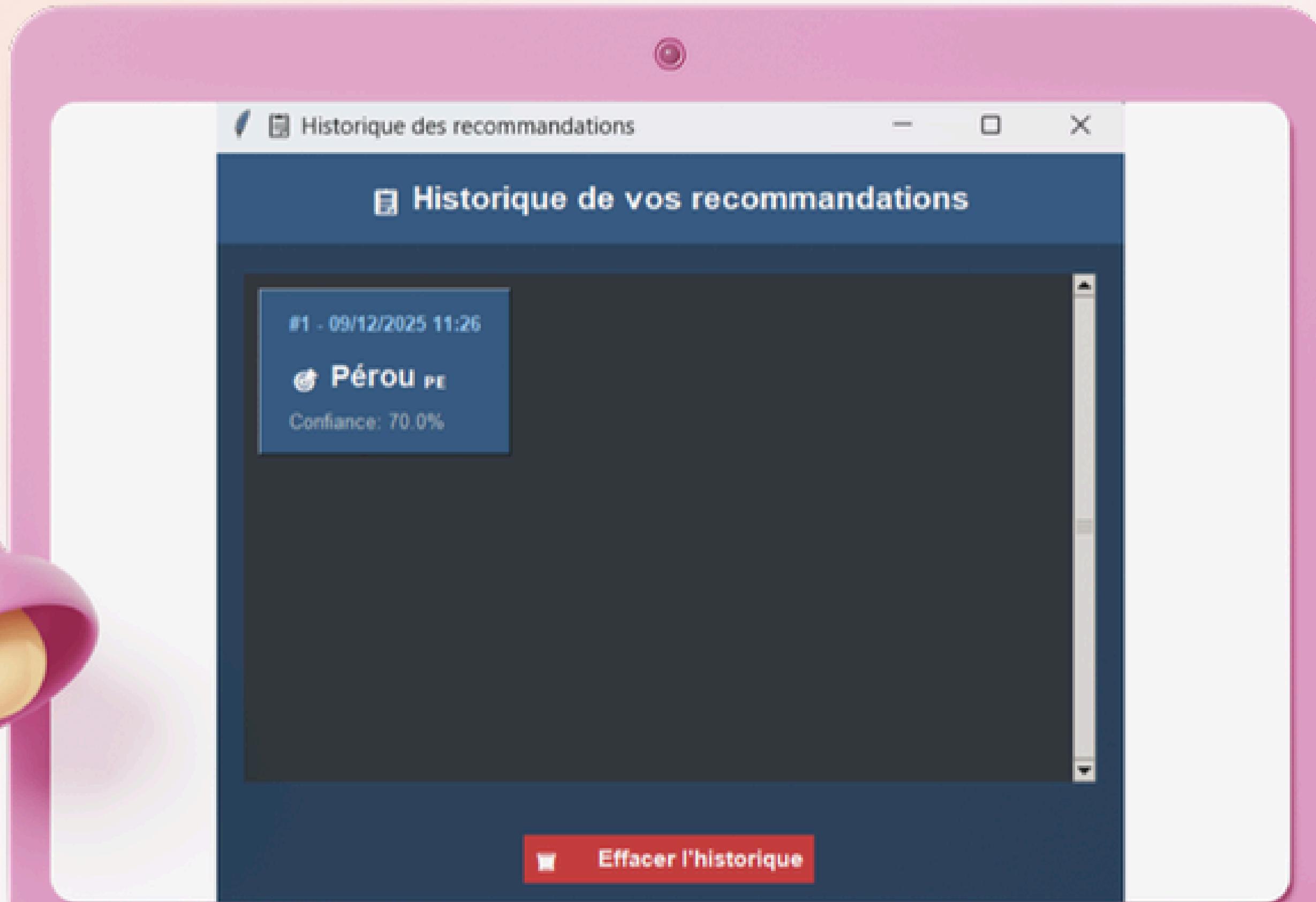
Arbre de Décision en Temps Réel

L'arbre se met à jour avec vos choix.

