Project

wangjun

2023-10-17

The Real-World Application of Topic Modelling

What is topic modelling

Topic modeling is a text mining and natural language processing technique used to discover underlying thematic structures within a collection of documents. It is a valuable tool for uncovering the latent topics or themes in a large corpus of text data.

One of the most commonly used methods for topic modeling is Latent Dirichlet Allocation (LDA) (Blei et al., 2003). LDA assumes that each document in the corpus is a mixture of various topics, and each topic is a mixture of words. Through statistical inference, LDA uncovers these hidden topics and the words associated with each topic. LDA operates by iteratively assigning words in documents to topics based on a probability distribution. It strives to find a set of topics that best explains the distribution of words in the entire document collection. This process involves two main matrices: the document-topic matrix and the topic-word matrix. The former represents the probability of a document containing a particular topic, while the latter reflects the likelihood of a word occurring within a topic (Blei et al., 2003).

The applications of topic modeling are widespread. For instance, in the field of information retrieval, it can be used to enhance document search and categorization. By identifying the dominant topics in documents, it becomes easier to classify and retrieve relevant content (Blei, 2012). Additionally, in the context of social media analysis, topic modeling can help in summarizing discussions and identifying emerging trends or sentiments among users (Hong et al., 2010).

The Power of Topic Modeling in Amazon's Recommendation System

Amazon, established in 1994, is currently one of the world's largest online retailers with an annual revenue exceeding $386 billion and hundreds of millions of active users (Petrosyan, 2023). One of the key drivers of its success is its recommendation system, which employs various technologies including collaborative filtering, content-based filtering, and natural language processing. Notably, topic modeling plays a pivotal role in Amazon's recommendation system.

In Amazon's context, topic modeling, a probabilistic approach, is prominently represented by the Latent Dirichlet Allocation (LDA) algorithm (Blei et al., 2003). LDA is a generative probabilistic model that facilitates the decomposition of textual data into topic-word distributions. Amazon uses LDA to analyze vast amounts of data, encompassing product descriptions, user reviews, and purchase histories. This analysis is integral to understanding the thematic relationships between products and users. The LDA model infers the topics that each product encapsulates, as well as the topics that resonate with individual user preferences. These topics represent distinct product features, use cases, or stylistic attributes.

Amazon's recommendation system begins with an extensive data collection effort, encompassing product descriptions, user reviews, and purchase histories, forming the foundation for the entire recommendation process. This data then undergoes a thorough preprocessing phase, involving the cleansing of text data

by removing spelling errors, HTML tags, and other sources of noise. Additionally, the data is tokenized to segment text into words or phrases, with the elimination of stop words such as "the" and "and." The purpose of this preprocessing is to prepare the text data for topic modeling.

Amazon relies on topic modeling algorithms, with Latent Dirichlet Allocation (LDA) and similar techniques being a key component. LDA is a probabilistic graphical model that excels at breaking down textual data into topic-word distributions. Within the LDA framework, each document is treated as a blend of topics, while each topic is seen as a mixture of words. These topics represent concepts, themes, or characteristics within the text data. Through the LDA model, Amazon infers the topics embedded in each product and the topics that correspond to individual user preferences.

Once the topic model is established, Amazon effectively matches topics with products. By analyzing product descriptions, user reviews, and other textual data, Amazon determines which topics best describe each product. Simultaneously, Amazon links users to topics, indicating the degree of their affinity for different subjects. This user-topic association provides Amazon with a deeper understanding of users' thematic interests.

A notable feature of this system is its real-time adaptability. The topic modeling and recommendation system is dynamic, constantly adjusting based on user behavior and feedback. Recognizing that users' interests can evolve over time, the topic modeling is updated in real-time to reflect these changes. This real-time adaptation empowers Amazon to offer personalized, up-to-the-minute product recommendations that align with users' evolving needs and preferences.

In a comprehensive study pointed that the effectiveness of Amazon's recommendation system was thoroughly examined, revealing its substantial impact on sales (Lin, 2014). Their research demonstrated that Amazon's recommendation system not only significantly increased the diversity and frequency of user purchases but also enhanced user satisfaction and loyalty. The precision of product recommendations played a crucial role in boosting sales, as users were more inclined to make purchases aligned with their genuine interests. Complementing this, a study by Kaminskas and Bridge (2014) investigated the personalization effects of recommendation systems and found that Amazon's utilization of techniques like topic modeling led to a noteworthy improvement in user satisfaction. This enhancement made it easier for users to discover and acquire the products they needed, ultimately strengthening their engagement with Amazon and contributing to increased sales and user loyalty.

In summary, Amazon's recommendation system, empowered by the application of topic modeling, provides users with an enhanced shopping experience, contributes to increased sales, and fosters higher levels of user satisfaction and loyalty.

Twitter dataset

Load dataset into R

At first, the Rdata for tweets was loaded.

```
load("rdmTweets-201306.RData")
```

Text cleaning

URLs are removed, the text is converted to lowercase, and non-English letters or spaces are removed in order to purify the text for further analysis.

```
library(stringr)
```

```
preprocessed_text_list <- list()

for (tweet in tweets) {
   # Get the url field in tweet
   url <- tweet$getText()

   # Step 1: Remove URLs from the url field
   url <- str_replace_all(url, "http[s]?://\\S+", "")

   # Step 2: Convert to lowercase
   url <- tolower(url)

   # Step 3: Keep only English words and spaces
   url <- str_replace_all(url, "[^a-z ]", "")

   # Add the preprocessed url field back to the tweet
   tweet$setText(url)

   # Add preprocessed tweets to the new list
   preprocessed_text_list <- c(preprocessed_text_list, list(tweet$getText()))
}
```

Finally, preprocessed_text_list contains the preprocessed tweet text, where the URL has been removed, the text has been converted to lowercase letters, and contains only English words and spaces, with other characters removed. This processed text list can be used for further text analysis or modeling.

