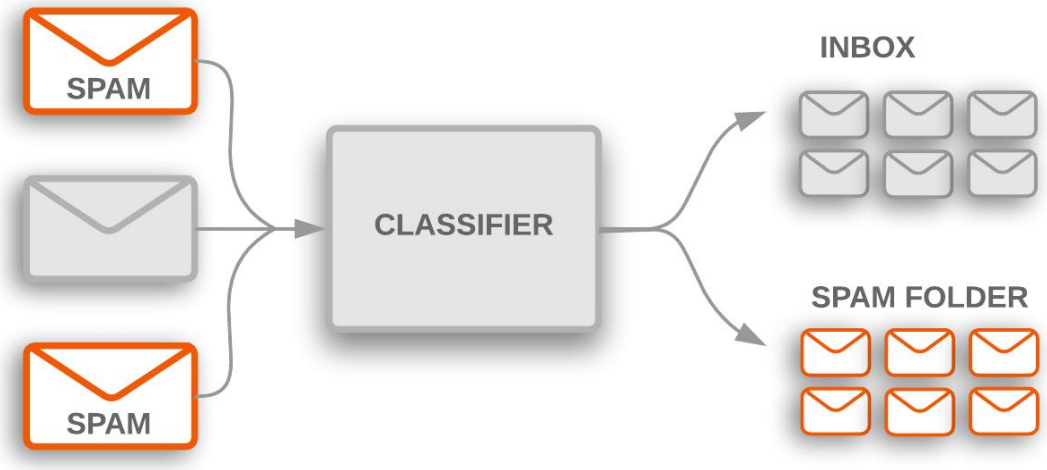
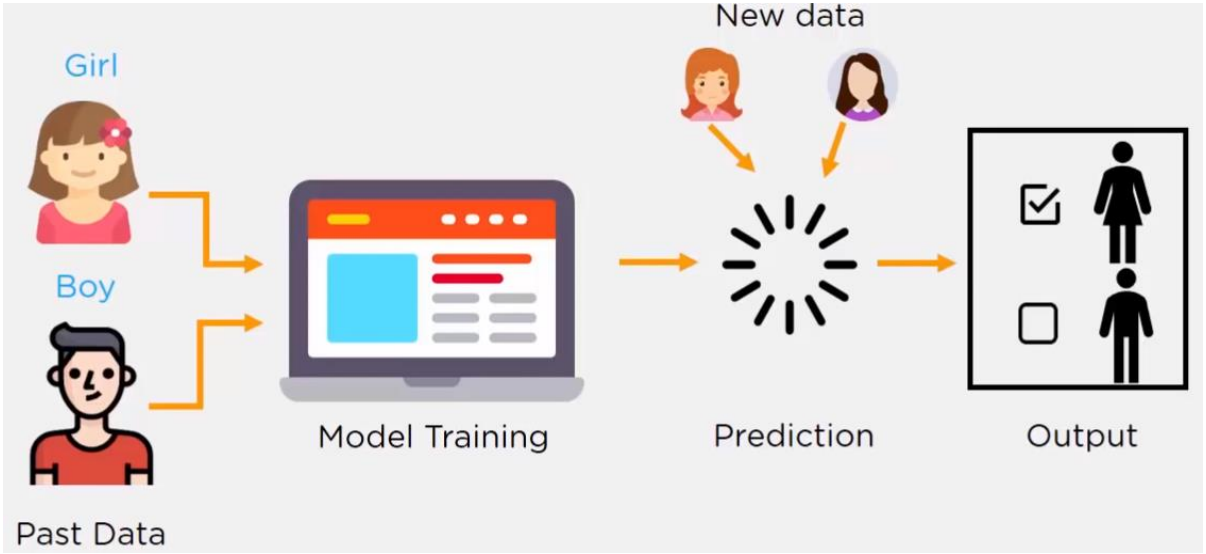
Day 3 Content

- Classification
- Naïve Bayes Algorithm
- Demo of Naive Bayes
- Text Analytics
- Spam Ham Classifier using Naïve Bayes
- Demo of Spam Ham Classifier
- Concept of Lexicons for Sentiment Analysis
- Vader Sentiment Analyzer

Classification





Naïve Bayes Classification

Probability Theory

- The measure of the likelihood that an event will occur in a Random Experiment
- Quantified as a number between 0 and 1

  0 → Impossibility

  1 → Certainty

Terminology

- **Random Experiment**

  A physical situation whose outcome cannot be predicted until it is observed.

- **Sample Space**

  A set of all possible outcomes of a random experiment

- **Conditional Probability P(A|B)**

  A measure of the probability of an event given that another event has already occurred.

- **Independence**

  One event it doesn’t affect the probability of the other.

