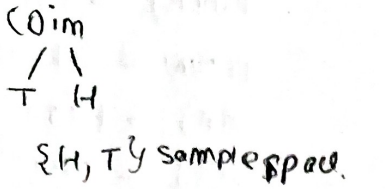
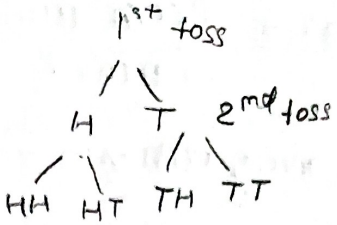
NAIVE BAYES CLASSIFIER.

Based on bayes theorem.



{H, T} sample space.



P(at least 1H) = 3/4

P(2H) = 1/4

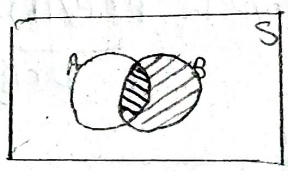
Conditional Prob P('H/'T) = 1/4

Prob of H if T is already occured.

Independent events —

A & B are independent

if $P(A \cap B) = P(A) * P(B)$ — ①



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

$P(A) = \frac{P(A \cap B)}{P(B)}$ {from ①}

$= \frac{\frac{n(A \cap B)}{n(S)}}{\frac{n(B)}{n(S)}} = \frac{n(A \cap B)}{n(B)}$

∴ P(A/B) = P(A)

as ocurence of B does not affect the ocurence of B.

Thus A & B are independent

Mutually exclusive events —

A & B are mutually exclusive

if $P(A \cap B) = 0$

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

$P(A/B) = 0$

BAYE'S THEOREM →

If there are two events A & B

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

given, $P(B) \neq 0$

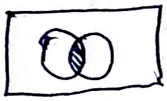
$P(A/B)$ → Posterior
$P(B/A)$ → likelihood
$P(A)$ → Prior
$P(B)$ → evidence

Proof -

Conditional Pr

$$P(A/B) = \frac{P(A \cap B)}{P(B)} \quad -①$$

$$A \cap B = B \cap A$$



$$P(B/A) = \frac{P(B \cap A)}{P(A)} = \frac{P(A \cap B)}{P(A)} \quad -②$$

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

---

ex →
if factory manufacture 100 Marker

20	30	50
made by	M2	M3
M1		
produce	3%	1%
5% defective		

given-

$P(M_1) = \frac{1}{5}$ $P(M_2) = \frac{3}{10}$ $P(M_3) = \frac{1}{2}$

$P(D/M_1) = \frac{1}{20}$ $P(D/M_2) = \frac{3}{100}$ $P(D/M_3) = \frac{1}{100}$

To find. $P(M_3/D) = \frac{P(D/M_3)\,P(M_3)}{P(D)}$

Prob that Marker is made by M_3 if it is defective.

Prob of Marker to be defective

$$P(D) = P(D \cap M_1) + P(D \cap M_2) + P(D \cap M_3)$$

$$\boxed{P(D) = P(D/M_1)P(M_1) + P(D/M_2)P(M_2) + P(D/M_3)P(M_3)}$$

Naive Baye's theorem –

CSK {lost, Mumbai, sunny}

toss	venue	Outlook	Result
won	Mumbai	overcast	won
lost	chennai	sunny	won
won	Kolkata	sunny	won
won	chennai	sunny	won
lost	Mumbai	sunny	Lost
won	chennai	overcast	Lost
won	Kolkata	overcast	Lost
won	Mumbai	Sunny	won

$P(Win/lost \cap Mumbai \cap Sunny)$

$$= \frac{P(Lost, Mumbai, Sunny/W)\,P(W)}{P(lost, Mumbai, Sunny)}$$

$P(lost/lost, Mumbai, sunny)$

$$= \frac{P(lost, Mumbai, Sunny/L)\,P(Lost)}{P(lost, Mumbai, sunny)}$$

denominator is same so we remove the denominator

$$P(W) = \frac{5}{8}$$

$$P(L) = \frac{3}{8}$$

$P(W/Lost \cap Mumbai \cap Sunny) = P(Lost, Mumbai, Sunny/W)\,P(W)$

↓ replace

$P(lost/W)\,P(Mumbai/W)\,P(Sunny/W)\,P(W)$

$1/5 * 2/5 * 4/5 * \frac{5}{8}$

Numerical Data in Naive Baye's –

{H = 185, W = 170, G = ?}

Height	Weight	Gender
172	150	M
180	170	M
165	140	M
190	200	M
39	100	F
145	120	F
160	140	F
172	150	F

$P(M/H=185, W=170) =$

$P(H=185/M)\,P(W=170/M)\,P(M)$

Let us assume that the height is gaussian distributed random variable

$$f(x) = \frac{1}{6\sqrt{2\pi}}\, e^{-1/2\left(\frac{x-\mu}{6}\right)^2}$$

or we can assume

Bimonial Distribution

Multimonial Distribution

Poisson Distribution

Sentiment Analysis.