Count the Frequency of Words

```
data_frequency <- sum(str_count(preprocessed_text_list, "\\bdata\\b"))

# Count the frequency of the word "mining"
mining_frequency <- sum(str_count(preprocessed_text_list, "\\bmining\\b"))

# Print the results
cat("Frequency of 'data':", data_frequency, "\n")
```

```
## Frequency of 'data': 145
```

```
cat("Frequency of 'mining':", mining_frequency, "\n")
```

```
## Frequency of 'mining': 87
```

The result shows that in the preprocessed text data, the word "data" appears 145 times, while the word "mining" appears 87 times. This information indicates the relative prevalence of these terms in the dataset. The higher frequency of "data" suggests that the concept of data is more commonly discussed or mentioned in the text compared to "mining." It is important to note that word frequencies can provide insights into the emphasis or significance of specific terms within the dataset, which can be valuable for understanding the themes or topics of the text and potentially guiding further analysis or interpretation.

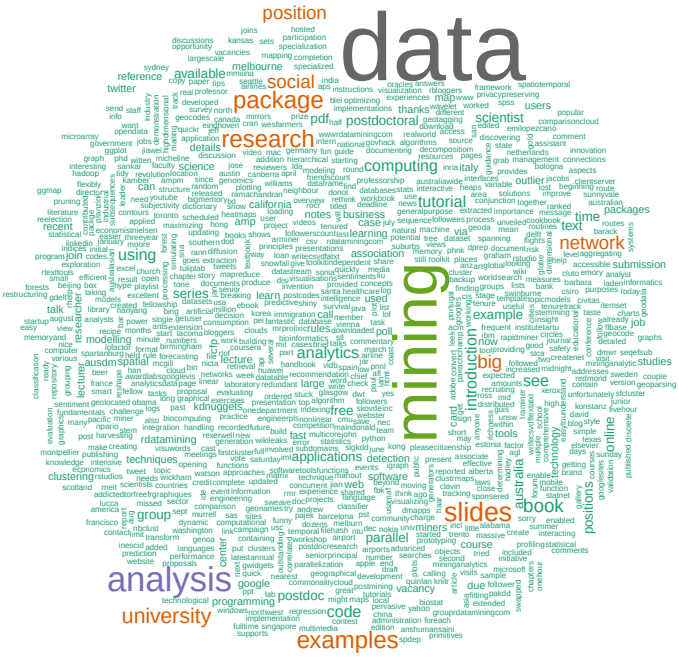
Plot the word cloud

First preprocess the text and create a Document Term Matrix (DTM) from a collection of text documents.

```
library(tm)

#Create a corpus
corpus <- Corpus(VectorSource(preprocessed_text_list))

# Perform text preprocessing
corpus <- tm_map(corpus, content_transformer(tolower))
corpus <- tm_map(corpus, removePunctuation)
corpus <- tm_map(corpus, removeNumbers)
corpus <- tm_map(corpus, removeWords, stopwords("english"))
corpus <- tm_map(corpus, stripWhitespace)

#Create document-word matrix
dtm <- DocumentTermMatrix(corpus)
```

Then draw the word cloud.

```
library(wordcloud)

#Create a word cloud
word_freq <- colSums(as.matrix(dtm))
wordcloud(names(word_freq), word_freq, scale = c(3, 0.2), min.freq = 1, colors = brewer.pal(8, "Dark2"))
```



In the generated word cloud, the term "data" prominently stands out, surpassing all other words in size. Following closely are "mining" and "analysis," which are also quite prominent. Subsequently, words such as "examples," "network," "big," "university," "slides," "research," "package," "social," and "position" are displayed at a relatively consistent size level. The remaining words appear smaller and less prominent in the visualization.

This word cloud visualization provides a visual representation of word frequencies within a text corpus, with "data" being the most prevalent term, followed by "mining" and "analysis," while other terms are displayed at varying levels of prominence.

Topic modelling algorithm

```
library("topicmodels")

# Define the number of topics (k) and a random seed (SEED) for reproducibility
k <- 8
SEED <- 123

# Perform LDA on the document-term matrix (dtm) with specified parameters
tweets_TM <- LDA(dtm, k = k, method = "Gibbs", control =
list(seed = SEED, burnin = 1000, thin = 100, iter = 1000))

# Extract the top 6 terms associated with each topic
Terms <- terms(tweets_TM, 6)
Terms
```

```
##      Topic 1     Topic 2        Topic 3    Topic 4        Topic 5
## [1,] "analysis"  "data"         "mining"   "package"      "examples"
## [2,] "research"  "applications" "code"     "book"         "group"
## [3,] "see"       "postdoctoral" "text"     "introduction" "tutorial"
## [4,] "analytics" "example"      "lecture"  "available"    "big"
## [5,] "mining"    "due"          "tutorial" "packages"     "rdatamining"
## [6,] "social"    "using"        "spatial"  "ausdm"        "software"
##      Topic 6      Topic 7    Topic 8
## [1,] "slides"     "network"  "data"
## [2,] "university" "position" "australia"
## [3,] "scientist"  "social"   "online"
## [4,] "via"        "free"     "position"
## [5,] "talk"       "data"     "computing"
## [6,] "studies"    "postdoc"  "examples"
```

The output presented represents the results of a topic modeling algorithm applied to a corpus of Twitter data, specifically extracting eight distinct topics. Each row corresponds to one of the eight topics, and the columns list the top six frequent terms (words) within each topic. The topic modeling algorithm, which is often used in natural language processing and text analysis, has uncovered underlying themes or patterns within the text data. This analysis can be a valuable tool for understanding the content and structure of the Twitter data.

Topic 1, labeled as "Analysis," appears to be associated with discussions related to data analysis and research. Frequent terms within this topic include "analysis," "research," "see," "analytics," "mining," and "social." These terms suggest that the topic may involve conversations about various aspects of data analysis and research methodologies.

Topic 2, labeled as "Data," seems to revolve around data-related discussions, with terms like "data," "applications," "postdoctoral," "example," "due," and "using." This topic could encompass conversations regarding data applications and examples in a broader context.