Bayes Theorem

- To determine the probability of a hypothesis with prior knowledge.
- Depends on the conditional probability.

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

where,

P(A|B) → Posterior probability

P(B|A) → Likelihood probability

P(A) → Prior Probability

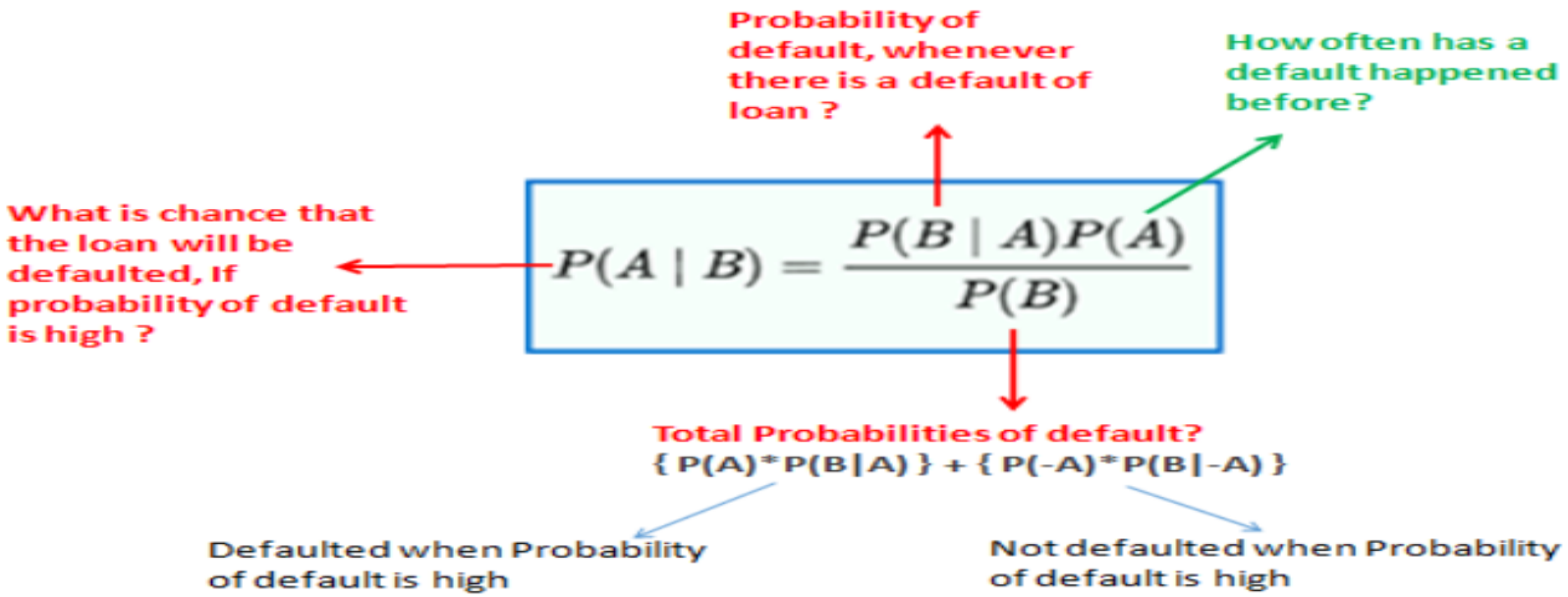
P(B) → Marginal Probability

Naive Bayes Theorem

- Supervised learning algorithm.
- Based on **Bayes theorem.**
- Helps in building the fast ML models that can make quick predictions.
- Probabilistic classifier
- **Examples:** Spam filtration, Sentimental analysis, Classifying articles etc.

Naive Bayes Theorem Example

Bank Fraud/ Loan Default



https://lh6.googleusercontent.com/ASB3dMyN-QOiJV5SH9MSk4SNcX6VI83jIkg5y9qYEL31gr3MAqfqpYXzpOSikay5fKZDEV6Mt6E5DGwGf7Bp-Ji6_8V1z9wacKM1QKvipm9UiFLV4gUu3TyElHTjrdpvo5J8vwef

Types of Naive Bayes Model

- Gaussian
- Multinomial
- Bernoulli

Naive Bayes Classifier Implementation

- Building the classifier and testing the output.

```
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train, y_train)

y_pred = classifier.predict(X_test)
```

