30. Data Science – Machine Learning – Naïve Bayes Classifier

Contents

30. Data Science – Machine Learning – Naïve Bayes Classifier

1. Naive Bayes Classifier

- ✓ Naïve Bayes algorithm is a supervised learning algorithm.
- ✓ This is based on Bayes theorem and used for solving classification problems.
- ✓ It is mainly used in text classification.
    - Examples like spam filtration, Sentimental analysis, and classifying articles etc.

2. Why is it called Naïve Bayes?

- ✓ **Naive:**
    - It is called Naive because it assumes that the occurrence of a certain feature is independent of the other features.
    - Such as if the fruit is identified on the bases of color, shape, and taste, then red, spherical, and sweet fruit is recognized as an apple.
    - Hence each feature individually contributes to identify that it is an apple without depending on each other.
- ✓ **Bayes:**
    - It is called Bayes because it depends on the principle of Bayes' Theorem.

3. Bayes theorem

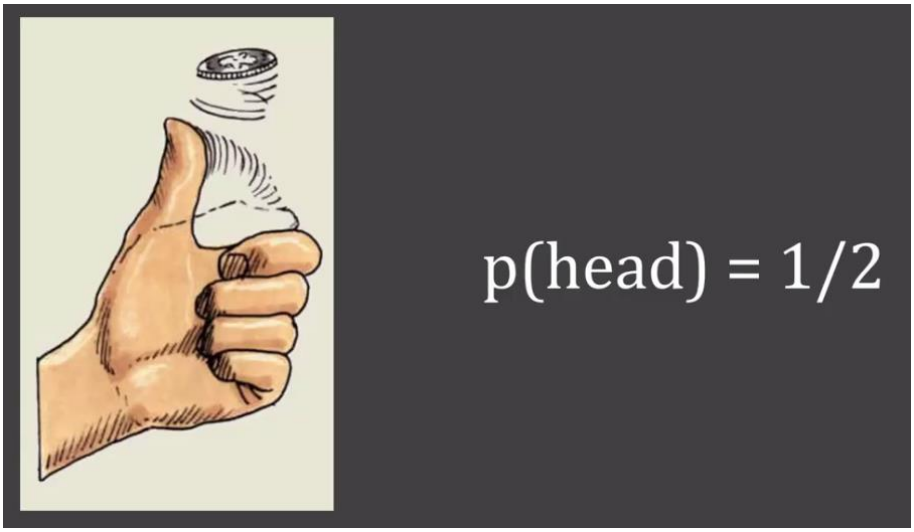
- ✓ Bayes' theorem is also known as Bayes' Rule or Bayes' law.
- ✓ It is used to determine the probability of a hypothesis with prior knowledge.
- ✓ It depends on the conditional probability.

$$P(A/B) = \frac{P(B/A) * P(A)}{P(B)}$$

Thomas Bayes

4. Scenario

- ✓ When we flip a coin, the probability of getting head or trail is 1/2, because there are two possibilities of outcomes
- ✓ So the chance of getting head or tail is 50%



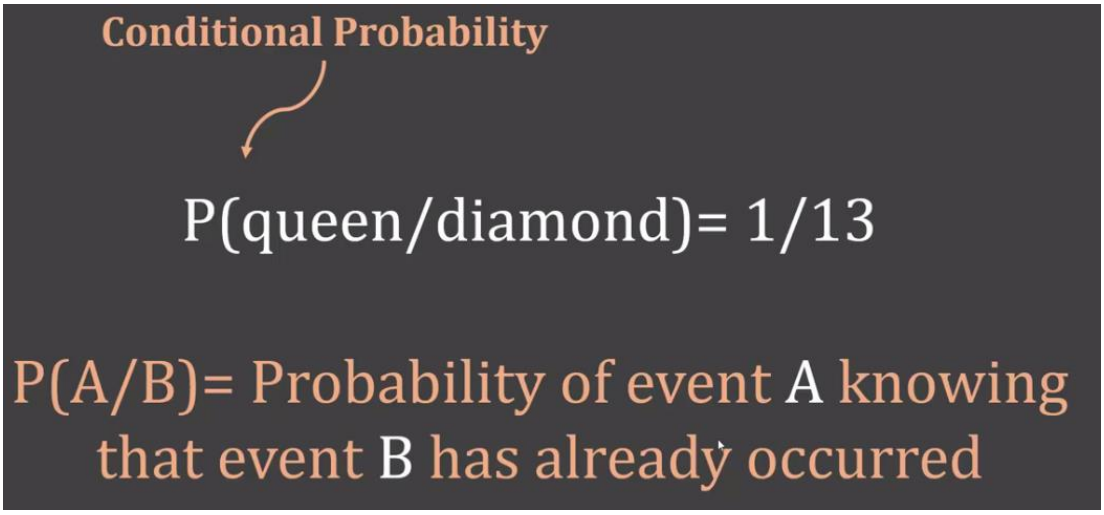
Pick a random card, what is the probability of getting a queen?