IMDB Dataset
50,000

review	sentiment
Great movie ... again	Positive
boring mean.	negative
.	.
.	.
.	.
.	.

① Tex Preprocessing.

② Vectorize (Bag of words)

③ Passing data into ALGO
↳ Train

④ Deployment

Bag of words →

For example we have only 4 unique words in review

	"great"	"awesome"	"poor"	"horrible"	label
no of occurrence for each review upto 50,000	3	1	0	0	1
	0	1	0	4	0
	.	.	.	.	
	.	.	.	.	
	.	.	.	.	

→ In this way we use the most common words
→ This is done by SKlearn library
↳ Count Vectorizer.

→ How Naive Bayes work —

{"the movie was great, but slightly boring"} → {0, 1, 1} label = ?

awesome	great	boring	label
3	1	0	Positive
0	0	1	negative
1	2	0	positive
.	.	.	
.	.	.	
.	.	.	

DATASET

$$P(\text{label} = +ve \mid f_1 = 0, f_2 = 1, f_3 = 1) = \frac{P(f_1 = 0 \mid +ve)\, P(f_2 = 1 \mid +ve)\, P(f_3 = 1 \mid +ve)\, P(+ve)}{}$$

$$P(\text{label} = -ve \mid f_1 = 0, f_2 = 1, f_3 = 0) = \frac{P(f_1 = 0 \mid -ve)\, P(f_2 = 1 \mid -ve)\, P(f_3 = 1 \mid -ve)\, P(-ve)}{}$$

Text cleaning →

1. Sample 1000 rows
2. Remove html tags
3. Remove special tags
4. Converting every string to lowercase
5. Removing stop words { is, are, the... }
6. Stemming { play, playing, played } ⇒ play

Advantages of NB Classifier.

- easy to implement and computationally efficient
- effective in cases with a large number of features
- Performs well even with limited data.
- Performs well in the presence of categorical feature.

Disadvantages of NB classifier.

- Assumes that features are independent, which may not always hold in real world data.
- Can be influenced by irrelevant attributes
- May assign zero probability to unseen events, leading to poor generalization.

Applications of NB Classifier.

- spam email filtering
- Text classification ← sentiment analysis
- Weather Prediction
- Credit scoring
- Medical Diagnosis

Sentiment Analysis Code for IMDB Dataset.

```
import numpy as np
import pandas as pd
```
} importing libraries

```
df = pd.read_csv('...')
```
loading data

```
df.head()
df['review'][0]
df.info()
df['sentiment'].replace({'positive':1,'negative':0},inplace=True)
```

TEXT CLEANING

- Sample 10000 rows
- Remove html tags
- Remove special characters
- converting everything to lower
- Removing stop words
- stemming

```
import re
clean = re.compile('<.*?>')
re.sub(clean,'',df.iloc[2].review)

def clean_html(text):
    clean = re.compile('<.*?>')
    return re.sub(clean,'',text)

df['review'] = df['review'].apply(clean_html)
```
} Removing html tags

```
def convert_lower(text):
    return text.lower()
```
} converting text to lower

```
def remove_special(text):
    x=''
    for i in text:
        if i.isalnum():
            x=x+i
        else:
            x=x+' '
    return x
df['review'] = df['review'].apply(remove_special)
```
} Removing special character.

```
import nltk
from nltk.corpus import stopwords

stopwords.words('english')
def remove_stopwords(text):
    x=[]
    for i in text.split():
        if i not in stopwords.words('english'):
            x.append(i)
    y=x[:]
    x.clear
    return y
df['review'] = df['review'].apply(remove_stopwords)
```

Removing stop words.

```
from nltk.stem.porter import PorterStemmer
ps = PorterStemmer()

y=[]
def stem_words(text):
    for i in text:
        y.append(ps.stem(i))
    z= y[:]
    y.clear()
    return z.

def join_back(list_input):
    return " ".join(list_input)
df['review'] = df['review'].apply(join_back)
```

Stemming.

```
from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer(max_features = 1000)
X = cv.fit_transform(df['review'])
```

Vetorize Bag of words

```
y = df.iloc[:, -1].values

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)

from sklearn.naive_bayes import GaussianNB, MultinomialNB, BernoulliNB
clf1 = GaussianNB()
clf2 = MultinomialNB()
clf3 = BernoulliNB()

clf1.fit(X_train, y_train)
clf2.fit(X_train, y_train)
clf3.fit(X_train, y_train)

y_pred1 = clf1.predict(X_test)
y_pred2 = clf2.predict(X_test)
y_pred3 = clf3.predict(X_test)

from sklearn.metrics import accuracy_score.

print("Gaussian", accuracy_score(y_test, y_pred1))
print("Multinomial", accuracy_score(y_test, y_pred2))
print("Bernoulli", accuracy_score(y_test, y_pred3))
```

Training

Predicting

Accuracy score