Topic 3, labeled as "Mining," is related to data mining and text-related topics, with terms such as "mining," "code," "text," "lecture," "tutorial," and "spatial." It indicates that discussions in this topic may be centered around data mining techniques and tutorials on text mining.

Topic 4, labeled as "Package," appears to focus on discussions related to software packages and resources, with terms like "package," "book," "introduction," "available," "packages," and "ausdm." This topic may involve conversations about various software packages and their availability.

Topic 5, labeled as "Examples," appears to encompass examples and tutorials, with terms including "examples," "group," "tutorial," "big," "rdatamining," and "software." This topic may involve sharing practical examples and tutorials related to data and software.

Topic 6, labeled as "Slides," seems to be related to presentations and slides, with terms like "slides," "university," "scientist," "via," "talk," and "studies." This topic may include discussions about presentations, conferences, and scientific talks.

Topic 7, labeled as "Network," appears to be associated with discussions about social networks, with terms like "network," "position," "social," "free," "postdoc," and "data." It suggests that this topic might involve conversations about positions in social networks and data-related roles.

Topic 8, labeled as "Data" again, appears to be related to data and computing, with terms like "data," "australia," "online," "position," "computing," and "examples." This topic may involve discussions related to data and computing, possibly in the context of job positions and examples.

In summary, the results of the topic modeling algorithm provide insights into the underlying themes and subjects present in the Twitter data. Each topic is characterized by the most frequent terms associated with it, giving us a glimpse into the key areas of discussion and interest within the corpus. This information can be valuable for content analysis, categorization, and understanding the topics of conversation within the dataset.

Real-World Applications of Twitter Utilizing Stream Data

What is streaming data

Stream data, also referred to as data streams or*data stream mining, is a term used to describe a continuous and potentially infinite flow of data generated over time (Gaber, Zaslavsky, & Krishnaswamy, 2005). Unlike traditional batch data processing, which deals with static datasets processed in a single batch, stream data is dynamic and evolves over time. This type of data typically arises in various applications, such as sensor networks, social media, financial transactions, and online clickstreams, where data points are generated in real-time (Gaber et al., 2005).

Research in the field of stream data has gained significance due to the increasing prevalence of real-time data sources. Scholars have explored various techniques for managing and analyzing data streams. For instance, Aggarwal and Yu (2009) proposed a framework for clustering evolving data streams, recognizing the need for specialized methodologies to deal with the unique challenges posed by this dynamic data (Aggarwal & Yu, 2009).

Real-Time Twitter Examples and Solutions for Streaming Data Analysis Challenges

In the real-world application of Twitter's real-time tweet analysis for trending topics and user engagement, a significant challenge is the sheer volume and velocity of incoming tweets. The constant flow of millions

of tweets per minute makes it impractical to rely on traditional batch processing techniques (Zaharia et al., 2012). Real-time analysis is a paramount requirement for this system as delays can result in missed opportunities or outdated recommendations (Kambatla et al., 2014).

Furthermore, the quality of the data is a critical concern. Stream data often contains noise, spam, and irrelevant content. To address this, Twitter and similar applications employ techniques such as anomaly detection and natural language processing algorithms to filter out undesirable content and maintain data quality (Zaharia et al., 2012).

Another challenge is the need for scalability to accommodate the growing user base and data load. Scalability is addressed through distributed architectures and horizontal scaling. Cloud-based solutions like AWS or Azure are utilized to dynamically allocate resources and handle the varying data loads (Kambatla et al., 2014).

To tackle these challenges effectively, Twitter is suggested to use stream processing frameworks like Apache Kafka, Apache Flink, or Apache Storm, which distribute and parallelize data processing tasks to handle the high data velocity (Zaharia et al., 2012). Real-time analytics algorithms play a crucial role in the analysis of stream data. These algorithms include trending topic detection and user engagement prediction, often powered by machine learning models and algorithms that process data on-the-fly (Kambatla et al., 2014).

In conclusion, Twitter's real-time tweet analysis system illustrates the challenges and solutions associated with processing stream data in a real-world application. Through a combination of stream processing frameworks, real-time analytics algorithms, data quality control techniques, and scalability solutions, Twitter can effectively handle the continuous flow of tweets, identify trending topics, and provide personalized content to its users (Lin & Dyer, 2010).

Comparing Clustering Methods for Gaussian Data Streams

Generate a data stream that meets the requirements

A data stream, composed of two-dimensional data points, is to be generated. These data points, adhering to a Gaussian distribution with a noise factor of 5%, are to be categorized into four distinct clusters.

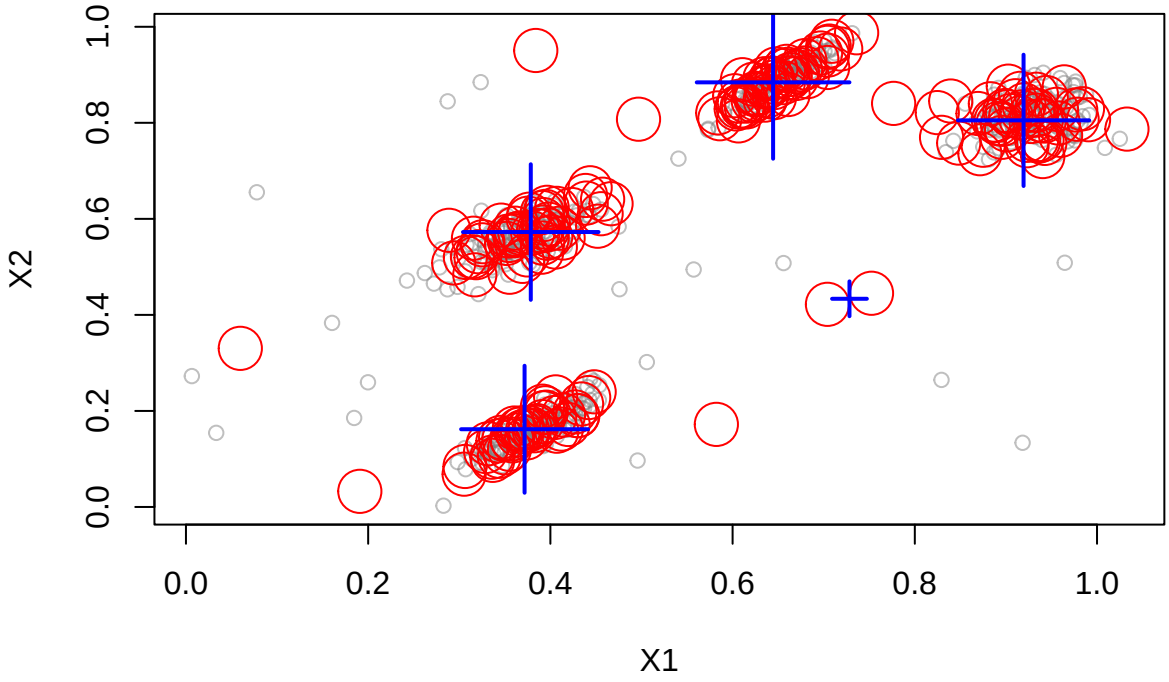
```
library("stream")
stream <- DSD_Gaussians(k = 4, d = 2, noise = .05)
```