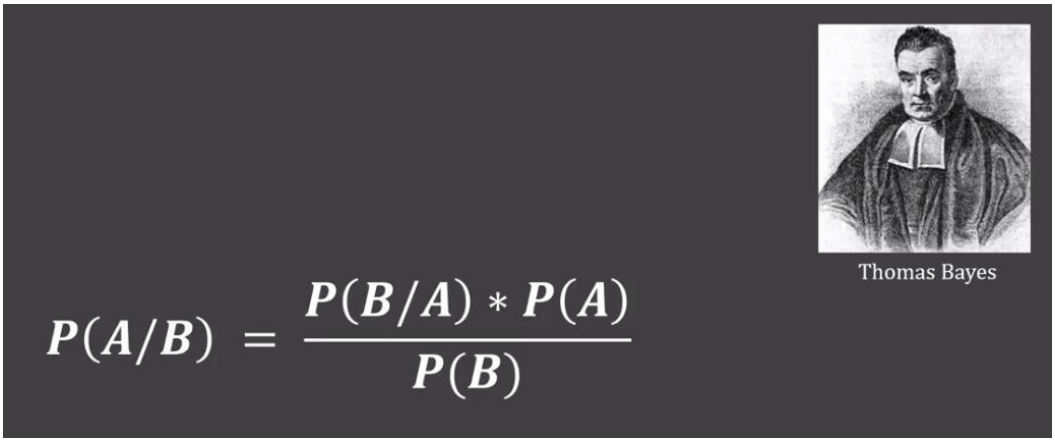

4 queens, 52 total cards
P(queen) = 4/52 = 1/13

5. Conditional probability

Pick a random card, you know it is a diamond. Now what is the probability of that card being a queen?


Total diamonds = 13
Queen = 1
P(queen/diamond)= 1/13


Conditional Probability
P(queen/diamond)= 1/13
P(A/B)= Probability of event A knowing that event B has already occurred



$$P(queen/diamond) = \frac{P(diamond/queen) * P(queen)}{P(diamond)}$$

P(diamond/queen) = 1/4

P(queen) = 1/13

P(diamond) = 1/4

$$= \frac{1/4 * 1/13}{1/4}$$

$$= 1/13$$

6. Use case

Passenger Id	Name	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Survived
1	Braund, Mr. Owen Harris	3	male	22	1	0	21171	7.25		S	0
2	Cumings, Mrs. John Bradley	1	female	38	1	0	17599	71.2833	C85	C	1
3	Heikkinen, Miss. Laina	3	female	26	0	0	3101282	7.925		S	1
4	Futrelle, Mrs. Jacques Heath	1	female	35	1	0	113803	53.1	C123	S	1
5	Allen, Mr. William Henry	3	male	35	0	0	373450	8.05		S	0
6	Moran, Mr. James	3	male		0	0	330877	8.4583		Q	0
7	McCarthy, Mr. Timothy J	1	male	54	0	0	17463	51.8625	E46	S	0
8	Palsson, Master. Gosta Leonard	3	male	2	3	1	349909	21.075		S	0
9	Johnson, Mrs. Oscar	3	female	27	0	2	347742	11.1333		S	1
10	Nasser, Mrs. Nicholas	2	female	14	1	0	237736	30.0708		C	1