- Summary of the predictions made by the classifier

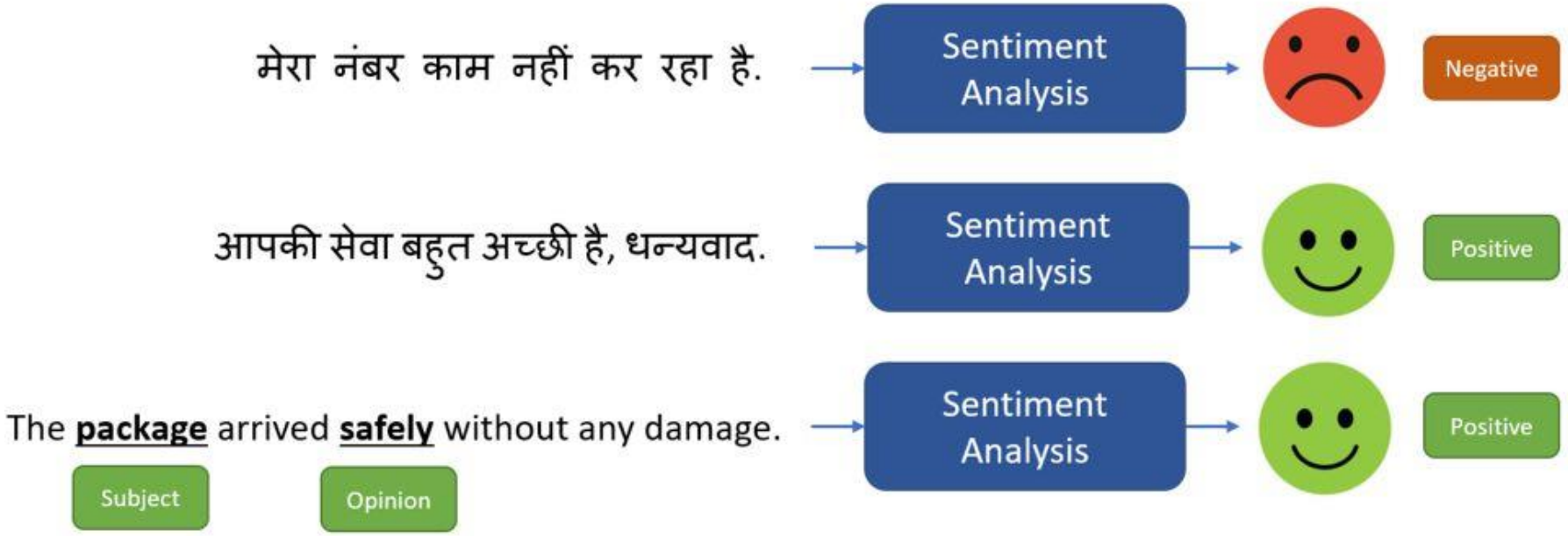
```
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
# Accuracy score
from sklearn.metrics import accuracy_score
print('accuracy is',accuracy_score(y_pred,y_test))
```

Text Analytics

Text analytics is the process of derivation of high-end information through established patterns and trends in a piece of text.

https://images.app.goo.gl/w6psDYmrqWcKehYt9

Text Analytics



https://images.app.goo.gl/w6psDYmrqWcKehYt9

Bag of Words Approach

The simplest form of text representation in numbers.

Example:

Review 1: This movie is very scary and long

Review 2: This movie is not scary and is slow

Review 3: This movie is spooky and good

	1 This	2 movie	3 is	4 very	5 scary	6 and	7 long	8 not	9 slow	10 spooky	11 good	Length of the review(in words)
Review 1	1	1	1	1	1	1	1	0	0	0	0	7
Review 2	1	1	2	0	0	1	1	0	1	0	0	8
Review 3	1	1	1	0	0	0	1	0	0	1	1	6

Vector of Review 1: [1 1 1 1 1 1 1 0 0 0 0]
Vector of Review 2: [1 1 2 0 0 1 1 0 1 0 0]
Vector of Review 3: [1 1 1 0 0 0 1 0 0 1 1]

https://cdn.analyticsvidhya.com/wp-content/uploads/2020/02/BoWBag-of-Words-model-2.png

Term Frequency-Inverse Document Frequency (TF-IDF)

- **Term Frequency (TF)**

  It is a measure of how frequently a term t appears in a document d.

$$tf_{t,d} = \frac{n_{t,d}}{Number\ of\ terms\ in\ the\ document}$$

Here, in the numerator, n is the number of times the term "t" appears in the document "d". Thus, each document and term would have its own TF value.