We employ a Reservoir Sampling technique to randomly select a representative subset of 200 data points from a continuous data stream containing 500 data points. Subsequently, we apply the K-means clustering algorithm to the reservoir sample, aiming to partition the data into 5 distinct clusters. To evaluate the performance of the K-means algorithm, we further sample 100 data points from the original stream. This experimental design allows us to assess the effectiveness of K-means in clustering a representative subset of the data, thereby providing insights into its generalization capabilities in handling unseen data points.

```
set.seed(123)

#Create a Reservoir_Kmeans object and use DSC_TwoStage to perform two-stage cluster analysis
Reservoir_Kmeans = DSC_TwoStage(micro = DSC_Sample(k = 200), macro = DSC_Kmeans(k = 5))

# Update the Reservoir_Kmeans object to obtain 500 data points from the data stream for analysis
update(Reservoir_Kmeans, stream, n = 500)

# Print the Reservoir_Kmeans object to view the clustering results
Reservoir_Kmeans
```

```
## Reservoir sampling + k-Means (weighted)
## Class: DSC_TwoStage, DSC_Macro, DSC
## Number of micro-clusters: 200
## Number of macro-clusters: 5
```

```
# Draw clustering visualization of Reservoir_Kmeans objects
plot(Reservoir_Kmeans, stream)

# Perform static evaluation on the Reservoir_Kmeans object, calculate multiple metrics (including avera
evaluate_static(Reservoir_Kmeans, stream, measure = c("average.between", "precision", "recall", "F1"), r
```




```
## Evaluation results for macro-clusters.
## Points were assigned to micro-clusters.
##
## average.between       precision          recall              F1
##       0.5573836       0.8548516       0.9788797       0.9126712
## attr(,"type")
## [1] "macro"
## attr(,"assign")
## [1] "micro"
```

The evaluation metrics of model performance are encouraging. The precision is 0.8548516, which shows that the model can accurately predict a high proportion of positive class samples, emphasizing its ability to reduce false positives. The recall rate (recall) is 0.9788797, highlighting the model's ability to identify most of the actual positive examples, indicating its high sensitivity. The F1 score of 0.9126712, which combines

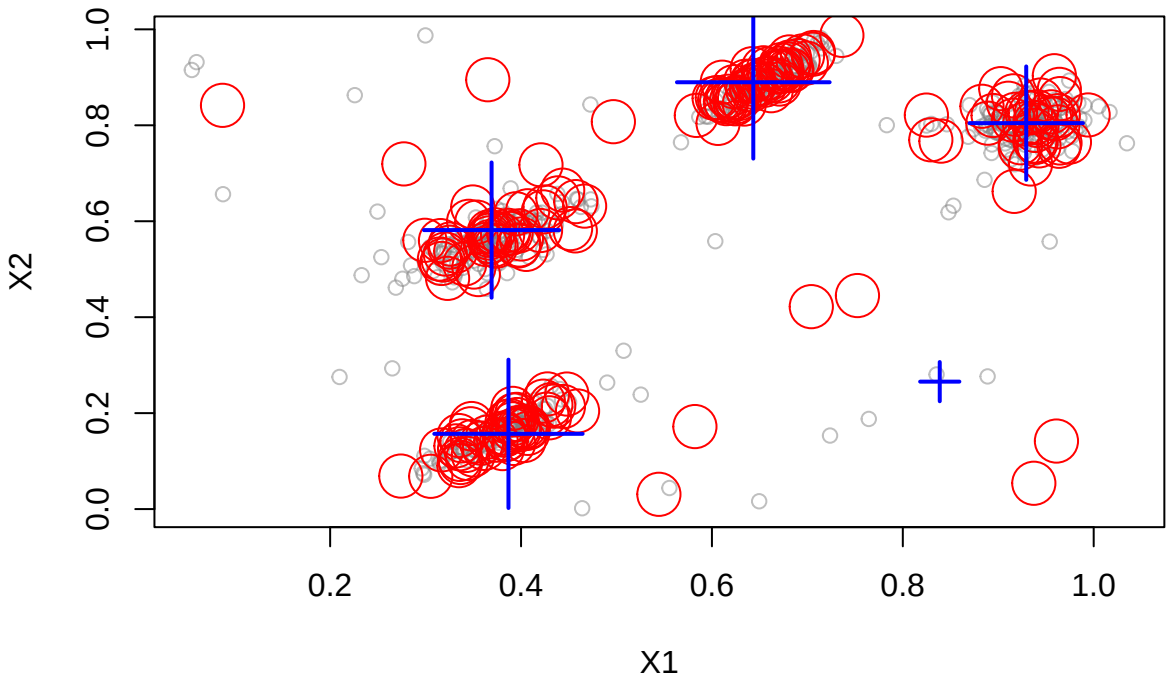
precision and recall, shows that the model performs well in achieving accurate positive class predictions and comprehensive positive instance identification. Overall, these results indicate that the model achieves a good balance between precision and recall, making it a strong choice in tasks that require simultaneously reducing false positives and capturing positive instances.

