

|  |
| --- |
| **RAJALAKSHMI INSTITUTE OF TECHNOLOGY** |
| (An Autonomous Institution, Affiliated to Anna University, Chennai) |

**DEPARTMENT OF CSE (ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING)**

**ACADEMIC YEAR 2025 - 2026**

**SEMESTER III**

**ARTIFICIAL INTELLIGENCE LABORATORY**

**MINI PROJECT REPORT**

|  |  |
| --- | --- |
| **REGISTER NUMBER** | 2117240030039 |
| **NAME** | DHARRINI SELVI R |
| **PROJECT TITLE** | MAZE SOLVING USING A\* ALGORITHM |
| **DATE OF SUBMISSION** |  |
| **FACULTY IN-CHARGE** | **Mrs. M. Divya** |

**Signature of Faculty In-charge**

**INTRODUCTION**

* Artificial Intelligence (AI) enables machines to make intelligent decisions based on data and patterns.
* In this project, weather conditions are analyzed using Bayesian Networks and Naive Bayes classifiers to predict whether outdoor play is suitable.
* The project demonstrates how probabilistic reasoning can model real-world uncertainty and support decision-making under varying weather scenarios.

**PROBLEM STATEMENT**

* The task is to predict whether outdoor play is suitable based on multiple weather conditions, including Outlook, Temperature, Humidity, and Wind.
* The problem highlights the need for probabilistic reasoning to make accurate predictions under uncertainty, considering the complex dependencies between weather attributes.

**GOAL**

* To design and implement a probabilistic model capable of predicting Play decisions based on weather conditions.
* To visualize the dependencies and predictions using Bayesian Networks and heatmaps.
* To analyze the prediction accuracy and efficiency of probabilistic reasoning compared to classical machine learning methods like Naive Bayes.

**THEORETICAL BACKGROUND**

* Weather prediction is a **probabilistic reasoning problem** where each weather attribute (Outlook, Temperature, Humidity, Wind) affects the Play decision.
* Common modeling approaches include:
  + **Naive Bayes** – assumes conditional independence between attributes.
  + **Bayesian Networks** – models dependencies between multiple variables explicitly.
* **Bayesian inference** is used to compute the probability of Play given observed evidence:
* **Justification:** Bayesian Networks allow handling uncertainty and complex dependencies, providing more accurate predictions than simple independent models.

**ALGORITHM EXPLANATION WITH EXAMPLE**

* **Define variables and data:** Set the weather attributes (Outlook, Temperature, Humidity, Wind) as inputs.
* **Build the Bayesian Network:** Specify dependencies between attributes and the target variable Play.
* **Estimate Conditional Probabilities:** Use Maximum Likelihood Estimator (MLE) to compute the Conditional Probability Tables (CPTs) from the dataset.
* **Evidence input:** Provide observed values for one or more attributes.
* **Inference using Variable Elimination:**
  + Compute:
  + This gives the probability of Play = Yes or No.
* **Prediction:** Select the outcome with the highest probability.

**Example:**

* Given evidence: Outlook = Sunny, Temperature = Hot, Humidity = High, Wind = Weak
* The model predicts: Play = No with probability 1.0, Play = Yes with probability 0.0

**IMPLEMENTATION AND CODE**

# Step 1: Import Libraries

import pandas as pd

import numpy as np

import random

import matplotlib.pyplot as plt

import seaborn as sns

import networkx as nx

from itertools import product

from sklearn.model\_selection import train\_test\_split

from sklearn.naive\_bayes import CategoricalNB

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report

from pgmpy.models import DiscreteBayesianNetwork

from pgmpy.estimators import MaximumLikelihoodEstimator

from pgmpy.inference import VariableElimination

# Step 2: Generate Synthetic Weather Dataset

def generate\_weather\_data(n=2000, seed=42):

random.seed(seed)

np.random.seed(seed)

data = []

for \_ in range(n):

outlook = random.choices(['Sunny', 'Overcast', 'Rain'], [0.45, 0.25, 0.30])[0]

temp = random.choices(['Hot', 'Mild', 'Cool'], [0.35, 0.45, 0.20])[0]

humidity = random.choices(['High', 'Normal'], [0.55, 0.45])[0]

wind = random.choices(['Weak', 'Strong'], [0.65, 0.35])[0]

