**Clustering on Book Data**

**Group Name: Group Rohan Aryan**

**Group Members:**

|  |  |  |
| --- | --- | --- |
| **First name** | **Last Name** | **Student number** |
| **Aishlee** |  | **C0913045** |
| **Guruprasad** | **Tandlekar** | **C0902813** |
| **Jayachandhran** | **Saravanan** | **C0910392** |
| **Prashanta** | **Timsina** | **C0916254** |
| **Rohan** | **Aryan** | **C0912902** |
| **Yogita** | **Sharma** | **C0913011** |

**Submission date: *15-APRIL-2024***

Contents

[**Abstract** 3](#_Toc164085410)

[**Introduction** 3](#_Toc164085411)

[**Literature Review** 4](#_Toc164085412)

[**Methods** Data 5](#_Toc164085413)

[**Data validation** 6](#_Toc164085414)

[**Methods** 7](#_Toc164085415)

[EDA 7](#_Toc164085416)

[Inference 8](#_Toc164085417)

[Pandas profiling 9](#_Toc164085418)

[Text processing 10](#_Toc164085419)

[Visualization 11](#_Toc164085420)

[**Model** 13](#_Toc164085421)

[Multinomial Naive Bayes classifier with TF-IDF 14](#_Toc164085422)

[Logistic regression classifier with word2Vec 14](#_Toc164085423)

[hyperparameter tuning 15](#_Toc164085424)

[hyperparameter tuning\_2 15](#_Toc164085425)

[Experiment Model\_1 17](#_Toc164085426)

[Experiment Model\_2 17](#_Toc164085427)

[Approach 3: SVC 18](#_Toc164085428)

[Approach 4: pre-trained LLMs 18](#_Toc164085429)

[**Results and comparison** 19](#_Toc164085430)

[**Evaluation metrics (custom)** 20](#_Toc164085431)

[**Web Application** 20](#_Toc164085432)

[**Conclusion:** 22](#_Toc164085433)

[**Future Work:** 22](#_Toc164085434)

[Dataset and Onedrive links 22](#_Toc164085435)

[**Reference :** 22](#_Toc164085436)

# **Abstract**

This report covers the end-to-end steps followed to build a recommendation system on top of the book data using description (text) as the main feature. It also discusses the basic EDA and processing steps required for all kinds of data types. The word embedding techniques are evaluated with the scrapped data, and the performance of the model is tabulated. This project uses a hybrid-based modelling approach, consisting of both unsupervised and supervised models, to build a similarity function. The multinomial classifier is trailing behind the logistic regression with an accuracy of 65%, and the silhouette score was below the average of .6, suggesting more feature sets for both regression and clustering types of modelling. The USE and word2vec outperform TF-IDF with a new evaluation metric called precision@K by getting the relevance score. Finally, the application is built using Stream lit to showcase the performance of the model on the input data, where the user can get recommendations of books based on content only (with no influence of favor or rating )

**Key words: EDA, recommendation system, precision@K, USE, word2vec, embeddings, user, stream lit, TF-IDF and hybrid model**

# **Introduction**

In this project, the detailed analysis of the “Book” data fetched from the ebooks website is carried out. The following sections will discuss in detail each step involved and the experiments carried out. Recommendation systems are an integral part of various businesses and marketing by leveraging the data with potential combination and similarity formulation.

**Problem Statement:**

This project revolves around creating a dedicated recommendation system for bibliophiles or researchers with more focus on content-based recommendation rather than hybrid or collaborative systems. The current generic recommenders are not specific in nature and also bring indirect preference or favouritism-based ranking on the recommendations. There is a potential use case of applying content (text-only) based recommendation tools on the unused library and research paper data to aid the content ambiguity.

**Objective:**

* To build a machine-learning recommendation engine based on content and description data
* Understand how to utilize data with minimalistic features and the high dimension of records to populate inferences
* Apply the learning objective of the course with real-world data and address the challenges
* Exploring various ML-based approaches and metrics to improve the performance

**significance and scope:**

The ultimate scope of the project is limited to the content-based textual data, and the adjacent approaches, like collaborative or hybrid methods, are not suitable for the defined problem statement. The advancement in the field of recommendation systems is considered and few approaches were tried out in this activity.