Passenger Id	Name	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Survived
1	Braund, Mr. Owen Harris	3	male	22	1	0	21171	7.25		S	0
2	Cumings, Mrs. John Bradley	1	female	38	1	0	17599	71.2833	C85	C	1
3	Heikkinen, Miss. Laina	3	female	26	0	0	3101282	7.925		S	1
4	Futrelle, Mrs. Jacques Heath	1	female	35	1	0	113803	53.1	C123	S	1
5	Allen, Mr. William Henry	3	male	35	0	0	373450	8.05		S	0
6	Moran, Mr. James	3	male		0	0	330877	8.4583		Q	0
7	McCarthy, Mr. Timothy J	1	male	54	0	0	17463	51.8625	E46	S	0
8	Palsson, Master. Gosta Leonard	3	male	2	3	1	349909	21.075		S	0
9	Johnson, Mrs. Oscar	3	female	27	0	2	347742	11.1333		S	1
10	Nasser, Mrs. Nicholas	2	female	14	1	0	237736	30.0708		C	1

$$P\left(\frac{Survived}{Male \,\&\, Class \,\&\, Age \,\&\, Cabin \,\&\, Fare}\right)$$

Make a naïve assumption that features such as male, class, age , cabin, fare etc. are independent of each other

7. Math problem and solution

Problem

- ✓ If the weather is sunny, then the Player should play or not?

Solution

- ✓ To solve this, first consider the below dataset

	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

Frequency table for the Weather Conditions:

Weather	Yes	No
Overcast	5	0
Rainy	2	2
Sunny	3	2
Total	10	5

Likelihood table weather condition:

Weather	No	Yes	
Overcast	0	5	5/14= 0.35
Rainy	2	2	4/14=0.29
Sunny	2	3	5/14=0.35
All	4/14=0.29	10/14=0.71	

Applying Bayes'theorem:

P(Yes|Sunny)= P(Sunny|Yes)*P(Yes)/P(Sunny)

P(Sunny|Yes)= 3/10= 0.3

P(Sunny)= 0.35

P(Yes)=0.71

So P(Yes|Sunny) = 0.3*0.71/0.35= **0.60**

P(No|Sunny)= P(Sunny|No)*P(No)/P(Sunny)

