Arhaan_220962050_lab7

September 6, 2024

Q1

[1]:
```
import pandas as pd
from collections import defaultdict
import numpy as np
from collections import defaultdict
from typing import List, Dict
```

(a)

[2]:
```
P_hosteler = 0.60
P_day_scholar = 0.40
P_A_given_hosteler = 0.30
P_A_given_day_scholar = 0.20
P_A = (P_A_given_hosteler * P_hosteler) + (P_A_given_day_scholar *␣
  ↪P_day_scholar)
P_hosteler_given_A = (P_A_given_hosteler * P_hosteler) / P_A
```

[3]:
```
P_hosteler_given_A
```

[3]:
```
0.6923076923076923
```

(b)

[4]:
```
P_disease = 0.01
P_no_disease = 0.99
P_positive_given_disease = 0.99
P_positive_given_no_disease = 0.02
P_positive_test = (P_positive_given_disease * P_disease) +␣
  ↪(P_positive_given_no_disease * P_no_disease)
P_disease_given_positive = (P_positive_given_disease * P_disease) /␣
  ↪P_positive_test
```

[5]:
```
P_disease_given_positive
```

[5]:
```
0.3333333333333333
```

Q2

[6]:
```
data = pd.read_csv('student.csv')
features = ['age', 'income', 'student', 'credit']
```

```
target = 'computer'
```

[7]:

```
class_counts = data[target].value_counts()
total_samples = len(data)
```

[8]:

```
priors = class_counts / total_samples
likelihoods = {}
```

[9]:

```
for feature in features:
    feature_likelihoods = defaultdict(lambda: defaultdict(int))
    for (feature_value, target_value), count in data.groupby([feature, target]).
  ↪size().items():
        feature_likelihoods[feature_value][target_value] = count /␣
  ↪class_counts[target_value]
    likelihoods[feature] = feature_likelihoods
```

[10]:

```
# likelihoods
```

[11]:

```
age = int(input("Age (0, 1, 2): "))
income = int(input("Income (0 = low, 1 = medium, 2 = high): "))
student = int(input("Student (0 = no, 1 = yes): "))
credit = int(input("Credit (0 = fair, 1 = excellent): "))
sample = {
    'age': age,
    'income': income,
    'student': student,
    'credit': credit
}
```

[12]:

```
posteriors = {}
for class_value in class_counts.index:
    prior = priors[class_value]
    likelihood = prior
    for feature in features:
        feature_value = sample[feature]
        feature_likelihood = likelihoods[feature].get(feature_value, {}).
  ↪get(class_value, 1e-6)
        likelihood *= feature_likelihood
    posteriors[class_value] = likelihood
```

[13]:

```
sample
```

[13]:

```
{'age': 1, 'income': 1, 'student': 1, 'credit': 0}
```

[14]:

```
prediction = max(posteriors, key=posteriors.get)
```

[15]:

```
if prediction == 0:
    print("!!!No Computer!!!")
else:
    print("!!!Computer!!!")
```

```
!!!No Computer!!!
```

Q3

[16]:

```
training_data = pd.read_csv('sports.csv')
X = training_data['Text']
y = training_data['Tag']
```

[17]:

```
train_size = int(0.8 * len(X))
X_train, X_test = X[:train_size], X[train_size:]
y_train, y_test = y[:train_size], y[train_size:]
```

[18]:

```
class_priors = {}
word_freqs = defaultdict(lambda: defaultdict(int))
vocab = set()
```

[19]:

```
class_counts = y_train.value_counts().to_dict()
total_count = len(y_train)
```

[20]:

```
for cls in class_counts.keys():
    class_priors[cls] = class_counts[cls] / total_count
```

[21]:

```
class_priors
```

[21]:

```
{'Sports': 0.75, 'Not sports': 0.25}
```

[22]:

```
for text, cls in zip(X_train, y_train):
    words = text.lower().split()
    for word in words:
        word_freqs[cls][word] += 1
        vocab.add(word)
```

[23]:

```
vocab
```

[23]:

```
{'but',
 'clean',
 'election',
 'forgettable',
 'game"',
 'great',
 'match"',
 'over"',
 'was',
 '"a',
 '"the',
 '"very'}
```

[24]:

```
word_freqs
```

[24]:

```
defaultdict(<function __main__.<lambda>()>,
            {'Sports': defaultdict(int,
                         {'"a': 2,
                          'great': 1,
                          'game"': 2,
                          '"very': 1,
                          'clean': 2,
                          'match"': 1,
                          'but': 1,
                          'forgettable': 1}),
             'Not sports': defaultdict(int,
                         {'"the': 1, 'election': 1, 'was': 1, 'over"': 1})})
```

[25]:

```
def predict(text: str) -> str:
    words = text.lower().split()
    class_scores = {}
    for cls in class_priors.keys():
        score = np.log(class_priors[cls])
        total_words = sum(word_freqs[cls].values())
        for word in words:
            word_prob = (word_freqs[cls][word] + 1) / (total_words + len(vocab))
            score += np.log(word_prob)
        class_scores[cls] = score
    return max(class_scores, key=class_scores.get)
```

[26]:

```
y_pred = [predict(text) for text in X_test]
```

[27]:

```
print("True Labels:", y_test.tolist())
print("Predicted Labels:", y_pred)
```

```
True Labels: ['Not sports']
Predicted Labels: ['Not sports']
```

[28]:

```
y_true_list = y_test.tolist()
```

[29]:

```
correct_predictions = sum(t == p for t, p in zip(y_true_list, y_pred))
accuracy = correct_predictions / len(y_true_list)
```

[30]:

```
accuracy
```

[30]:

```
1.0
```

[31]:

```
tp = sum((t == 'Sports' and p == 'Sports') for t, p in zip(y_true_list, y_pred))
fp = sum((t != 'Sports' and p == 'Sports') for t, p in zip(y_true_list, y_pred))
fn = sum((t == 'Sports' and p != 'Sports') for t, p in zip(y_true_list, y_pred))
tn = sum((t != 'Sports' and p != 'Sports') for t, p in zip(y_true_list, y_pred))
```

[32]:

```
precision = tp / (tp + fp) if (tp + fp) > 0 else 0
recall = tp / (tp + fn) if (tp + fn) > 0 else 0
```

[33]:

```
precision
```

[33]:

```
0
```

[34]:

```
recall
```

[34]:

```
0
```

[35]:

```
new_sentence = "Game was very boring"
predicted_tag = predict(new_sentence)
print(f"The sentence '{new_sentence}' is classified as: {predicted_tag}")
```

```
The sentence 'Game was very boring' is classified as: Not sports
```