AI LAB WEEK 8

Dr. Mubashir Ahmad

NAÏVE BAYES' BASE THEOREM

- Bayes theorem is the cornerstone of Naïve Bayes Classifier because it provides a way to calculate the posterior probability **P(h|D), from the prior** probability **P(h), together with P(D) and P(D(h).**

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)} \longrightarrow \text{Posterior probability}$$

$(P(D) = P(y) + P(N)$

```
D is the datset and h is the hypothesis, for example yes/no
The hypothesis which give the maximum value, you consider it
as classification results.
```

NAÏVE BAYES' BASE THEOREM

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

Posterior probability

maximum Hypothesis

$$\begin{aligned} h_{MAP} &\equiv \underset{h \in H}{\operatorname{argmax}}\, P(h|D) \\ &= \underset{h \in H}{\operatorname{argmax}}\, \frac{P(D|h)P(h)}{\cancel{P(D)}} \\ &= \underset{h \in H}{\operatorname{argmax}}\, P(D|h)P(h) \end{aligned}$$

P(D) is divided with every hypothesis

NAÏVE BAYES' BASE THEOREM

- The Bayesian approach to classifying the new instance is to assign the most probable target value, $\mathbf{v_{MAP}}$ given the attribute values **<a1, a2 . . .an> that describe** the instance.

$$v_{MAP} = \underset{v_j \in V}{\operatorname{argmax}} P(v_j | a_1, a_2 \ldots a_n)$$

- We can use Bayes theorem to rewrite this expression as

$$v_{MAP} = \underset{v_j \in V}{\operatorname{argmax}} \frac{P(a_1, a_2 \ldots a_n | v_j) P(v_j)}{\cancel{P(a_1, a_2 \ldots a_n)}}$$

$$= \underset{v_j \in V}{\operatorname{argmax}} P(a_1, a_2 \ldots a_n | v_j) P(v_j)$$

- Naïve Bayes Classifier: $v_{NB} = \underset{v_j \epsilon V}{\operatorname{argmax}}\, p(v_j) \prod_i p(a_i | v_j)$

$$\boxed{\begin{aligned} h_{MAP} &\equiv \underset{h \in H}{\operatorname{argmax}} P(h|D) \\ &= \underset{h \in H}{\operatorname{argmax}} \frac{P(D|h)P(h)}{P(D)} \\ &= \underset{h \in H}{\operatorname{argmax}} P(D|h)P(h) \end{aligned}}$$

hypothesis

NAÏVE BAYESIAN CLASSIFIER

```
import pandas as pd
from collections import defaultdict
data = {
'Day': ['D1', 'D2', 'D3', 'D4', 'D5', 'D6', 'D7', 'D8', 'D9', 'D10', 'D11', 'D12', 'D13', 'D14'],
'Outlook': ['Sunny', 'Sunny', 'Overcast', 'Rain', 'Rain', 'Rain', 'Overcast', 'Sunny', 'Sunny', 'Rain', 'Sunny', 'Overcast',
'Overcast', 'Rain'],
 'Temp.': ['Hot', 'Hot', 'Hot', 'Mild', 'Cool', 'Cool', 'Cool', 'Mild', 'Cool', 'Mild', 'Mild', 'Mild', 'Hot', 'Mild'],
 'Humidity': ['High', 'High', 'High', 'High', 'Normal', 'Normal', 'Normal', 'High', 'Normal', 'Normal', 'Normal', 'High',
'Normal', 'High'],
'Wind': ['Weak', 'Strong', 'Weak', 'Weak', 'Weak', 'Strong', 'Strong', 'Weak', 'Weak', 'Weak', 'Strong', 'Strong', 'Weak',
'Strong'],
'Decision': ['No', 'No', 'Yes', 'Yes', 'Yes', 'No', 'Yes', 'No', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'No']
}

df = pd.DataFrame(data)
```

NAÏVE BAYESIAN CLASSIFIER

```
# Function to calculate probabilities
def calculate_probabilities(df):
    # Calculate prior probabilities
    total_instances = len(df)
    decision_counts = df['Decision'].value_counts()
    prior_prob_yes = decision_counts['Yes'] / total_instances
    prior_prob_no = decision_counts['No'] / total_instances
```

NAÏVE BAYESIAN CLASSIFIER

```
# Calculate conditional probabilities
    conditional_probs = defaultdict(dict)
    for column in df.columns[1:-1]:
        # Exclude 'Day' and 'Decision' columns
        for value in df[column].unique():
            for decision in df['Decision'].unique():
                count = len(df[(df[column] == value) & (df['Decision'] == decision)])
                total_decision_count = decision_counts[decision]
                conditional_probs[column][(value, decision)] = count / total_decision_count
    return prior_prob_yes, prior_prob_no, conditional_probs
```

NAÏVE BAYESIAN CLASSIFIER

```
# Function to predict decision for new instance
def predict_decision(instance, prior_prob_yes, prior_prob_no, conditional_probs):
    prob_yes = prior_prob_yes
    prob_no = prior_prob_no
    for attr, value in instance.items():
        if attr != 'Decision':
            prob_yes *= conditional_probs[attr].get((value, 'Yes'), 0)
            prob_no *= conditional_probs[attr].get((value, 'No'), 0)
    return 'Yes' if prob_yes > prob_no else 'No'
```

NAÏVE BAYESIAN CLASSIFIER

```
# Calculate probabilities
prior_prob_yes, prior_prob_no, conditional_probs = calculate_probabilities(df)

# Test with a new instance
new_instance = {'Outlook': 'Sunny', 'Temp.': 'Cool', 'Humidity': 'high', 'Wind': 'Strong'}
predicted_decision = predict_decision(new_instance, prior_prob_yes, prior_prob_no,
conditional_probs)
print("Predicted decision for the new instance:", predicted_decision)
```

TASKS

#1: Implement the given dataset in list of list.

#2: Function to calculate probabilities (Prior Prob)

#3: Calculate conditional probabilities. (Posterior Prob)

#4: Function to predict decision for new instance

#5: Test with a new instance