# **Literature Review**

Complete gap analysis with a proper review of the available recommendation systems is done. The following topics and works are considered to benchmark our solution of building content-based filtering

[1] Filtering Recommendation System by Michael D. Ekstrand, John T. Riedl and Joseph A. Konstan gives a complete understanding of content-based (text) recommendation systems, which analyzes the significance of the NLP in retrieving the context. Further, their work provides effective handling of text data, starting from extraction to evaluation of the algorithms in real-world

[2] Evaluation metrics, numerous papers, and metric systems are studied, and the method that is most suitable for the given characteristics is considered. Ability of the recommendation system with no influence of other factors is closely monitored

[3] Similarity and hybrid approach: a detailed literature review on the usage of available similarity methods like Manhattan, Jaccard distance and Cosine (most popular) suitable for this problem is done, and the hybrid approach of utilizing the unsupervised and supervised techniques on building a similarity-based distance function is applied here after thorough review.

# **Methods** Data

The data is scrapped using the API method, and the following diagram represents the working principle of the scrapper. The justification of choosing the data is provided in the later part of this section.

A diagram of a software application

Description automatically generated

A screenshot of a computer program

Description automatically generatedA computer screen shot of a program

Description automatically generated

The scraper fetched around 43,000 records across 11 ebooks website sub-categories; the API cap's limitation was restricted to fetch more useful information. The book title, price, description, and other author-related information are notable fields in the scrapped data. The ethical considerations for scrapping and usage of the data are highly followed by adhering to the privacy policies.

# **Data validation**

The raw data is first validated against various parameters to check the quality of the data and authenticate it for further processing. The following table provides a clear description of the data.

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Value** | **Remarks** |
| Shape and size | 43055 rows  16 columns  688880 value cells | A good amount of data to proceed with |
| Column type | Category- 8  Binary(boolean) - 1  Numerical - 7 | Raw data does not have the data in the correct format, and the categorical column contains date-related columns |
| Missing values | 5 out of 16 columns contain missing values | Missing values of columns book\_price,author\_name(s),sub\_title\_book,book\_edition, and description are found with counts with percentages of 30% and 92% |
| Duplicate values | A total of 788 repetitive values are identified | Using first occurrence of data for processing |
| Data understanding | Significance of each fields are listed out by exploring its impact in the project | Non-significance parameters are ignored S |

# **Methods**

|  |  |
| --- | --- |
| **Process** | **Brief Description** |
| EDA | Data Analysis on the numerical and categorical columns |
| Profiling | Using pandas profiling to understand the data more clearly and closer |
| Handling outliers for numeric values | Extreme numeric values are optimized by floor and cap method based on the significance |
| Visualization | Insights, hidden patterns and correlations are observed |
| Text processing | NLP-based method to process the text data and feature engineering of description data is done |
| Vectorization and word embeddings | Using TF-IDF and word2Vec to get the vector embedding representation of the description data |
| Embedding chart | Visual representation of the word embeddings |
| pre-processing | 3 distinct pre-processing techniques are applied based on the model requirement |
| Unsupervised model(hybrid\_layer\_0) | Clustering validation of the genre is carried out |
| Supervised model (hybrid\_layer1) | A classification model based on the genre with a word embedding vector is applied |
| Supervised model (hybrid\_layer3) | A regression model based on the price and the feature present are trained |
| Neural Network+similarity function (hybrid\_layer\_2) | The distance-based neighbour points are trained on the baseline k-NN model |
| Evaluation and verification | Models and performance are tested based on the type and used precision@K and recall@K |
| testing | Sample user inputs are tested to recommend books |

The above table gives an overall view of the methodologies used in this project; following this section, discuss the important steps followed and the respective inference in detail and with clear notes. The learning takeaways are addressed in the next section, which comprises the post-model analysis and the simple User Interface to conceptualize the application idea.

## EDA

The numeric, categorical, and date columns are extensively analyzed, and the required processing is applied to smooth data for the models.

* The categorical columns are checked for the variance of data to imply its significance
* The numerical columns are analyzed for the outliers and distribution range
  + The missing values and outliers are handled in the pre-processing steps based on the requirements
* The date-based columns are curated by converting them from the categorical and numerical values.
  + Maximum and minimum values are observed
  + The unique value counts are stored
  + Interestingly, there are some human-data entry errors as one of the date columns shows future data  **04-Mar-2025**
* The total statistical analysis of the numerical values is plotted with defined percentile to get a more granular level of understanding of the distribution
* The text-based data, i.e., Title, sub\_title and descriptions, are fed to the basic NLP-based analysis like word, sentence counts and stopwords
* Vectorized text is also examined for the sparse, and the dimensions of the vectors are learned

The handling of the extreme or potential bottleneck situation is discussed in the following sections. The best-fit parameters and the pipeline for the project is formed based on this analysis.