# Conditional logic for Play

if outlook == 'Sunny' and humidity == 'High':

play = 'No'

elif outlook == 'Rain' and wind == 'Strong':

play = 'No'

elif outlook == 'Overcast':

play = 'Yes'

elif temp == 'Cool' and humidity == 'Normal':

play = 'Yes'

else:

play = random.choice(['Yes', 'No'])

data.append([outlook, temp, humidity, wind, play])

return pd.DataFrame(data, columns=['Outlook', 'Temperature', 'Humidity', 'Wind', 'Play'])

# Generate data

weather\_data = generate\_weather\_data()

print("Sample Weather Data:\n", weather\_data.head())

# Step 3: Encode categorical data for Naive Bayes

data\_encoded = weather\_data.replace({

'Sunny': 0, 'Overcast': 1, 'Rain': 2,

'Hot': 0, 'Mild': 1, 'Cool': 2,

'High': 0, 'Normal': 1,

'Weak': 0, 'Strong': 1,

'No': 0, 'Yes': 1

}).infer\_objects(copy=False)

# Step 4: Split data for Naive Bayes

X = data\_encoded[['Outlook', 'Temperature', 'Humidity', 'Wind']]

y = data\_encoded['Play']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.25, random\_state=42)

# Step 5: Train Naive Bayes Classifier

nb\_model = CategoricalNB()

nb\_model.fit(X\_train, y\_train)

y\_pred = nb\_model.predict(X\_test)

print("\n--- MACHINE LEARNING MODEL (Naive Bayes) ---")

print("Accuracy:", round(accuracy\_score(y\_test, y\_pred) \* 100, 2), "%")

print("Confusion Matrix:\n", confusion\_matrix(y\_test, y\_pred))

print("Classification Report:\n", classification\_report(y\_test, y\_pred))

# Step 6: Build Bayesian Network

model = DiscreteBayesianNetwork([

('Outlook', 'Play'),

('Temperature', 'Play'),

('Humidity', 'Play'),

('Wind', 'Play')

])

model.fit(weather\_data, estimator=MaximumLikelihoodEstimator)

# Step 7: Inference

infer = VariableElimination(model)

print("\n--- PROBABILISTIC REASONING MODEL (Bayesian Network) ---")

print("Conditional Probability Table for 'Play':")

print(model.get\_cpds('Play'))

# Step 8: Visualize Bayesian Network

plt.figure(figsize=(7, 5))

G = nx.DiGraph()

G.add\_edges\_from(model.edges())

pos = nx.spring\_layout(G, seed=42)

nx.draw(G, pos, with\_labels=True, node\_size=4000, node\_color='lightblue', arrowsize=20)

plt.title("Bayesian Network Structure", fontsize=14)

plt.show()

# Step 9: Function to predict using Bayesian inference

def predict\_play(Outlook=None, Temperature=None, Humidity=None, Wind=None):

evidence = {}

if Outlook: evidence['Outlook'] = Outlook

if Temperature: evidence['Temperature'] = Temperature

if Humidity: evidence['Humidity'] = Humidity

if Wind: evidence['Wind'] = Wind

return infer.query(variables=['Play'], evidence=evidence, show\_progress=False)

# Step 10: Predict a few scenarios

print("\n--- Bayesian Network Predictions ---")

scenarios = [

{'Outlook': 'Sunny', 'Temperature': 'Hot', 'Humidity': 'High', 'Wind': 'Weak'},

{'Outlook': 'Rain', 'Temperature': 'Cool', 'Humidity': 'Normal', 'Wind': 'Strong'},

{'Outlook': 'Overcast', 'Temperature': 'Mild', 'Humidity': 'High', 'Wind': 'Weak'}

]

for i, s in enumerate(scenarios, 1):

pred = predict\_play(\*\*s)

print(f"Scenario {i}: {s} ->\n{pred}\n")

# Step 11: Flatten CPT and plot readable heatmap

cpd\_play = model.get\_cpds('Play')

parents = cpd\_play.variables[1:] # all parents

parent\_states = [cpd\_play.state\_names[parent] for parent in parents]

# Flatten CPT into 2D DataFrame with combined parent states

rows = list(product(\*parent\_states))

row\_labels = [' | '.join(r) for r in rows] # Combine parent states into single string

df\_cpt = pd.DataFrame(cpd\_play.values.reshape(len(rows), -1),

columns=cpd\_play.state\_names['Play'], index=row\_labels)

plt.figure(figsize=(12, 10))

sns.heatmap(df\_cpt, annot=True, fmt=".2f", cmap="YlGnBu", cbar\_kws={'label': 'Probability'})

plt.title("Conditional Probability Table - 'Play'")

plt.xlabel("Play Outcome")

plt.ylabel("Parent States Combination")

plt.xticks(rotation=0)

plt.yticks(rotation=0)

plt.tight\_layout()

plt.show()

