MACHINE LEARNING CLASSIFICATION

Naive Bayes
Algorithm

Predicting Tennis Play Based on Weather Conditions

Presented By: Partha Sarathi R | Ayyapparaja VJ | Thirupathi Yaswanth

Probability
Foundation of prediction

Bayes Theorem
Conditional probability

Classification
Yes or No prediction

DATASET	FEATURES	ALGORITHM
1,202 Records	4 Weather Attributes	CategoricalNB

Understanding **Probability**

Independent Events

Events that don't affect each other's probability

Rolling a Dice {1, 2, 3, 4, 5, 6}

$P(1) = {}^1/_6$ $P(2) = {}^1/_6$ $P(3) = {}^1/_6$

Each roll is independent - previous results don't affect future rolls!

Dependent Events

Events where one affects the probability of another

Marble Bag Problem

What is the probability of removing a **white marble** and then a **yellow marble**?

First Draw

○ ○ ○ ○ ○

$P(W) = {}^3/_5 \rightarrow$ 1st Event

↓

Second Draw (after white removed)

○ ○ ○ ○

$P(Y|W) = {}^2/_4 = {}^1/_2 \rightarrow$ 2nd Event

↳ Conditional Probability

$P(W \text{ and } Y) = P(W) \times P(Y|W) \Rightarrow$ Independent Event

↳ Conditional Probability

CORE FORMULA

Bayes' Theorem

1. $P(A \text{ and } B) = P(B \text{ and } A)$
2. $P(A) \times P(B|A) = P(B) \times P(A|B)$

★

$$P(B|A) = \frac{P(B) \times P(A|B)}{P(A)}$$

⇒ Bayes' Theorem

Dataset Structure

INDEPENDENT FEATURES

X_1 | X_2 | X_3 | X_4 | ... | X_n → Y

Outlook | Temp | Humidity | Wind | Play?

Extended Naive Bayes Formula

General Form: $P(Y|X_1,X_2,X_3...X_n) = P(Y) \times P(X_1,X_2,X_3,X_4...X_n|Y) / P(X_1,X_2,X_3...X_n)$

Naive Assumption: $= P(Y) \times P(X_1|Y) \times P(X_2|Y) \times P(X_3|Y) ... \times P(X_n|Y) / P(X_1,X_2,X_3...X_n)$

*"Naive" assumes all features are **independent** of each other!*

TRAINING DATA

Play Tennis Dataset

Let's Solve This Problem

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes

P(Yes) $^{9}/_{14}$

P(No) $^{5}/_{14}$

Outlook

	Yes	No	P(E|Yes)	P(E|No)
Sunny	2	3	$^{2}/_{9}$	$^{3}/_{5}$
Overcast	4	0	$^{4}/_{9}$	$^{0}/_{5}$
Rain	3	2	$^{3}/_{9}$	$^{2}/_{5}$

Temperature

	Yes	No	P(E|Yes)	P(E|No)
Hot	2	2	$^{2}/_{9}$	$^{2}/_{5}$
Mild	4	2	$^{4}/_{9}$	$^{2}/_{5}$
Cool	3	1	$^{3}/_{9}$	$^{1}/_{5}$

STEP-BY-STEP

Test: (Sunny, Hot) → Play?

Calculating P(Yes | Sunny, Hot)

$$P(Yes|(Sunny,Hot)) = \frac{P(Yes) \times P(Sunny|Yes) \times P(Hot|Yes)}{\cancel{P(Sunny) \times P(Hot)} \leftarrow \text{Constant}}$$

$$= \frac{9}{14} \times \frac{2}{9} \times \frac{2}{9}$$

$= \frac{2}{63} =$ **0.031**

Calculating P(No | Sunny, Hot)

$$P(No|(Sunny,Hot)) = \frac{P(No) \times P(Sunny|No) \times P(Hot|No)}{\cancel{P(Sunny) \times P(Hot)} \leftarrow \text{Constant}}$$

$$= \frac{5}{14} \times \frac{3}{5} \times \frac{2}{5}$$

$= \frac{3}{35} =$ **0.085**

Final Prediction

Normalizing Probabilities

$$P(Yes|(Sunny,Hot)) = \frac{0.031}{0.031 + 0.085} = \mathbf{0.27} = 27\%$$

$$P(No|(Sunny,Hot)) = \frac{0.085}{0.031 + 0.085} = \mathbf{0.73} = 73\%$$

Probability Comparison

Play Tennis (Yes) **27%**

Don't Play (No) **73%**

Test Data: **(Sunny, Hot) = 73%**

↓

🎾❌

They will NOT play Tennis

⇒ Person is n ... play Tennis

🐍 Training Code (train_model.py)

```
import pandas as pd
from sklearn.naive_bayes import CategoricalNB
from sklearn.preprocessing import LabelEncoder

# Load dataset (1202 records)
df = pd.read_csv('play_tennis_1202.csv')

# Encode categorical features
encoders = {}
for col in df.columns:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
    encoders[col] = le

# Train Naive Bayes model
model = CategoricalNB()
model.fit(X_train, y_train)

# Model accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy * 100:.2f}%")
```

🎮 Try Live Prediction

Outlook: ☀️ Sunny

Temperature: 🔥 Hot

Humidity: 💧 High

Wind: 🍃 Weak

🎾 **Predict**

Select weather conditions and click Predict!

📊 **1,202** Training Records

🎯 **96.25%** Model Accuracy

⚡ **4** Weather Features

🧠 **NB** Algorithm