A screenshot of a graph

Description automatically generated

## Inference

* the columns like common height, common\_width, and book\_id are not significant
* the book price --> one of the indirect target variables is showing positive skewness with a heavy tail
* the date fields are showing negative skewness of the date columns
* 27K+ authors are found
* more than 900 publishers
* the editions of the books are going till 2

## Pandas profiling

Data validation is referred to with the pandas profiling report created on the raw data to cross-verify our inference.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

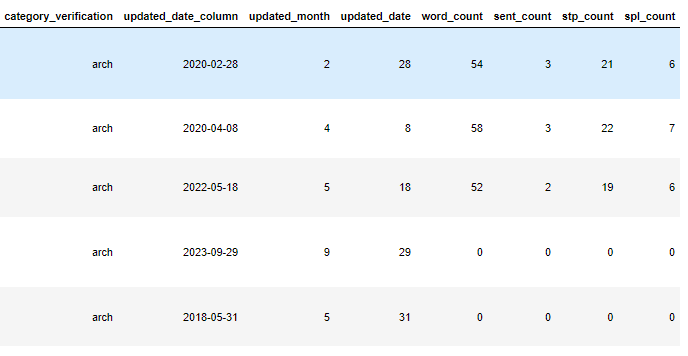
Description automatically generated

Justification

* The Pandas profiling validates our inference in most of the aspects
* The following steps are carried out based on the successful proof of inference
  + Outlier handling
  + Duplicate value removal
  + Feature selections
  + Data type conversion
  + Normalization of the data
  + Correlation analysis

## Text processing

The main component of this project is processing the text data, particularly the concatenation of the text data. The basic analytical features are added to the existing data frame to bring more context to the data and possibly utilize it in the training phase.



* Paraphrase the data in hand to generate custom sub-titles or descriptions, merging the features to have a final column for the corpus
* Word\_embedding like word2vec and vectorization of the text using TF-IDF is carried out, and we notice words like 's' etc., in the embeddings. This occurs because the tokenization might recognize 's' as a token itself. In cases like "it's", when we tokenize this, we can see the tokens formed are 'it' and 's'. Hence, we notice that here
  + for words that are in contracted format, we see the tokenization creating two separate tokens. Generally, we see contraction words as not having more than two words after the apostrophe symbol.

**Issue faced:** During the processing of embeddings for large sentences, the issue of NaN and the very big dimension arose, and it created a hurdle for proceeding further

**Solution tried:** compute the average of each recorded vector embedding to reduce the inaccuracy in dimensionality per title using ht pre-trained model

* Using a pair plot, the 150 dimension vectors are plotted and checked for their presence in the distribution graph
* Concatenating the word\_embedding vector with the other set was challenging during the training phase; the  Singular Value Decomposition and Principal Component Analysis were applied
  + The PCA was underperforming as the vector is non-linear
  + SVD helped in getting more meaningful data retrieved after reducing it to vector if dimension 10

A screen shot of a graph

Description automatically generated

A graph with text on it

Description automatically generated with medium confidence

## Visualization

Along with the EDA, a few visualizations are applied to get hidden insights and details related to it.

1. The distribution plot is used to understand the outliers present in the data before and after handling the data
2. The correlation heat map helps in feature selection in this project
3. Distribution of the categories is viewed through the pie chart
4. Word cloud is used to understand the frequent words used in the description
5. Authors use **conference, year, and business  words** in the book mostly

Learning: Usage of the 3D plot using Plotly to analyze the year price and author count in 3 D

A graph of a distribution of book prices

Description automatically generatedA blue and white squares with white text

Description automatically generated

A colorful pie chart with text

Description automatically generated

A close up of words

Description automatically generatedA graph with numbers and a number of numbers

Description automatically generated with medium confidence

A graph with different colored bars

Description automatically generated

A graph with blue dots

Description automatically generated

Outlier handling: Using the floor and cap method, the numeric outliers are handled

## **Model**

Approach 0: Supervised model – classifier

* The first layer model is to classify the query into a specified category of genre
* It saves high computation time, and the efficiency of the model is increased drastically
* The classification model used here will be trained on
  + Features: processed description vectors, concatenated text and other features (trial and error approach)
  + Target: Category column

***Common Pre\_processing\_consideration:***

* The label encoder is used to get the numeric representation of the classes present in the target variable
* **The pipeline** is constructed to pass the model and vectorizer to train the data and generate the model
* Test and train is split with a raito of 80:20, and the stratified method is applied