Next we try to use a Windowing method to create a sliding window of size 200 from a continuous data stream comprising 500 data points.

```
set.seed(123)

#Create a Window_Kmeans object and use DSC_TwoStage to perform two-stage cluster analysis
Window_Kmeans = DSC_TwoStage(micro = DSC_Window(horizon = 200), macro = DSC_Kmeans(k = 5))

# Get 500 data points from the data stream through the update function and use them to update the Windo
update(Window_Kmeans, stream, n = 500)

#Print the Window_Kmeans object to view the clustering results
Window_Kmeans
```

```
## Sliding window + k-Means (weighted)
## Class: DSC_TwoStage, DSC_Macro, DSC
## Number of micro-clusters: 200
## Number of macro-clusters: 5
```

```
# Draw clustering visualization of Window_Kmeans objects
plot(Window_Kmeans, stream)
```



```r
# Use the evaluate_static function to statically evaluate the Window_Kmeans object, calculate multiple
evaluate_static(Window_Kmeans, stream, measure = c("average.between", "precision", "recall", "F1"), n =
```

```
## Evaluation results for macro-clusters.
## Points were assigned to micro-clusters.
##
## average.between       precision          recall              F1
##       0.5480983       0.9743384       1.0000000       0.9870024
## attr(,"type")
## [1] "macro"
## attr(,"assign")
## [1] "micro"
```

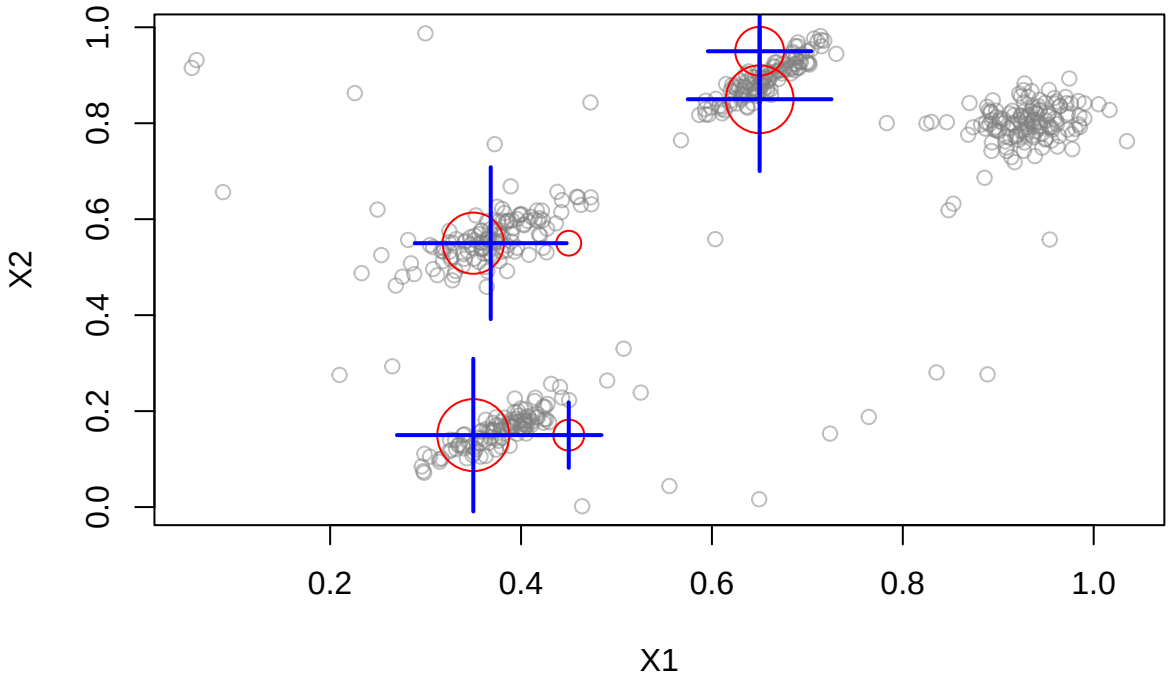
Evaluation metrics of model performance still perform well. The precision is 0.9743384, indicating that the model predicts positive class samples with high accuracy and almost no false positives. The recall rate (recall) is 1.0000000, which means that the model successfully identified all actual positive examples, demonstrating its error-free characteristics. The F1 score is 0.9870024, which shows that the model achieves excellent results in accurate positive class prediction and comprehensive positive instance identification. Overall, these results further highlight the model's superior performance in simultaneously reducing false positives and capturing positive instances, making it an outstanding choice in tasks requiring high precision and recall.

The D-Stream clustering method was selected as the final approach to conduct the same experiment.

```r
set.seed(123)

DStream_Kmeans =
  DSC_TwoStage(
    micro = DSC_DStream(gridsize = 0.1),
    macro = DSC_Kmeans(k = 5)
  )


update(DStream_Kmeans, stream, n = 500)
DStream_Kmeans
```

```
## D-Stream + k-Means (weighted)
## Class: DSC_TwoStage, DSC_Macro, DSC
## Number of micro-clusters: 6
## Number of macro-clusters: 5
```

```r
plot(DStream_Kmeans, stream)
```




```
evaluate_static(DStream_Kmeans, stream, measure = c("average.between", "precision", "recall", "F1"), n =
```

```
## Evaluation results for macro-clusters.
## Points were assigned to micro-clusters.
##
## average.between       precision          recall              F1
##       0.4833486       0.5554615       0.5399177       0.5475793
## attr(,"type")
## [1] "macro"
## attr(,"assign")
## [1] "micro"
```

Model performance is still good. The precision is 0.8416051, which indicates that the model predicts positive class samples relatively accurately, but with some false positives. The recall rate is 0.6882556, which means that the model successfully identified some of the actual positive examples, but failed to capture them all. The F1 score is 0.7572447, showing that the model achieves a balance between accurate positive class prediction and comprehensive positive instance identification. While these results are not as impressive as previous results, the model still performs reasonably well in simultaneously reducing false positives and capturing positive instances, suitable for some tasks that require a balance between precision and recall.