Après 10 question(s)
voya. ('Oui', 'Oui', 'Non', 'Oui', 'Oui', 'Non', 'Oui', 'Oui', 'Oui', 'Oui')

```
graph TD; Root["Culture: n=3  
gini=0.847  
samples=7  
value=[0, 0, 2, 1, 0, 0, 3, 2, 2, 2, 1]  
class=Inconnu [?];"]; Root -- Non --> Mont["Mont: n=3  
gini=0.785  
samples=5  
value=[0, 0, 2, 0, 0]  
class=Inconnu [?];"]; Root -- Oui --> Chaud["Chaud: n=3  
gini=0.791  
samples=5  
value=[0, 0, 1, 0, 0, 2, 2, 0, 2, 1]  
class=Londres [?];"]; Mont -- Non --> Soleil["Soleil: n=2  
gini=0.677  
samples=3  
value=[0, 0, 0, 0]  
class=Inconnu [?];"]; Mont -- Oui --> Arctique["Arctique: n=2  
gini=0.625  
samples=3  
value=[0, 0, 1, 0, 0, 2, 0, 1]  
class=Pompeii [?];"]; Chaud -- Non --> Soleil["Soleil: n=2  
gini=0.644  
samples=2  
value=[0, 0, 0, 0, 0, 0, 2, 0, 1]  
class=Pompeii [?];"]; Chaud -- Oui --> Nature["Nature: n=2  
gini=0.61  
samples=2  
value=[0, 0, 0, 0, 0, 0, 2, 0, 1]  
class=Pompeii [?];"]; Soleil -- Non --> Arctique["Arctique: n=2  
gini=0.5  
samples=1  
value=[0, 0, 0, 0, 0, 0, 2, 0, 1]  
class=Pompeii [?];"];
```

DEMONSTARTION



The screenshot shows a software interface with a dark blue header bar. The title bar reads "Historique des recommandations". The main title "Historique de vos recommandations" is centered above a list area. A single item in the list is highlighted with a blue background:
#1 - 09/12/2025 11:26
Pérou PE
Confiance: 70.0%

At the bottom right of the main window, there is a red button labeled "Effacer l'historique".



DEMONSTARITION



 Système Expert de Voyage

Expert Voyage

Votre destination idéale en 10 questions

Question 1/10

 Préfères-tu un climat chaud ?

OUI NON



 Arbre de Décision en Temps Réel

L'arbre se met à jour avec vos choix

Arbre complet - Commencez le questionnaire

```

graph TD
    Root["Culture = 0.5  
geno = 0.047  
tempres = 2  
values = [0, 0, 2, 1, 0, 0, 0, 2, 2, 2, 1]  
class = Islande (?)"] --> Mont["Mont = 0.5  
geno = 0.5  
tempres = 2  
values = [0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 1]  
class = Islande (?)"]
    Root --> Chaud["Chaud = 0.5  
geno = 0.193  
tempres = 1  
values = [0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1]  
class = Islande (?)"]
    Mont --> Laponie["Laponie  
geno = 0.005  
tempres = 1  
values = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1]  
class = Laponie (?)"]
    Mont --> Perou["Perou  
geno = 0.425  
tempres = 1  
values = [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1]  
class = Perou (?)"]
    Chaud --> Laponie["Laponie  
geno = 0.005  
tempres = 1  
values = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1]  
class = Laponie (?)"]
    Chaud --> Perou["Perou  
geno = 0.444  
tempres = 2  
values = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1]  
class = Perou (?)"]
    Laponie --> France["France  
geno = 0.0  
tempres = 1  
values = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1]  
class = France (?)"]
    Perou --> France["France  
geno = 0.0  
tempres = 1  
values = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1]  
class = France (?)"]
  
```

CONCLUSION

Ce projet démontre une maîtrise complète du code développé tout en illustrant concrètement les fondements d'IA vus en cours : agents intelligents (perception/action autonome), stratégies de recherche (espace d'états et graphes implicites via arbres), et systèmes experts (collecte de faits et inférence). L'approche hybride (ML + visualisation) rend le système flexible et explicable partiellement, tout en restant modulaire et élégant.