***Experiment Model\_1:***

### Multinomial Naive Bayes classifier with TF-IDF

* Using MB\_Naive Bayes to classify the discrete features here, we consider
  + Description text vector in 150-dimension
* A pipeline with both vectorizer and model is fitted
* This method is efficient in text based classifier (baseline model)
* Following are the results attached:
  + M\_NB (Default Parameters):
    - Training Accuracy: 66.26%
    - Testing Accuracy: 62.42%

***Experiment Model\_2:***

### Logistic regression classifier with word2Vec

* Using logistic regression, the classification is applied using the probability of features belonging to one particular class
* A pipeline with both embedding and model is fitted
  + Pandas series object is transformed into numpy array for easy processing
* This method is very simple and effective
* Handle multiclass variables smoothly
* Convergence is set to 1000 to ensure maximum values
* Following are the results attached:
  + log\_regression(Default Parameters):
    - Training Accuracy: 65.926%
    - Testing Accuracy: 64.142%

***Experiment Model\_3***

### hyperparameter tuning

*Multinomial Naive Bayes classifier with TF-IDF (hyperparameter tuning)*

* On top of the default baseline model, hyper-parameter tuning is conducted
* The alpha: Laplace parameter is tuned, it is used for smoothing the learning rate
  + Values from the range of (0.01) to 10 are considered (trial and error)
  + Cross-validation using a random search method is applied with a CV score of 5, and accuracy is chosen as the scoring method
  + The best value parameter is **0.2158**
* A pipeline with both vectorizer and model with parm\_grid is fitted
* Following are the results attached:
  + M\_NB (hyperparameter tuning ):
    - Training Accuracy: 73.51%
    - Testing Accuracy: 65.50%

A graph with red and green lines

Description automatically generated

**Experiment Model\_4**

### hyperparameter tuning\_2

*Logistic regression classifier with word2Vec(hyperparameter tuning)*

* On top of the default model, the following hyper parameters are added:
  + C: regularization to strength control
  + Penalty: L1 or L2 penalty methods
* A pipeline with both vectorizer and model with parm\_grid  is fitted
* Grid search is used and the best parameter is found as **'C': 0.1, 'max\_iter': 200, 'penalty': 'l2'**
* Following are the results attached:
  + log\_regression (hyperParameters):
    - Training Accuracy: 65.26%
    - Testing Accuracy: 64.42%

**A graph with red and green lines

Description automatically generated**

Approach 1: K-Nearest Neighbours

* With deep research into the topic of recommendation systems, there are multiple literature reviews of top-notch inference engines that perform according to market standards.
  + best recommendations based on ‘Cosine Similarity’ either directly or indirectly.
* In this attempt, an in-house query-able recommendation system is built, all possible test cases are formulated.
  + Eculidan and mahateen-based distance techniques are not valid as the data representation of processed word embedding
  + Cosine similarity is finalized as our distance measuring method as the vector representation accommodates the versatility of data

**Assumption** :

* KNearest Neighbours is used in both experiments

**Reason:**

* performs well in a scenario where when working with high-dimensional data
* In text data, the direction of the vectors is more important than their size. Cosine similarity is frequently utilized.

Prior analysis: TFIDF and BOW are inefficient in similarity checks; further results are attached in the later sections

### Experiment Model\_1

***Similarity model= wrod2vec (shallow neural network) – static***

* retains the semantic meaning of the word used and has the word converted to a meaningful vector of its corresponding word with the best possible dimensions
* User-based prompts are used to test the similarities. Later, the precision@K is used for testing the performance

A screen shot of a computer program

Description automatically generated

### Experiment Model\_2

***Similarity model= USE(*** *Universal Sentence Encoder by google****) – contextual based***

* Universal Sentence Encoder instead of word2vec for word-embedding
* aims to convert a word into a high-dimensional vector that can be further used for any Natural Language Processing-based task.
* Encoder greatly benefits our product build as this is the best in case of retaining the true semantic meaning of a sentence
* Test prompts were executed on this experimentation and the results were captured.

