Shweta_Shinde_Math_Assignment_4

September 28, 2024

0.1 Shweta Ajay Shinde 017548687

MSDA, SJSU , Data 220- Math Method for DA

[10]:
```
    #Bayes Theorem
    # Returns:
    #P(D')= (1 - P(D))
    #P(T):- P (T/D) * P(D) + P(T | D') * P(D')

    #Formula P(D/T)=  P(T/D) * P(D)
                    #  ------------
                    #      P(T)
```

Scenario: Disease Screening

A particular disease is more common in older people. Let's consider two age groups: under 50 and over 50. The prevalence of the disease is 2% in the under 50 age group and 8% in the over 50 age group. A screening test has different accuracy levels for these age groups: 95% accuracy for under 50s and 90% for over 50s. However, the test has a 5% false positive rate in both groups.

The task is to calculate the probability of having the disease given a positive test result in each age group. Make a python code (Submit your .ipynb file) for this task.

1) Make and implement this function (2 pts):

2) Complete the above task by setting variables and this function for both groups (1 pts).

3) Obtain the probability for both groups (2 pts).

[11]:
```
import numpy as np

# Define the prior, accuracy, false_positive_rate
def bayesian_update_disease(prior, accuracy, false_positive_rate):
    # - prior: P(D): The probability of having the disease
    # - accuracy:P(T | D): The probability of a positive test result given that␣
 ↪the person has the disease
    # - false_positive_rate: P(T | D`): The probability of a positive test␣
 ↪result given that the person does not have the disease.

    #True Positives (TP): accuracyP(T/D) * prior P(D)
    TP = accuracy * prior
    # False Positives (FP): false_positive_rate  * (1 - prior)
    #                        P(T | D') *  P(D')
    FP = false_positive_rate * (1 - prior)

    # Total probability of a positive test
    total_positive = TP + FP

    # Probability of having the disease given a positive test result (Bayes'␣
  ↪theorem)
    probability = TP / total_positive

    return probability

# test has a 5% false positive rate
false_positive_rate = 0.05

# Variables for under 50
prior_under_50 = 0.02
accuracy_under_50 = 0.95

# Calculate probability of positive test for under 50
under = bayesian_update_disease(prior_under_50, accuracy_under_50,␣
  ↪false_positive_rate)


# Variables for over 50
prior_over_50 = 0.08
accuracy_over_50 = 0.90

# Calculate probability of positive test for over 50
over = bayesian_update_disease(prior_over_50, accuracy_over_50,␣
  ↪false_positive_rate)

# Display the result
print(f"The probability of having the disease given a positive test result for␣
  ↪under 50 age: {under:.2f}")

print(f"The probability of having the disease given a positive test result for␣
  ↪over 50 age: {over:.2f}")
```

```
The probability of having the disease given a positive test result for under 50
age: 0.28
The probability of having the disease given a positive test result for over 50
age: 0.61
```

```
[ ]:
```

Suppose a weather forecast model predicts rain with a 70% accuracy rate. However, it also incorrectly predicts rain 30% of the time when it's not going to rain. Let's say the actual chance of rain on any given day in a particular region is 20%.

We want to calculate the probability of it actually raining given that the forecast predicts rain.

Make a python code (Submit your .ipynb file) for this task.

1) Make and implement this function (2 pts):
2) Complete the above task by setting variables and calling this function (1 pts).
3) Obtain the probability of it actually raining given the forecast predicts rain (2 pts).

[12]:

```python
def bayesian_update_weather(prior, livelihood, false_positive_rate):

    # True Positives (TP): livelihood * prior
    TP = livelihood * prior

    # False Positives (FP): false_positive_rate * (1 - prior)
        #                      P(T | R') *  P(R')
    FP = false_positive_rate * (1 - prior)

    # Total probability of a positive forecast (predicting rain)
    total_positive = TP + FP

    # Probability of rain given a positive forecast (Bayes' theorem)
    probability = TP / total_positive

    return probability

# Variables for the weather forecast
prior_rain = 0.20  # Actual chance of rain P(R)
livelihood = 0.70  # livelihood of the forecast p(T/R)
false_positive_rate = 0.30  # Rate of incorrectly predicting rain P(T/R')

# Calculate the probability of it actually raining given that the forecast predicts rain
probability_of_rain = bayesian_update_weather(prior_rain, livelihood, false_positive_rate)

# Display the result
print(f"The probability of it actually raining given that the forecast predicts rain: {probability_of_rain:.2f}")
```

```
The probability of it actually raining given that the forecast predicts rain:
0.37
```

[ ]:

```python

```