P(Sunny|NO)= 2/4=0.5

P(No)= 0.29

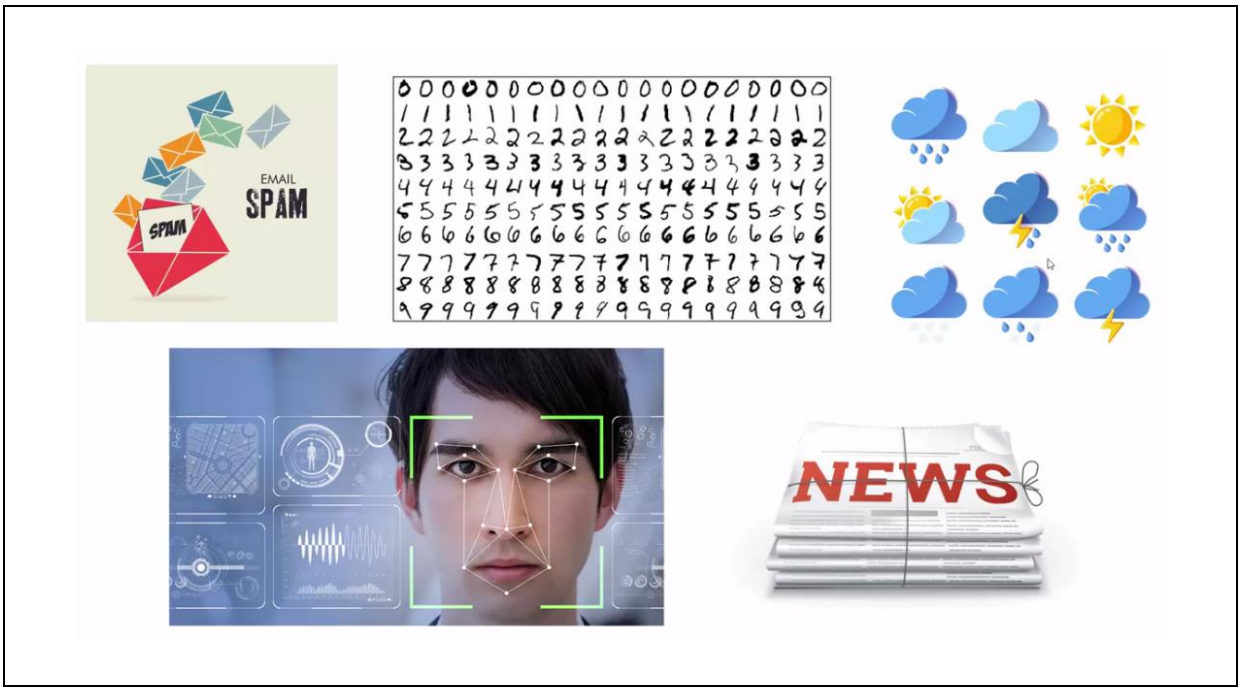
P(Sunny)= 0.35

So P(No|Sunny)= 0.5*0.29/0.35 = **0.41**

So as we can see from the above calculation that **P(Yes|Sunny)>P(No|Sunny)**

Hence on a Sunny day, Player can play the game.

8. Use cases



Program Name: Loading the dataset
demo1.py

```
import pandas as pd

df = pd.read_csv("titanic.csv")

print(df.head())
```

Output

```
   Pclass     Sex   Age     Fare  Survived
0       3    male  22.0   7.2500         0
1       1  female  38.0  71.2833         1
2       3  female  26.0   7.9250         1
3       1  female  35.0  53.1000         1
4       3    male  35.0   8.0500         0
```

Program Name: Preparing input
demo2.py

```
import pandas as pd

df = pd.read_csv('titanic.csv', usecols = ['Pclass', 'Sex', 'Age', 'Fare',
'Survived'])
inputs = df.drop('Survived', axis='columns')

print(inputs.head())
```

Output

```
   Pclass     Sex   Age     Fare
0       3    male  22.0   7.2500
1       1  female  38.0  71.2833
2       3  female  26.0   7.9250
3       1  female  35.0  53.1000
4       3    male  35.0   8.0500
```

Program Name: Preparing target
demo3.py

```
import pandas as pd

df = pd.read_csv('titanic.csv', usecols = ['Pclass', 'Sex', 'Age', 'Fare', 'Survived'])
inputs = df.drop('Survived', axis='columns')

target = df.Survived

print(target)
```

Output

```
0      0
1      1
2      1
3      1
4      0
      ..
886    0
887    1
888    0
889    1
890    0
Name: Survived, Length: 891, dtype: int64
```

Program Name: Creating dummy variables
demo4.py

```
import pandas as pd

df = pd.read_csv('titanic.csv', usecols = ['Pclass', 'Sex', 'Age', 'Fare',
'Survived'])
inputs = df.drop('Survived', axis = 'columns')
target = df.Survived

dummies = pd.get_dummies(inputs.Sex, dtype = int)

print(dummies.head())
```

Output

```
   female  male
0       0     1
1       1     0
2       1     0
3       1     0
4       0     1
```

Program Name: Concatenating the dataframe
demo5.py

```
import pandas as pd

df = pd.read_csv('titanic.csv', usecols = ['Pclass', 'Sex', 'Age', 'Fare',
'Survived'])

inputs = df.drop('Survived', axis='columns')
target = df.Survived
dummies = pd.get_dummies(inputs.Sex, dtype = int)

inputs = pd.concat([inputs, dummies], axis = 'columns')

print(inputs.head())
```