A screenshot of a computer program

Description automatically generated

## Approach 3: SVC

***Experiment Model\_1***

## Approach 4: pre-trained LLMs

***Experiment Model\_1***

* Implementing a pre-trained model can greatly enhance any natural language processing (NLP)
* A screenshot of a graph

  Description automatically generatedproject. Such models have already undergone extensive training on vast amounts of labelled
* and unlabeled data, including images and text. For our project, we opted to utilize the renowned
* pre-trained model, GPT-2, developed by OpenAI. GPT-2 has been pre-trained on a diverse
* corpus of text data, providing a solid foundation for various NLP tasks.
* Our specific objective was to fine-tune the GPT-2 model using our book dataset. This dataset
* comprises book titles and additional details such as author names, publication
* years, prices, and book descriptions. The subsequent sections outline the processes involved in

**A table with numbers and letters

Description automatically generated**

## **Results and comparison**

In this project, the overall analysis and processing are discussed in the above sections, the results and inference of the processes carried are below

|  |  |  |
| --- | --- | --- |
| **Model** | **Purpose** | **Remarks** |
| Multinominal Naive Bayes(default) | Layer\_1 - classification | Overfitting with a difference of 4% |
| Multinominal Naive Bayes(hyper\_parameter) | Layer\_1 - classification | Overfitting with a difference of 8% |
| Logistics regression(default) | Layer\_1 - classification | Slightly overfitting - but decent normalization |
| Logistic regression(hyperparamters) | Layer\_1 - classification | Perfect model |
| SVC() | Layer\_1 - classification |  |
| Logistics regression(default) | Layer\_3 -regression | A high value of MSE score → signifies no scope of using regression with the given parameters |

A graph of a performance

Description automatically generated

***Learning curve:***

This learning curve illustrates the performance of a machine learning model as the number of training examples increases. The red line represents the training score, showing how well the model fits the training data. The green line represents the testing score, indicating the model's performance on unseen data. From the plot, it's evident that as the number of training examples increases, the training score decreases slightly, suggesting that the model might be overfitting to the training data. However, the testing score improves initially with more data but starts to plateau, indicating that adding more training examples beyond a certain point might not significantly improve the model's performance on unseen data.

## **Evaluation metrics (custom)**

* For this project, hybrid models are used, but to test the overall performance of the application with desired goals, we need a more authenticated method to test it
* Here, the **precision@K** method is used to check the recommendation with the user prompt
  + Mathematical equation is
    - abs(relevant recommendation by system in top 10)/ K
  + It provided a transparent and accurate measure of relevance than any other known technique to the team
* Though ranking and complex testing are not possible, it satisfy the assumption and testing criteria
* The USE word embedding outperformed the word2vec by a score of 85% whereas word2vec gave 70%

## **Web Application**

* This system provides users with recommendations for books based on their input. It incorporates data loading, a recommendation engine, and user interface components.
* Data Loading:
  + The system loads data from pickle files (book.pkl and cosine\_sim.pkl) containing information about books and precomputed cosine similarity scores, respectively.
* Recommendation Engine:
  + The get\_recommendations function inputs a book title and returns a list of recommended books based on precomputed similarity scores.
* User Interface:
  + The Streamlit application interface is defined using various components such as text inputs, buttons, and data displays.
* Input Book Title:

Users can enter the title of a book into the text input field provided.

* Get Recommendations:
  + Upon clicking the "Get Recommendations" button, the system retrieves and displays a list of recommended books based on the input title.
* View Recommendations:
  + The recommended books are displayed below the input section, showing the book titles and author names.

**Tech Stack**

Streamlit, Pandas, Pickle, Base64

A screenshot of a computer screen

Description automatically generated

A screenshot of a book recommendation system

Description automatically generated

## **Conclusion:**

Thus, the recommendation based on content (text) is carried out with a hybrid model, and the results are discussed. Also, it is noted that the assumed hypothesis gives a decent performance on the pilot test, and the simple user interface is developed to test the same.

## **Future Work:**

* Exploring and experimenting with various advanced models and vectorization like matrix factorization techniques, deep learning models
* Scraping more useful data and making it scalable
* Leveraging the collaborative based recommendation models
  + Azure price planner
  + Spotify music recommendation

## Dataset and Onedrive links

* Dataset -
* pandas\_profiling-
* Whole working folder -
* Git hub - <https://github.com/svjai/AML_2203_project>

## **Reference :**

Ekstrand, M. D., Riedl, J. T., & Konstan, J. A. (2011). Collaborative filtering recommender systems. In P. Brusilovsky, A. Kobsa, & W. Nejdl (Eds.), The adaptive web: methods and strategies of web personalization (pp. 291–324). Springer.

Deutschman, Z. (2023, August 7). Recommender Systems: Machine learning metrics and business metrics. neptune.ai. <https://neptune.ai/blog/recommender-systems-metrics>

Rajani, N. F., Mooney, R. J., & Association for Computational Linguistics. (n.d.). Combining supervised and unsupervised ensembles for knowledge base population. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (pp. 1943–1948) [Conference-proceeding]. Association for Computational Linguistics. <https://www.cs.utexas.edu/users/ml/papers/rajani.emnlp16.pdf>