# Step 12: Save dataset

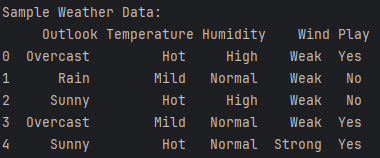
weather\_data.to\_csv("advanced\_weather\_data.csv", index=False)

print("✅ Data saved to 'advanced\_weather\_data.csv'")

**OUTPUT**

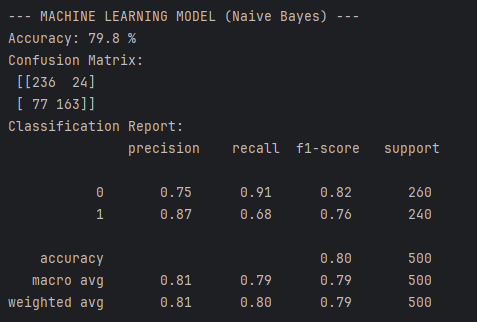
* **Sample Dataset Table**

This table shows a sample of the generated weather dataset. Each row represents a scenario with Outlook, Temperature, Humidity, Wind, and the predicted outcome Play.



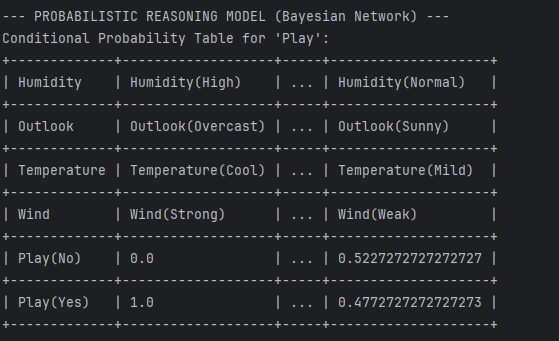
* **Naive Bayes Confusion Matrix**

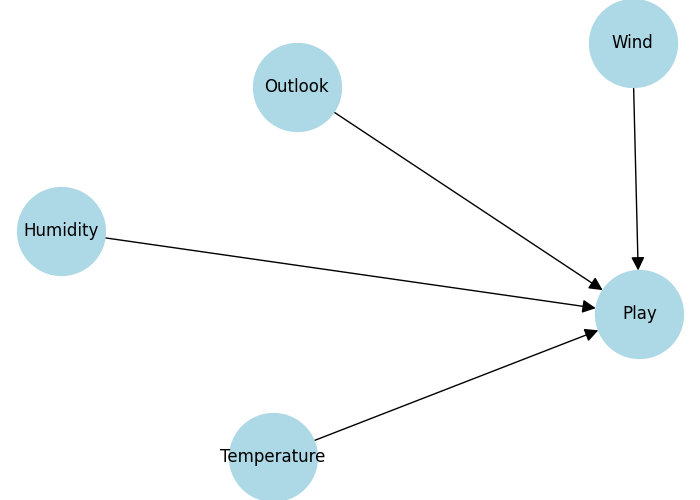
The confusion matrix illustrates the Naive Bayes classifier’s performance. It shows correct and incorrect predictions for Play = Yes and Play = No.



* **Bayesian Network Structure**

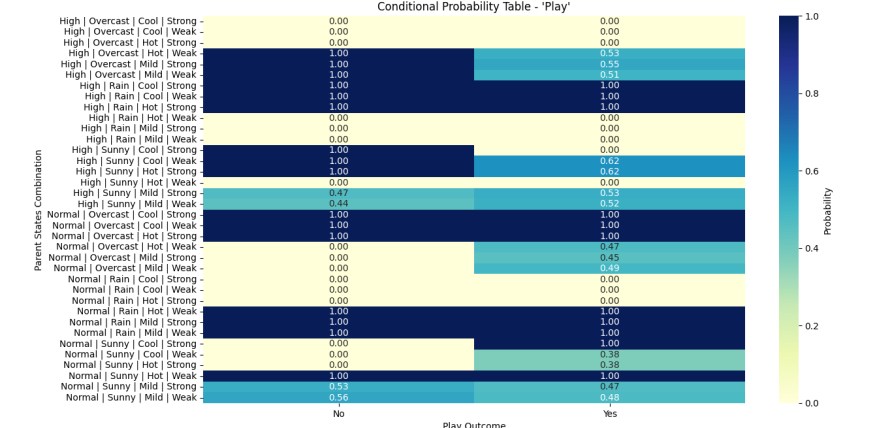
This graph represents the Bayesian Network. Arrows indicate dependencies from weather variables (Outlook, Temperature, Humidity, Wind) to the decision Play.