Output

```
   Pclass     Sex   Age     Fare  female  male
0       3    male  22.0   7.2500       0     1
1       1  female  38.0  71.2833       1     0
2       3  female  26.0   7.9250       1     0
3       1  female  35.0  53.1000       1     0
4       3    male  35.0   8.0500       0     1
```

Program Name: Dropping unnecessary column
demo6.py

```
import pandas as pd

df = pd.read_csv('titanic.csv', usecols = ['Pclass', 'Sex', 'Age', 'Fare',
'Survived'])

inputs = df.drop('Survived', axis='columns')
target = df.Survived
dummies = pd.get_dummies(inputs.Sex, dtype = int)
inputs = pd.concat([inputs, dummies], axis = 'columns')

inputs.drop(['Sex', 'male'], axis = 'columns', inplace = True)

print(inputs.head())
```

Output

```
   Pclass   Age     Fare  female
0       3  22.0   7.2500       0
1       1  38.0  71.2833       1
2       3  26.0   7.9250       1
3       1  35.0  53.1000       1
4       3  35.0   8.0500       0
```

Program Name: Checking empty values
demo7.py

```
import pandas as pd

df = pd.read_csv('titanic.csv', usecols = ['Pclass', 'Sex', 'Age', 'Fare',
'Survived'])

inputs = df.drop('Survived', axis = 'columns')
target = df.Survived
dummies = pd.get_dummies(inputs.Sex, dtype = int)
inputs = pd.concat([inputs, dummies], axis = 'columns')
inputs.drop(['Sex', 'male'],axis='columns', inplace = True)

print(inputs.columns[inputs.isna().any()])
```

Output

```
Index(['Age'], dtype='object')
```

Program Name: Checking empty values
demo8.py

```
import pandas as pd

df = pd.read_csv('titanic.csv', usecols = ['Pclass', 'Sex', 'Age', 'Fare',
'Survived'])

inputs = df.drop('Survived', axis='columns')
target = df.Survived
dummies = pd.get_dummies(inputs.Sex, dtype = int)

inputs = pd.concat([inputs, dummies], axis = 'columns')

inputs.drop(['Sex', 'male'],axis='columns', inplace=True)

print(inputs.Age)
```

Output

```
0      22.0
1      38.0
2      26.0
3      35.0
4      35.0
       ...
886    27.0
887    19.0
888     NaN
889    26.0
890    32.0
Name: Age, Length: 891, dtype: float64
```

Program Name: Filling Age Nan values with mean

demo9.py

```
import pandas as pd

df = pd.read_csv('titanic.csv', usecols = ['Pclass', 'Sex', 'Age', 'Fare',
'Survived'])

inputs = df.drop('Survived', axis='columns')
target = df.Survived
dummies = pd.get_dummies(inputs.Sex, dtype = int)
inputs = pd.concat([inputs, dummies], axis = 'columns')
inputs.drop(['Sex', 'male'],axis='columns', inplace=True)
inputs.Age = inputs.Age.fillna(inputs.Age.mean())

print(inputs)
```

Output

```
     Pclass        Age     Fare  female
0         3  22.000000   7.2500       0
1         1  38.000000  71.2833       1
2         3  26.000000   7.9250       1
3         1  35.000000  53.1000       1
4         3  35.000000   8.0500       0
..      ...        ...      ...     ...
886       2  27.000000  13.0000       0
887       1  19.000000  30.0000       1
888       3  29.699118  23.4500       1
889       1  26.000000  30.0000       0
890       3  32.000000   7.7500       0

[891 rows x 4 columns]
```

Program Name: Splitting the dataset
demo10.py

```
import pandas as pd
from sklearn.model_selection import train_test_split

df = pd.read_csv('titanic.csv', usecols = ['Pclass', 'Sex', 'Age', 'Fare',
'Survived'])

inputs = df.drop('Survived', axis='columns')
target = df.Survived
dummies = pd.get_dummies(inputs.Sex, dtype = int)
inputs = pd.concat([inputs, dummies], axis = 'columns')
inputs.drop(['Sex', 'male'],axis='columns', inplace=True)
inputs.Age = inputs.Age.fillna(inputs.Age.mean())

X_train, X_test, y_train, y_test = train_test_split(inputs, target,
test_size=0.3)

print("Splitting the dataset")
```