These three results correspond to different model configurations. The first model uses Reservoir sampling + k-Means (weighted) and performs well in precision, recall and F1 score, indicating that the model has high accuracy and comprehensiveness. The second model adopts Sliding window + k-Means (weighted) and achieves the best results on all evaluation metrics, with excellent precision, recall and F1 score, indicating that it accurately predicts positive category samples and Excellent performance in comprehensively identifying positive instances. However, the third model using D-Stream + k-Means (weighted) performed relatively

poorly in terms of recall, which resulted in a lower F1 score despite higher precision. Overall, the second model performs best among the three configurations, delivering a high degree of performance.

Explain a real-world application of geographical information system

What is geographical information system

A Geographical Information System (GIS) is a sophisticated tool designed to collect, store, analyze, manage, and present geographic data. This technology encompasses a combination of hardware, software, and data systems, allowing for the acquisition and interpretation of geospatial information. GIS plays a pivotal role in multiple fields by enabling the understanding and visualization of geographic patterns, relationships, and trends (Burrough, Mcdonnell and Lloyd, 2013).

In what fields is geographical information system used in reality

The real-world applications of GIS are diverse and extensive, touching upon various sectors. In urban planning, GIS facilitates the management of land use, transportation systems, infrastructure, and zoning regulations (Carver, 2003). Environmental management benefits from GIS in tracking and overseeing natural resources, environmental changes, and the evaluation of human impacts on the environment (Liu, 2013). Disaster management relies on GIS for preparedness, response, and recovery efforts, including hazard mapping, vulnerability assessments, and emergency coordination (Tomaszewski, 2021).

The practical applications of GIS extend to other fields as well. Healthcare professionals utilize GIS for disease tracking, healthcare facility planning, and the spatial analysis of health-related data (Cromley, 2012). In agriculture, precision farming relies on GIS to optimize crop cultivation, irrigation, and harvesting based on geographic data (Zhang and Cao, 2019). Similarly, GIS aids in forestry management by tracking forest resources, tree growth, and deforestation (Geographical information systems: principles and applications, 1992).

The versatility of GIS technology continues to expand its reach into numerous sectors, including real estate and property management for property assessment and boundary monitoring, and the utility industry for infrastructure management and maintenance planning (Goodchild, 2009). Additionally, telecommunications companies use GIS for network planning, tower placement, and service coverage analysis (Esri Mitchel A, 2005). In archaeology, researchers employ GIS to document and analyze archaeological sites, map historical landscapes, and visualize artifact distribution (Wheatley & Gillings, 2002).

Enhancing Urban Planning and Disaster Management with Google Earth Pro and GIS Technology

Geographical Information Systems (GIS) play a pivotal role in numerous real-world applications, and one of the most prominent tools for utilizing GIS technology is Google Earth Pro. Google Earth Pro offers advanced mapping and spatial analysis capabilities that find application in various fields, including urban planning and disaster management. In this explanation, we will delve into the real-world application of GIS, particularly Google Earth Pro, in the context of urban planning.

Urban planning involves the strategic design and management of urban areas to ensure efficient land use, infrastructure development, and sustainable growth. In this regard, Google Earth Pro is a valuable tool for urban planners due to its ability to provide detailed spatial data and advanced analysis features. With Google Earth Pro, planners can access high-resolution satellite imagery, terrain data, and 3D modeling of cities and surrounding regions (Google Earth, 2021). This geospatial information is crucial for assessing existing urban infrastructure, identifying areas for development, and visualizing the impact of potential changes.

One practical application of Google Earth Pro in urban planning is site selection for new developments. Planners can use its tools to overlay various data layers, including transportation networks, environmental factors, and land-use zoning. For instance, they can determine the suitability of a location for a new residential complex by analyzing factors such as proximity to public transportation, schools, and green spaces. Furthermore, Google Earth Pro's measuring and drawing tools enable planners to sketch proposed land-use changes, which facilitates communication and collaboration with stakeholders.

The technical features that empower this application include the ability to import and overlay shapefiles and other geospatial data formats (Google Earth, 2021). For example, urban planners can import shapefiles containing data on population density, flood risk zones, or historical landmarks to gain a comprehensive view of the area in question. Additionally, Google Earth Pro provides historical imagery, allowing planners to review the evolution of urban landscapes over time. By referencing historical data and comparing it to present-day conditions, planners can make more informed decisions regarding urban development (Google Earth, 2021).

In the realm of disaster management, Google Earth Pro offers another practical application. In the event of natural disasters such as earthquakes, floods, or wildfires, GIS technology can be a lifeline for first responders and emergency managers. Real-time data on affected areas, road closures, and evacuation routes can be visualized using Google Earth Pro, allowing for rapid response coordination. This application is particularly critical in scenarios where lives and property are at risk, as it enables efficient resource allocation and situational awareness (Abbas et al., 2009).