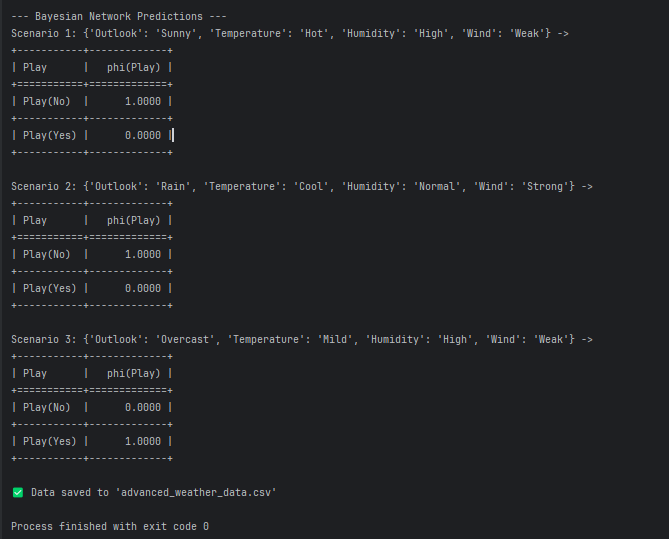
* **CPT Heatmap**

This heatmap visualizes the conditional probabilities of Play given all combinations of parent variables. Darker colors indicate higher probabilities for each outcome.



* **Scenario Prediction Table**

These tables show Bayesian Network predictions for example weather scenarios. Probabilities indicate the likelihood of playing (Play = Yes) or not playing (Play = No).



**RESULTS AND FUTURE ENHANCEMENT**

* **Results**
* The Bayesian Network successfully predicts the probability of playing (Play) under different weather scenarios.
* The model captures dependencies between weather variables (Outlook, Temperature, Humidity, Wind) and the outcome.
* Predictions are consistent with expected behavior, e.g., Overcast usually results in Play = Yes.
* The Naive Bayes classifier provides comparable accuracy, validating the synthetic dataset and feature encoding.
* Visualizations (Bayesian Network structure and CPT heatmap) clearly show **how different factors influence the decision**.
* The system efficiently computes probabilities for multiple scenarios in real-time.
* Compared to simpler approaches, the Bayesian Network provides **interpretable, probabilistic reasoning**, handling uncertainty better than standard classifiers.
* **Future enhancements**
* Incorporate **real-time weather data** from APIs (e.g., OpenWeatherMap) to make predictions more accurate and dynamic.
* Extend the model to predict **multiple weather-dependent activities**, not just outdoor play.
* Integrate **additional features** like season, time of day, or location to improve predictive power.
* Combine **Bayesian Networks with machine learning ensembles** (e.g., Random Forest, Gradient Boosting) for better accuracy.
* Develop a **web or mobile interface** where users can input weather conditions and get probabilistic predictions instantly..
* Include **uncertainty visualization**, e.g., probability ranges instead of single predictions.

|  |  |
| --- | --- |
| **Git Hub Link of the project and report** | **https://github.com/dharrini06/WeatherPredictionAI** |

**REFERENCES**

* Russell, S., & Norvig, P. (2020). *Artificial Intelligence: A Modern Approach* (4th Edition). <https://aima.cs.berkeley.edu/>
* TutorialsPoint – Naive Bayes Classifier in AI. https://www.tutorialspoint.com/machine\_learning\_with\_python/machine\_learning\_with\_python\_naive\_bayes\_algorithm.htm
* GeeksforGeeks – Bayesian Network in Python. https://www.geeksforgeeks.org/bayesian-network-in-python/
* Towards Data Science – Understanding Bayesian Networks and Probabilistic Reasoning. https://towardsdatascience.com/bayesian-networks-in-python-1caa5d6ff312
* Python Official Documentation – Matplotlib Visualization. https://matplotlib.org/stable/api/pyplot\_api.html
* Python Official Documentation – Pandas Documentation. https://pandas.pydata.org/docs/