Output

```
Splitting the dataset
```

Program Name: Creating model
demo11.py

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB

df = pd.read_csv('titanic.csv', usecols = ['Pclass', 'Sex', 'Age', 'Fare',
'Survived'])

inputs = df.drop('Survived', axis='columns')
target = df.Survived
dummies = pd.get_dummies(inputs.Sex, dtype = int)
inputs = pd.concat([inputs, dummies], axis = 'columns')
inputs.drop(['Sex', 'male'],axis='columns', inplace=True)
inputs.Age = inputs.Age.fillna(inputs.Age.mean())
X_train, X_test, y_train, y_test = train_test_split(inputs, target,
test_size=0.3)

model = GaussianNB()
model.fit(X_train, y_train)

print("Model got trained")
```

Output

```
Model got trained
```

Program Name: Prediction
demo12.py

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB

df = pd.read_csv('titanic.csv', usecols = ['Pclass', 'Sex', 'Age', 'Fare',
'Survived'])

inputs = df.drop('Survived', axis='columns')
target = df.Survived
dummies = pd.get_dummies(inputs.Sex, dtype = int)
inputs = pd.concat([inputs, dummies], axis = 'columns')
inputs.drop(['Sex', 'male'],axis='columns', inplace=True)
inputs.Age = inputs.Age.fillna(inputs.Age.mean())
X_train, X_test, y_train, y_test = train_test_split(inputs, target,
test_size=0.3)

model = GaussianNB()
model.fit(X_train, y_train)

print(X_test[0:5])
print(model.predict(X_test[0:5]))
```

Output

```
     Pclass    Age     Fare  female
445       1   4.00  81.8583       0
831       2   0.83  18.7500       0
294       3  24.00   7.8958       0
736       3  48.00  34.3750       1
525       3  40.50   7.7500       0
[1 0 0 1 0]
```

Program Name: Prediction probability
demo13.py

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB

df = pd.read_csv('titanic.csv', usecols = ['Pclass', 'Sex', 'Age', 'Fare',
'Survived'])

inputs = df.drop('Survived', axis='columns')
target = df.Survived
dummies = pd.get_dummies(inputs.Sex, dtype = int)
inputs = pd.concat([inputs, dummies], axis = 'columns')
inputs.drop(['Sex', 'male'],axis='columns', inplace=True)
inputs.Age = inputs.Age.fillna(inputs.Age.mean())

X_train, X_test, y_train, y_test = train_test_split(inputs, target,
test_size=0.3)

model = GaussianNB()
model.fit(X_train, y_train)

print(model.predict(X_test[0:5]))
print(model.predict_proba(X_test[:10]))
```

Output

```
[0 1 1 1 0]
[[9.36162489e-01 6.38375110e-02]
 [4.15057870e-01 5.84942130e-01]
 [1.59954910e-10 1.00000000e+00]
 [2.61425161e-01 7.38574839e-01]
 [9.38899635e-01 6.11003653e-02]]
```

Program Name: Cross validation score
demo14.py

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import cross_val_score

df = pd.read_csv('titanic.csv', usecols = ['Pclass', 'Sex', 'Age', 'Fare',
'Survived'])

inputs = df.drop('Survived', axis='columns')
target = df.Survived
dummies = pd.get_dummies(inputs.Sex, dtype = int)
inputs = pd.concat([inputs, dummies], axis = 'columns')
inputs.drop(['Sex', 'male'],axis='columns', inplace=True)
inputs.Age = inputs.Age.fillna(inputs.Age.mean())

X_train, X_test, y_train, y_test = train_test_split(inputs, target,
test_size=0.3)

model = GaussianNB()
model.fit(X_train, y_train)

model.predict(X_test[0:5])

print(cross_val_score(GaussianNB(),X_train, y_train, cv=5))
```

Output

```
[0.768      0.792      0.776      0.80645161 0.75806452]
```