Week8

September 20, 2023

Example :- Suppose the probability of the weather being cloudy is 40%. Also suppose the probability of rain on a given day is 20%. Also suppose the probability of clouds on a rainy day is 85%. If it's cloudy outside on a given day, what is the probability that it will rain that day?

```
[7]: def bayesTheorem(pA, pB, pBA):
         return pA*pBA/pB

     pCloudy = 0.4
     pRain = 0.2
     pCloudyRain = 0.85

     bayesTheorem(pRain, pCloudy, pCloudyRain)
```

```
[7]: 0.425
```

1 Example

```
[9]: from sklearn.feature_extraction.text import CountVectorizer
     from sklearn.naive_bayes import MultinomialNB
     from sklearn.metrics import accuracy_score, classification_report
     from sklearn.model_selection import train_test_split

     # Sample data
     texts = ["This is a positive review.", "Negative sentiment detected.", "A very␣
      ↪positive experience.", "I didn't like this at all."]

     # Corresponding labels (1 for positive, 0 for negative)
     labels = [1, 0, 1, 0]

     # Split the data into a training set and a test set
     X_train, X_test, y_train, y_test = train_test_split(texts, labels, test_size=0.
      ↪2, random_state=42)

     vectorizer = CountVectorizer()
     X_train_vec = vectorizer.fit_transform(X_train)
     X_test_vec = vectorizer.transform(X_test)
```

```python
clf = MultinomialNB()
clf.fit(X_train_vec, y_train)

y_pred = clf.predict(X_test_vec)

accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)
print(f"Accuracy: {accuracy}")
print(report)
```

```
Accuracy: 0.0
              precision    recall  f1-score   support

           0       0.00      0.00      0.00       1.0
           1       0.00      0.00      0.00       0.0

    accuracy                           0.00       1.0
   macro avg       0.00      0.00      0.00       1.0
weighted avg       0.00      0.00      0.00       1.0


/usr/lib/python3/dist-packages/sklearn/metrics/_classification.py:1221:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/lib/python3/dist-packages/sklearn/metrics/_classification.py:1221:
UndefinedMetricWarning: Recall and F-score are ill-defined and being set to 0.0
in labels with no true samples. Use `zero_division` parameter to control this
behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

2 Questions

1. Implement in python of the following problems using Bayes Theorem.

a) Of the students in the college, 60% of the students reside in the hostel and 40% of the students are day scholars. Previous year results report that 30% of all students who stay in the hostel scored A Grade and 20% of day scholars scored A grade. At the end of the year, one student is chosen at random and found that he/she has an A grade. What is the probability that the student is a hosteler?

[18]:

```python
pH, pD = 0.6, 0.4
pAH, pAD = 0.3, 0.2

def bayes(h,d,ah,ad):
    return ((h*ah)/((h*ah)+(d*ad)))
```

```
prob = round((bayes(pH,pD,pAH,pAD)),4)
print(f"Probility of student being hosteller given A grade = {prob*100}%")
```

```
Probility of student being hosteller given A grade = 69.23%
```

b) Suppose you're testing for a rare disease, and you have the following information: • The disease has a prevalence of 0.01 (1% of the population has the disease). • The test is not perfect: • The test correctly identifies the disease (true positive) 99% of the time (sensitivity). • The test incorrectly indicates the disease (false positive) 2% of the time (1 - specificity). Calculate the probability of having the disease given a positive test result using Bayes' theorem.

[19]:

```
pD, pN = 0.01, 0.99
pTD, pTN = 0.99, 0.02

def bayes(d,n,td,tn):
    num = d*td
    denom = (d*td)+(n*tn)
    return num/denom

prob = bayes(pD,pN,pTD,pTN)
print(f"Probility = {prob*100}%")
```

```
Probility = 33.33333333333333%
```

2. Write a program to implement the naïve Bayesian classifier without using scikit-learn library for the following sample training data set stored as a .CSV file. Calculate the accuracy, precision, and recall for your train/test data set. To classify 'If the weather is sunny, then the Player should play or not'?

[20]:

```
import pandas as pd
data = pd.read_csv('data.csv')
data
```

[20]:

```
     Outlook Play
0      Rainy  Yes
1      Sunny  Yes
2   Overcast  Yes
3   Overcast  Yes
4      Sunny   No
5      Rainy  Yes
6      Sunny  Yes
7   Overcast  Yes
8      Rainy   No
9      Sunny   No
10     Sunny  Yes
11     Rainy   No
12  Overcast  Yes
13  Overcast  Yes
```

[27]:

```
class NaiveBayesClassifier:
    def __init__(self,X,y):
        self.X,self.y = X,y
        self.N = len(self.X) # Length of the training set
        self.dim = len(self.X[0]) # Dimension of the vector of features
        self.attrs = [[] for _ in range(self.dim)] # Here we'll store the columns of the training set

        self.output_dom = {} # Output classes with the number of ocurrences in the training set. In this case we have only 2 classes

        self.data = [] # To store every row [Xi, yi]

        for i in range(len(self.X)):
            for j in range(self.dim):
                # if we have never seen this value for this attr before,
                # then we add it to the attrs array in the corresponding position
                if not self.X[i][j] in self.attrs[j]:
                    self.attrs[j].append(self.X[i][j])

            # if we have never seen this output class before,
            # then we add it to the output_dom and count one occurrence for now
            if not self.y[i] in self.output_dom.keys():
                self.output_dom[self.y[i]] = 1
            # otherwise, we increment the occurrence of this output in the training set by 1
            else:
                self.output_dom[self.y[i]] += 1
            # store the row
            self.data.append([self.X[i], self.y[i]])

    def classify(self, entry):
        solve = None # Final result
        max_arg = -1 # partial maximum

        for y in self.output_dom.keys():

            prob = self.output_dom[y]/self.N # P(y)

            for i in range(self.dim):
                cases = [x for x in self.data if x[0][i] == entry[i] and x[1] == y] # all rows with Xi = xi
                n = len(cases)
                prob *= n/self.N # P *= P(Xi = xi)
            # if we have a greater prob for this output than the partial␣
 ↪maximum...
            if prob > max_arg:
                max_arg = prob
                solve = y

        return solve
```

[28]:

```
y = list(map(lambda v: 'Yes' if v == 1 else 'no', data['Play'].values)) #␣
 ↪target values as string
X = data[['Outlook']].values

y_train = y[:8]
y_val = y[8:]

X_train = X[:8]
X_val = X[8:]

nbc = NaiveBayesClassifier(X_train, y_train)

total_cases = len(y_val) # size of validation set

# Well classified examples and bad classified examples
good = 0
bad = 0

for i in range(total_cases):
    predict = nbc.classify(X_val[i])
#     print(y_val[i] + ' --------------- ' + predict)
    if y_val[i] == predict:
        good += 1
    else:
        bad += 1

print('TOTAL EXAMPLES:', total_cases)
print('RIGHT:', good)
print('WRONG:', bad)
print('ACCURACY:', good/total_cases)
```

```
TOTAL EXAMPLES: 6
RIGHT: 6
WRONG: 0
ACCURACY: 1.0
```

[ ]: