Final Project

CISC5450 Mathematics for Data Science

VAISHALI SHARMA

May 10, 2021

Problem 1: Using KNN, predict whether Movie Fargo is recommended to Ken or not?

NAME	THE PIANO	PULP FICTION	CLUELESS	CLIFFHANGER	FARGO
AMY(+)	0	0	1	0	0
AMY(-)	1	1	0	1	1
JEF(+)	0	1	0	0	1
JEF(-)	1	0	0	1	0
MIKE(+)	1	1	0	1	1
MIKE(-)	0	0	1	0	0
CHRIS(+)	0	0	1	0	0
CHRIS(-)	0	1	0	1	1
KEN (CLASS)	+	+	-	+	?

Using Mathematical Calculations

K-Nearest Neighbor

K-Nearest Neighbor (or KNN) is a supervised machine learning algorithm useful for classification problems. It calculates the distance between the test data and the input and gives the prediction according.

Euclidean Distance is calculated as:

$$dist = \sqrt{\sum_{k=1}^{n}(p_k - q_k)^2}$$

Where, n is the number of dimensions (attributes) and p_k and q_k are, respectively, the k^{th} attributes (components) or data objects/points p and q.

The above formula takes in n number of dimensions or here we can say them as our features in machine learning. The data point which is located at the minimum distance from the test point is assumed to belong to the same class. The above formula works the same in n number of dimensions and therefore it can be used with n number of features.

Calculation Method

We usually use Euclidean distance to calculate the nearest neighbor. If we have two points (x, y) and (a, b). The formula for Euclidean distance (d) will be,

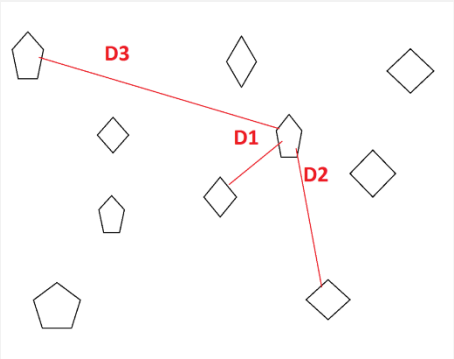
Final Project

$$d = \text{sqrt}((x-a)^2+(y-b)^2)$$

We try to get the smallest Euclidean distance and based on the number of smaller distances we perform our calculation.

if D1 < D2 and D1<D3

then, we will choose D1 as the minimum Euclidean distance or the closest point and make our predictions.



Euclidean Distance

Problem Solution

Above method is used to make the movie recommendations for Ken. Following steps are taken to make the predictions:

Step1: Compute the square of the distance between the two points.

FARGO as x, THE PIANO as 'a', PULP FICTION as 'b', CLUELESS as 'c', CLIFFHANGER as 'd'

Distances	$(x-a)^2$	$(x-b)^2$	$(x-c)^2$	$(x-d)^2$
AMY(+)	0	0	1	0
AMY(-)	0	0	1	0
JEF(+)	1	0	1	1
JEF(-)	1	0	0	1
MIKE(+)	0	0	1	0
MIKE(-)	0	0	1	0
CHRIS(+)	0	0	1	0
CHRIS(-)	1	0	1	0

Step2: Compute the square root of the sum of square of the distance between the two points.

FARGO as x, THE PIANO as 'a', PULP FICTION as 'b', CLUELESS as 'c', CLIFFHANGER as 'd'

Distances	$(x-a)^2$	$(x-b)^2$	$(x-c)^2$	$(x-d)^2$
AMY(+)	0	0	1	0
AMY(-)	0	0	1	0
JEF(+)	1	0	1	1
JEF(-)	1	0	0	1
MIKE(+)	0	0	1	0
MIKE(-)	0	0	1	0
CHRIS(+)	0	0	1	0
CHRIS(-)	1	0	1	0
Sum of square of distances	3	0	7	2
Square root of Sum of Square of distances	$\sqrt{3}$ = 1.73	$\sqrt{0}$ = 0	$\sqrt{7}$ = 2.65	$\sqrt{2}$ =1.41

As per the calculation,

distance between Fargo and The Piano is = $\sqrt{3}$ = 1.73

Final Project

distance between Fargo and Pulp Fiction is $\sqrt{0} = 0$

distance between Fargo and Clueless is = $\sqrt{7} = 2.65$

distance between Fargo and Cliffhanger is = $\sqrt{2} = 1.41$

Step 3: Compute average distance of positive(+) class and negative(-) class movie.

NAME	THE PIANO	PULP FICTION	CLUELESS	CLIFFHANGER	FARGO
KEN (CLASS)	+	+	-	+	?

Average distance of positive class = [distance between Fargo and The Piano + distance between Fargo and Pulp Fiction + distance between Fargo and Cliffhanger] / 3

= [1.73 + 0 + 1.41] / 3

= 1.0466

Since, there is only one negative class so average will be equal to the distance between Fargo and Clueless

Average distance of negative class = 2.65

Step 4: Compare the average distance of positive and negative class and on the basis of that, make the movie prediction for Ken.

Since, the average distance of + class < average distance of negative class (i.e., 1.0466 < 2.64). So, **Fargo movie belongs to positive class** and is recommended to Ken.

Step 5: Record predictions in the table.

Name	THE PIANO	PULP FICTION	CLUELESS	CLIFFHANGER	FARGO
AMY(+)	0	0	1	0	0
AMY(-)	1	1	0	1	1
JEF(+)	0	1	0	0	1
JEF(-)	1	0	0	1	0
MIKE(+)	1	1	0	1	1
MIKE(-)	0	0	1	0	0
CHRIS(+)	0	0	1	0	0
CHRIS(-)	0	1	0	1	1
KEN (CLASS Prediction for FARGO)	+	+	-	+	**+ (Recommended)**

Final Result: Based on the calculation, FARGO can be recommended to Ken

Final Project

Developing Python Code for KNN

```
import numpy as np
import pandas as pd
import math
```

```
#Movie DataSet
Piano = [0,1,0,1,1,0,0,0]
Pulp_fiction = [0,1,1,0,1,0,0,1]
Clueless = [1,0,0,0,0,1,1,0]
Cliffhanger = [0,1,0,1,1,0,0,1]
Fargo = [0,1,1,0,1,0,0,1]
```

```
#Computing the euclidean_distance between two points

def euclidean_distance(Movie1, Movie2):
    sum_of_squares_of_distances =np.sum(((np.array(Movie1))- (np.array(Movie2))) ** 2)
    sqrt_sumofsquares = math.sqrt(sum_of_squares_of_distances)
    return sqrt_sumofsquares

print("\nComputing Euclidean Distance between two movies.............")
distF_P = euclidean_distance(Fargo, Piano)
print("\nEuclidean distance between Fargo and Piano is:  ", distF_P)
distF_PF = euclidean_distance(Fargo, Pulp_fiction)
print("Euclidean distance between Fargo and Pulp Fiction is: ", distF_PF)
distF_C = euclidean_distance(Fargo, Clueless)
print("Euclidean distance between Fargo and Clueless is:  ", distF_C)
distF_CL = euclidean_distance(Fargo, Cliffhanger)
print("Euclidean distance between Fargo and Cliffhanger is:  ", distF_CL)

print("\nFinding Class and average ratings from the distances..........")
Ken = [1,1,0,1]
if Ken[i] == 1:
    Positive_Class=[distF_P, distF_PF, distF_CL]
    print("\nThe elements in positive class are:  ", (Positive_Class))
    average_ratings_positive_class = np.mean(Positive_Class)
    print("Average ratings for positive class is:  ", average_ratings_positive_class)

negative_class=[distF_C]
print("The elements in negative class are:   ",  negative_class)
print("Average ratings for negative class is:  ", negative_class)
```

```
Computing Euclidean Distance between two movies.............

Euclidean distance between Fargo and Piano is:   1.7320508075688772
Euclidean distance between Fargo and Pulp Fiction is:  0.0
Euclidean distance between Fargo and Clueless is:   2.6457513110645907
Euclidean distance between Fargo and Cliffhanger is:   1.4142135623730951

Finding Class and average ratings from the distances..........

The elements in positive class are:   [1.7320508075688772, 0.0, 1.4142135623730951]
Average ratings for positive class is:   1.0487547899806575
The elements in negative class are:    [2.6457513110645907]
Average ratings for negative class is:   [2.6457513110645907]
```

```
#Searching for Recommendations
print("Searching for recommendations......")
if average_ratings_positive_class > negative_class:
    print("Fargo is not recommended to Ken")
else:
    print("Fargo is recommended to Ken")
```

```
Searching for recommendations......
Fargo is recommended to Ken
```

Final Project

Problem 2: Using Naïve Bayes, to predict whether Movie Fargo is recommended to Ken or not?

NAME	THE PIANO	PULP FICTION	CLUELESS	CLIFFHANGER	FARGO
AMY(+)	0	0	1	0	0
AMY(-)	1	1	0	1	1
JEF(+)	0	1	0	0	1
JEF(-)	1	0	0	1	0
MIKE(+)	1	1	0	1	1
MIKE(-)	0	0	1	0	0
CHRIS(+)	0	0	1	0	0
CHRIS(-)	0	1	0	1	1
KEN (CLASS)	+	+	-	+	?

Using Mathematical Calculations

Naïve Bayes

Naive Bayes classifiers are a collection of classification algorithms based on Bayes' Theorem. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other. The fundamental Naive Bayes assumption is that each makes an independent equal contribution to the outcome.

It works on the Bayes' Theorem.

Bayes' Theorem finds the probability of an event occurring given the probability of another event that has already occurred. Bayes' theorem is stated mathematically as the following equation:

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

where A and B are events and P(B) ? 0.

- Basically, we are trying to find probability of event A, given the event B is true. Event B is also termed as **evidence**.
- P(A) is the **priori** of A (the prior probability, i.e. Probability of event before evidence is seen). The evidence is an attribute value of an unknown instance(here, it is event B).
- P(A|B) is a posteriori probability of B, i.e. probability of event after evidence is seen.

Naïve Assumption to the Bayes' Theorem:

Let's put a naive assumption to the Bayes' theorem, which is, independence among the features. So now, split evidence into the independent parts.

Now, if any two events A and B are independent, then

P(A,B) = P(A) P(B)

Hence, $P(y|x_1, ..., x_n) = \frac{P(x_1|y)P(x_2|y)...P(x_n|y)P(y)}{P(x_1)P(x_2)...P(x_n)}$

which can be expressed as:

$$P(y|x_1,...,x_n) = \frac{P(y)\prod_{i=1}^{n} P(x_i|y)}{P(x_1)P(x_2)...P(x_n}$$

Now, as the denominator remains constant for a given input, we can remove that term:

$$P(y|x_1,...,x_n) \propto P(y)\prod_{i=1}^{n} P(x_i|y)$$

Now, let's create a classifier model. For this, we find the probability of given set of inputs for all possible values of the class variable y and pick up the output with maximum probability. This can be expressed mathematically as:

$$y = argmax_y P(y)\prod_{i=1}^{n} P(x_i|y)$$

Now, the task of calculating P(y) and P(xi | y). Please note that P(y) is also called class probability and P(xi | y) is called conditional probability. The different naive Bayes classifiers differ mainly by the assumptions they make regarding the distribution of P(xi | y).

Let's apply Naïve Bayes to the given problem:

NAME	THE PIANO	PULP FICTION	CLUELESS	CLIFFHANGER	Class
AMY(+)	0	0	1	0	0
AMY(-)	1	1	0	1	1
JEF(+)	0	1	0	0	1
JEF(-)	1	0	0	1	0
MIKE(+)	1	1	0	1	1
MIKE(-)	0	0	1	0	0
CHRIS(+)	0	0	1	0	0
CHRIS(-)	0	1	0	1	1
KEN (CLASS)	+	+	-	+	?

Problem Solution:

THE PIANO	PULP FICTION	CLUELESS	CLIFFHANGER	Class(FARGO)
1	1	0	1	1
0	1	0	0	1
1	1	0	1	1
0	1	0	1	1
THE PIANO	PULP FICTION	CLUELESS	CLIFFHANGER	Class(FARGO)
0	0	1	0	0
1	0	0	1	0
0	0	1	0	0
0	0	1	0	0

P(C=1) = 4/8 = 1/2

P(C=0) = 4/8 = 1/2

Final Project

Without Laplace Smoothing:

	Class=1	Class=0	P(PIANO| C=1)	P(PIANO| C=0)
PIANO=1	2	1	2/4	1/4
PIANO=0	2	3	2/4	3/4

	Class=1	Class=0	P(PULP FICTION| C=1)	P(PULP FICTION| C=0)
PULP FICTION=1	4	0	4/4	0/4
PULP FICTION=0	0	4	0/4	1/4

	Class=1	Class=0	P(CLUELESS| C=1)	P(CLUELESS| C=0)
CLUELESS=1	0	3	0/4	3/4
CLUELESS=0	4	1	4/4	1/4

	Class=1	Class=0	P(CLIFFHANGER| C=1)	P(CLIFFHANGER| C=0)
CLIFFHANGER=1	3	1	3/4	1/4
CLIFFHANGER=0	1	3	1/4	3/4

Because there are zero probability values in the above table, we need to use Laplace Smoothing

Applying Laplace Smoothing:

	Class=1	Class=0	P(PIANO| C=1)	P(PIANO| C=0)
PIANO=1	2	1	(2+1)/(4+4)	(1+1)/(4+4)
PIANO=0	2	3	(2+1)/(4+4)	(3+1)/(4+4)

	Class=1	Class=0	P(PULP FICTION| C=1)	P(PULP FICTION| C=0)
PULP FICTION=1	4	0	(4+1)/(4+4)	(0+1)/(4+4)
PULP FICTION=0	0	4	(0+1)/(4+4)	(1+1)/(4+4)

	Class=1	Class=0	P(CLUELESS| C=1)	P(CLUELESS| C=0)
CLUELESS=1	0	3	(0+1)/(4+4)	(3+1)/(4+4)
CLUELESS=0	4	1	(4+1)/(4+4)	(1+1)/(4+4)

	Class=1	Class=0	P(CLIFFHANGER| C=1)	P(CLIFFHANGER| C=0)
CLIFFHANGER=1	3	1	(3+1)/(4+4)	(1+1)/(4+4)
CLIFFHANGER=0	1	3	(1+1)/(4+4)	(3+1)/(4+4)

Final Project

Given Test data

KEN	PIANO=1	PULP FICTION=1	CLUELESS=0	CLIFFHANGER=1	???

Should FARGO be recommended to Ken or not?

Calculations of naïve bayes probabilities:

P(C=1|PIANO=1, PULP FICTION=1, CLUELESS=0, CLIFFHANGER=1)= P(C=1)* P(PIANO=1, PULP FICTION=1, CLUELESS=0, CLIFFHANGER=1|C=1)

= P(C=1) * P(PIANO=1|C=1) * P(PULP FICTION=1|C=1) * P(CLUELESS=0|C=1) * P(CLIFFHANGER=1|C=1)

= (1/2) * (3/8) * (5/8) * (5/8) * (4/8)

=0.03662

P(C=0|PIANO=1, PULP FICTION=1, CLUELESS=0, CLIFFHANGER=1)= P(C=0)* P(PIANO=1, PULP FICTION=1, CLUELESS=0, CLIFFHANGER=1|C=0)

= P(C=0) * P(PIANO=1|C=0) * P(PULP FICTION=1|C=0) * P(CLUELESS=0|C=0) * P(CLIFFHANGER=1|C=0)

= (1/2) * (2/8) * (1/8) * (2/8) * (2/8)

=0.0009

Since, **P(C=1|PIANO=1, PULP FICTION=1, CLUELESS=0, CLIFFHANGER=1) > P(C=0|PIANO=1, PULP FICTION=1, CLUELESS=0, CLIFFHANGER=1)**

Therefore, Class (Fargo)=1. Recommend movie FARGO to Ken.

Final Project

Developing Python Code for Naïve Bayes

```
#Dataset used for the study
Piano = [0,1,0,1,1,0,0,0]
Pulp_fiction = [0,1,1,0,1,0,0,1]
Clueless = [1,0,0,0,0,1,1,0]
Cliffhanger = [0,1,0,1,1,0,0,1]
Fargo = [0,1,1,0,1,0,0,1]
```

```
movie_dataset = [
    [1,1,0,1,1],
    [0,1,0,0,1],
    [1,1,0,1,1],
    [0,1,0,1,1],
    [0,0,1,0,0],
    [1,0,0,1,0],
    [0,0,1,0,0],
    [0,0,1,0,0]
    ]
```

```
#Seperating using class
def separate_by_class(movie_dataset):
    separated = dict()
    for i in range(len(movie_dataset)):
        row = movie_dataset[i]
        value = row[-1]
        if (value not in separated):
            separated[value] = list()
        separated[value].append(row)
    return separated
separated = separate_by_class(movie_dataset)
for label in separated:
    print(label)
    for values in separated[label]:
        print(values)
```

```
1
[1, 1, 0, 1, 1]
[0, 1, 0, 0, 1]
[1, 1, 0, 1, 1]
[0, 1, 0, 1, 1]
0
[0, 0, 1, 0, 0]
[1, 0, 0, 1, 0]
[0, 0, 1, 0, 0]
[0, 0, 1, 0, 0]
```

```
Ken = [1,1,0,1]
```

```
#Compute total positive class
prob_positive_class = 1
count = 0
add = 0
for row in movie_data:
    if(row[-1] == 1):
        count+=1
    add+=1
print("Total Positive Class: "+str(count)+" / "+str(add))
prob_positive_class *= count/add
```

```
Total Positive Class: 4 / 8
```

```
#Compute total Negative class
prob_negative_class = 0
count = 0
add = 0
for row in movie_data:
    if(row[-1] == 0):
        count+=1
    add+=1
print("Total Negative Class: "+str(count)+" / "+str(add))
prob_negative_class *= count/add
```

```
Total Negative Class: 4 / 8
```

Final Project

```
#What is the probability of all movies with positive ratings
for i in range(len(Ken)):
    count = 1
    add = 4
    for row in ydataset[1]:
        if(Ken[i] == row[i]):
            count += 1
        add += 1
    print('FOR MOVIE ' + str(i+1))
    print(str(count)+" / " +str(add))
    prob_positive_class *= count/add
print(prob_positive_class)
```

```
FOR MOVIE 1
3 / 8
FOR MOVIE 2
5 / 8
FOR MOVIE 3
5 / 8
FOR MOVIE 4
4 / 8
0.03662109375
```

```
#What is the probability of all movies with negative ratings
Ken = [1,1,0,1]
for i in range(len(Ken)):
    count = 0
    add = 0
    for row in ydataset[0]:
        if(Ken[i] == row[i]):
            count += 1
        add += 1
    print('FOR MOVIE ' + str(i+1))
    print(str(count+1)+" / " + str(add+4))
    prob_negative_class *= count/add
```

```
FOR MOVIE 1
2 / 8
FOR MOVIE 2
1 / 8
FOR MOVIE 3
2 / 8
FOR MOVIE 4
2 / 8
```

```
#Searching for Recommendations
print("Searching for recommendations......")
if prob_positive_class > prob_negative_class:
    print("Fargo is recommended to Ken")
else:
    print("Fargo is not recommended to Ken")
```

```
Searching for recommendations......
Fargo is recommended to Ken
```