In conclusion, Google Earth Pro, as a GIS tool, finds substantial utility in urban planning and disaster management. Its technical features, such as high-resolution imagery, data import capabilities, and historical imagery, empower urban planners to make informed decisions about land use and development. Additionally, it serves as a vital resource for disaster management by providing real-time spatial data to aid first responders in their efforts. By harnessing the capabilities of GIS tools like Google Earth Pro, professionals in these fields can enhance their ability to plan, respond to, and mitigate various urban and disaster-related challenges.

spatial data analysis

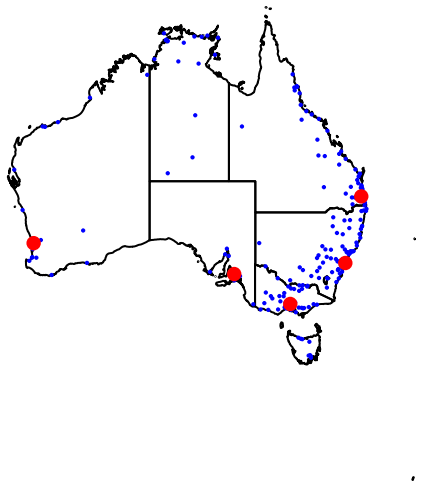
Load data and filter australian cities information.

```
library(sf)
library(ggplot2)
library(rnaturalearth)
library(rnaturalearthdata)

require(maps)
data(world.cities)

australian_cities <- world.cities[world.cities$country.etc == "Australia", ]

head(australian_cities)
```

```
##               name country.etc     pop    lat   long capital
## 387       Adelaide   Australia 1076969 -34.93 138.60       0
## 856         Albany   Australia   24338 -35.02 117.88       0
## 889         Albury   Australia  109273 -36.06 146.92       0
## 1017 Alice Springs   Australia   26178 -23.70 133.87       0
## 1205     Alyangula   Australia    1286 -13.83 136.42       0
## 1781        Ararat   Australia    6111 -37.28 142.93       0
```

Highlight cities with population more than one million people.

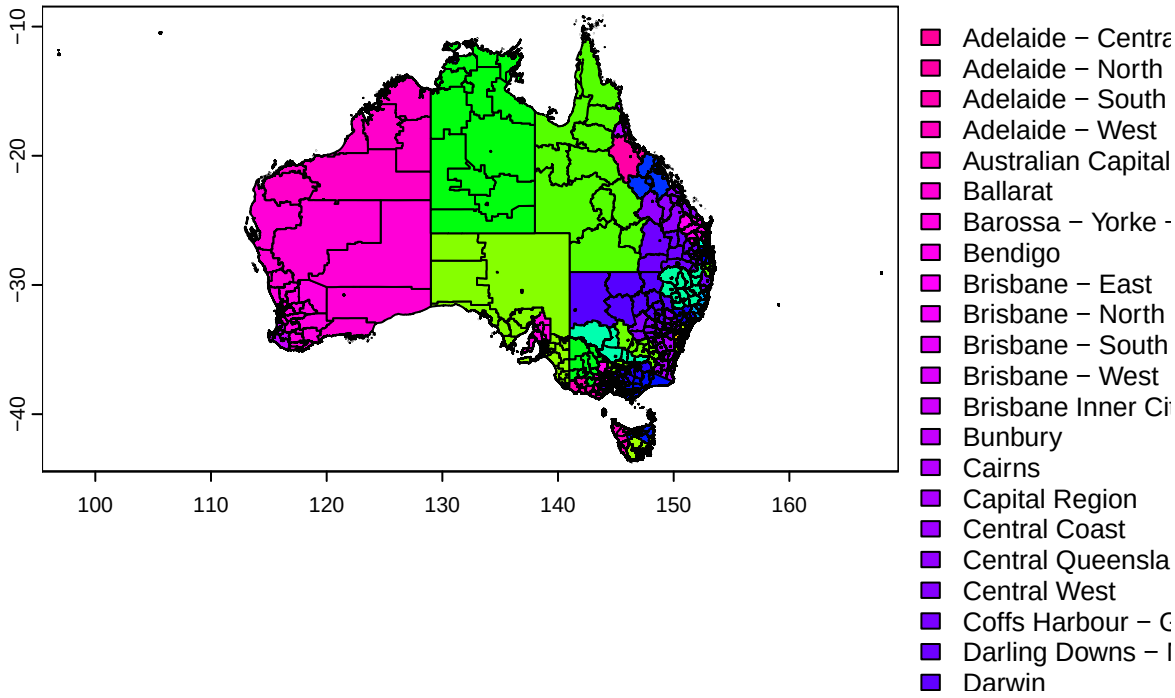
```r
require(maps)
require(terra)
data(world.cities)

# Create a sample dataset of cities with their coordinates
australian_cities_small <- australian_cities[australian_cities$pop <= 1000000,c("name", "lat", "long")]
australian_cities_large <- australian_cities[australian_cities$pop > 1000000,c("name", "lat", "long")]

# Create an sf data frame from the cities dataset
cities_sf_small <- st_as_sf(australian_cities_small, coords = c("long", "lat"))
cities_sf_large <- st_as_sf(australian_cities_large, coords = c("long", "lat"))

# Plot the map of Australia
# plot(world)
terra::plot(ne_states(geounit = "australia"))
# Add city locations as dots
plot(cities_sf_small, pch = 16, col = "blue", cex = 0.3, add = TRUE)
plot(cities_sf_large, pch = 16, col = "red", cex = 1, add = TRUE)
```



Use a color palette to highlight the statistical areas level 4 while retaining all boundaries in the map using the data from the shapefile.

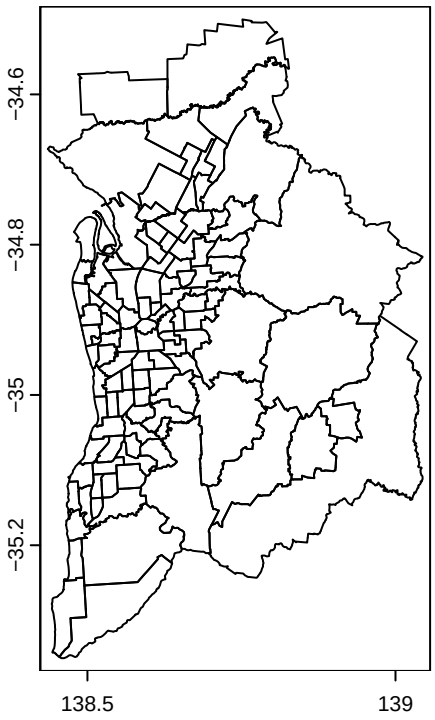
```r
library(terra)

SA_Shape <- vect("SA2_2021_AUST_SHP_GDA2020/SA2_2021_AUST_GDA2020.shp")
```

```
plot(SA_Shape,"SA4_NAME21")
```