Example:

Review 1: This movie is very scary and long

Review 2: This movie is not scary and is slow

Review 3: This movie is spooky and good

Term	Review 1	Review 2	Review 3	TF (Review 1)	TF (Review 2)	TF (Review 3)
This	1	1	1	1/7	1/8	1/6
movie	1	1	1	1/7	1/8	1/6
is	1	2	1	1/7	1/4	1/6
very	1	0	0	1/7	0	0
scary	1	1	0	1/7	1/8	0
and	1	1	1	1/7	1/8	1/6
long	1	0	0	1/7	0	0
not	0	1	0	0	1/8	0
slow	0	1	0	0	1/8	0
spooky	0	0	1	0	0	1/6
good	0	0	1	0	0	1/6

https://www.analyticsvidhya.com/blog/2020/02/quick-introduction-bag-of-words-bow-tf-idf/

Inverse Document Frequency (IDF)

$$idf_t = \log \frac{number\ of\ documents}{number\ of\ documents\ with\ term\ 't'}$$

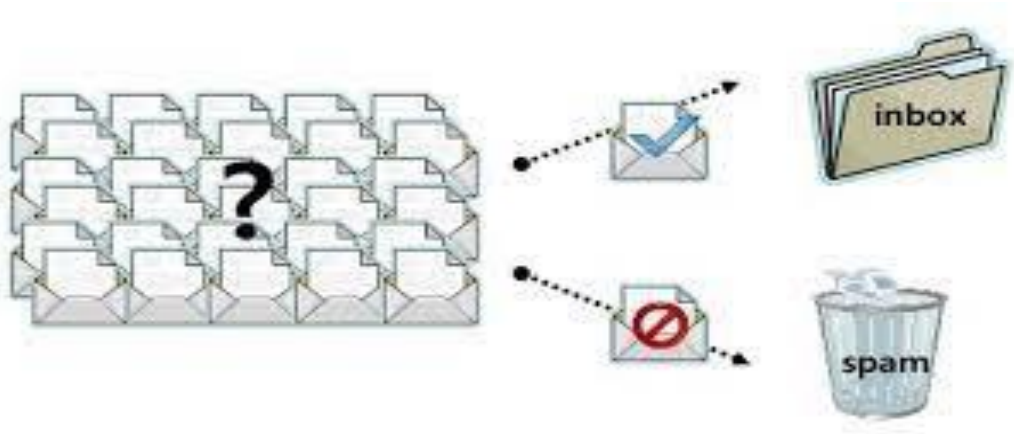
Example:

Review 1: This movie is very scary and long
Review 2: This movie is not scary and is slow
Review 3: This movie is spooky and good

IDF('this') = log(number of documents/number of documents containing the word 'this') = log(3/3) = log(1) = 0

Spam Ham Demonstration

- Spam Messages, Spam emails or messages belong to the broad category of unsolicited messages received by a user.
- Bag-of-words which is a natural language processing (NLP) algorithm can be used to classify messages as ham or spam.



https://miro.medium.com/max/362/1*4yyDiH-r0_q-aQ1w8nmGcw.jpeg

Lexicons for Sentiment Analysis

- Sentiment analysis is a process by which information is analyzed through the use of natural language processing (NLP).
- Lexicons calculate the sentiment from the semantic orientation of word or phrases that occur in a text.
- **Types:** Affin,Textblob, VADER etc.

Affin

- Lexicons used for sentiment analysis.
- Developed by Finn Årup Nielsen.
- It contains 3300+ words with a polarity score associated with each word.
- Initialize afinn sentiment analyzer.

```
from afinn import Afinn
af = Afinn()
```

Textblob

A simple python library that offers API access to different NLP tasks such as sentiment analysis, spelling correction etc.

```
from textblob import TextBlob

testimonial = TextBlob("The food was great!")
print(testimonial.sentiment)
```

```
Sentiment(polarity=1.0, subjectivity=0.75)
```

Polarity: float, lies between [-1,1]

-1 → negative sentiment

+1 → positive sentiments.

Subjectivity: float, lies in the range of [0,1].

generally refer to personal opinion, emotion, or judgment.

VADER

- Valence aware dictionary for sentiment reasoning
- Returns the probability of a given input sentence to be Positive, Negative, and Neutral.

```
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer


analyzer = SentimentIntensityAnalyzer()
sentence = "The food was great!"
vs = analyzer.polarity_scores(sentence)
print("{:-<65} {}".format(sentence, str(vs)))
```

```
{'compound': 0.6588, 'neg': 0.0, 'neu': 0.406, 'pos': 0.594}
```

REFERENCES

1. https://blog.floydhub.com/naive-bayes-for-machine-learning/
2. https://towardsdatascience.com/basic-probability-theory-and-statistics-3105ab637213
3. https://www.javatpoint.com/machine-learning-naive-bayes-classifier
4. https://www.analyticsvidhya.com/blog/2020/02/quick-introduction-bag-of-words-bow-tf-idf/
5. https://neptune.ai/blog/sentiment-analysis-python-textblob-vs-vader-vs-flair

THANK YOU