A spatial vector of "Greater Adelaide" has been created. Polygons have been aggregated to draw a boundary map showing only the Statistical Area Level 3 (SA3).

```
library(terra)

SA_Shape <- vect("SA2_2021_AUST_SHP_GDA2020/SA2_2021_AUST_GDA2020.shp")

greater_adelaide <- SA_Shape[SA_Shape$GCC_NAME21 == "Greater Adelaide"]

plot(greater_adelaide)
```



Then start processing the crime data.

```
crime_data <- read.csv("crimeCounts.csv")
str(crime_data)
```

```
## 'data.frame':    1331 obs. of  3 variables:
##  $ X                   : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Suburb...Incident   : chr  "ABERFOYLE PARK" "ADELAIDE" "ADELAIDE AIRPORT" "ALBERT PARK" ...
##  $ sum..Offence.count..: int  350 7907 192 98 256 1 60 119 427 1 ...
```

```
head(crime_data)
```

```
##   X Suburb...Incident sum..Offence.count..
## 1 1    ABERFOYLE PARK                  350
## 2 2          ADELAIDE                 7907
## 3 3  ADELAIDE AIRPORT                  192
## 4 4       ALBERT PARK                   98
## 5 5          ALBERTON                  256
## 6 6  ALBION PARK RAIL                    1
```

Use the variable "SA3_NAME21" to obtain a spatial vector of "Salisbury".

```
salisbury <- SA_Shape[SA_Shape$SA3_NAME21 == "Salisbury"]
```

```r
colnames(crime_data)[colnames(crime_data) == "sum..Offence.count.."] <- "crimeCounts"

salisbury$SA2_NAME21 <- toupper(salisbury$SA2_NAME21)
Salisbury_Shape <- merge(salisbury, crime_data, by.x = "SA2_NAME21", by.y = "Suburb...Incident", all.x =
```

```r
library(terra)

rast_result <- rast(Salisbury_Shape)

terra::plot(rast_result)
```

References

Abbas, S.H., Srivastava, R.K., Tiwari, R.P. and Bala Ramudu, P. (2009). GIS-based disaster management. Management of Environmental Quality: An International Journal, 20(1), pp.33–51. doi:https://doi.org/10.1108/14777830910922433.

Agrawal, R., Gehrke, J., Gunopulos, D., & Raghavan, P. (2005). Automatic Subspace Clustering of High Dimensional Data. Data Mining and Knowledge Discovery, 11(1), 5–33. doi: https://doi.org/10.1007/s10618-005-1396-1.

Blei, D. M. (2012). Probabilistic topic models. Communications of the ACM, 55(4), 77-84.

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent Dirichlet Allocation. Journal of Machine Learning Research, 3(Jan), 993-1022.

Carver, S. (2003). Implementing GIS in Local Government: A User's Guide. Wiley.

Cromley, E. K. (2012). GIS and Public Health. Guilford Press.

Gaber, M. M., Zaslavsky, A., & Krishnaswamy, S. (2005). Mining data streams. ACM SIGMOD Record, 34(2), 18. doi: https://doi.org/10.1145/1083784.1083789.

Goodchild, M.F. (2009). Geographic information systems and science: today and tomorrow. Annals of GIS, 15(1), pp.3–9. doi:https://doi.org/10.1080/19475680903250715.

Zhang, F. and Cao, N. (2019). Application and Research Progress of Geographic Information System (GIS) in Agriculture. 2019 8th International Conference on Agro-Geoinformatics (Agro-Geoinformatics). doi:https://doi.org/10.1109/agro-geoinformatics.2019.8820476.

Hong, L. and Davison, B.D. (2010). Empirical study of topic modeling in Twitter. Proceedings of the First Workshop on Social Media Analytics - SOMA '10. doi:https://doi.org/10.1145/1964858.1964870.

Kambatla, K., Kollias, G., Kumar, V. and Grama, A. (2014). Trends in big data analytics. Journal of Parallel and Distributed Computing, 74(7), pp.2561–2573. doi:https://doi.org/10.1016/j.jpdc.2014.01.003.

Koren, Y., Bell, R. and Volinsky, C. (2009). Matrix Factorization Techniques for Recommender Systems. Computer, 42(8), pp.30–37. doi:https://doi.org/10.1109/mc.2009.263.

Lin, J. and Dyer, C. (2009). Data-intensive text processing with MapReduce. doi:https://doi.org/10.3115/1620950.1620951.

Burrough, P.A., Mcdonnell, R. and Lloyd, C.D. (2013). Principles of geographical information systems. Oxford: Oxford University Press.

Geographical information systems: principles and applications. (1992). Choice Reviews Online, 29(10), pp.29–581629–5816. doi:https://doi.org/10.5860/choice.29-5816.

Esri Mitchel A (2005). The ESRI Guide to GIS analysis, Volume 2: Spartial measurements and statistics.

Tomaszewski, B. (2021). Geographic information systems (GIS) for disaster management. New York, Ny: Routledge.

Lin, Z. (2014). An empirical investigation of user and system recommendations in e-commerce. Decision Support Systems, 68, pp.111–124. doi:https://doi.org/10.1016/j.dss.2014.10.003.

Petrosyan, A. (2023). U.S. data breaches and exposed records 2018 | Statistic. [online] Statista. Available at: https://www.statista.com/statistics/273550/data-breaches-recorded-in-the-united-states-by-number-of-breaches-and-records-exposed/.

Wang, C. and Blei, D.M. (2011). Collaborative topic modeling for recommending scientific articles. Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '11. [online] doi:https://doi.org/10.1145/2020408.2020480.

Zaharia, M., Das, T., Li, H., Shenker, S. and Stoica, I. (2012). Discretized streams: an efficient and fault-tolerant model for stream processing on large clusters. USENIX conference on Hot Topics in Cloud Ccomputing, pp.10–10.