BIG DATA (BD) COURSEWORK REPORT

Do not write your name on your work unless your lecturer has explicitly told you to do so.

Student ID number	Title of degree studying	**Level/Year**
2200918	Bachelor of Science (Honours) Data Science and Analytics	1

Short unit name:	M32364 – Big Data				Due date: 29 May 2023		Deadline: 29 May
Full unit name:	Big Data						
Unit lecturer name:						Group: *(if applicable)*	
Additional items e.g. CD/disk/USB:	Yes		No	✓	**Details:**		

All additional items should be clearly labelled with ID number and unit name and securely attached to your work.

Candidates are reminded that the following are defined as Assessment Offences and will be dealt with in accordance with the University's Code of Student Discipline:

a) Any attempt to complete an assessment by means considered to be unfair;

b) Plagiarism, which the University defines as the incorporation by a student in work for assessment of material which is not their own, in the sense that all or a substantial part of the work has been copied without any adequate attempt at attribution or has been incorporated as if it were the student's own work when in fact it is wholly or substantially the work of another person or persons.

Please note: Group **coursework** will be filed under the first Student ID number at the top of the list. Ensure you know all **group member's ID numbers.**

NB: **Coursework not collected** will be disposed of six months **after** the hand-in date.

FOR OFFICIAL USE ONLY

Date received/Office stamp	Provisional mark % / Comments

Administration Office

Academic Staff Member

1.0 Introduction – TMDB 5000 Dataset

The TMDB 5000 dataset is a movie metadata dataset containing information about thousands of movies, including their titles, genres, release dates, ratings, and other attributes.

The problem statement for TMDB 5000 movie content-based filtering by movie tags is to create a movie recommendation system that recommends movies based on similar textual descriptions as the input movie and recommending those as potential movies that the user might be interested in.

The goal is to enhance the user experience by recommending movies that align with their interests and preferences, thereby increasing the chances of the user engaging with the recommended movies.

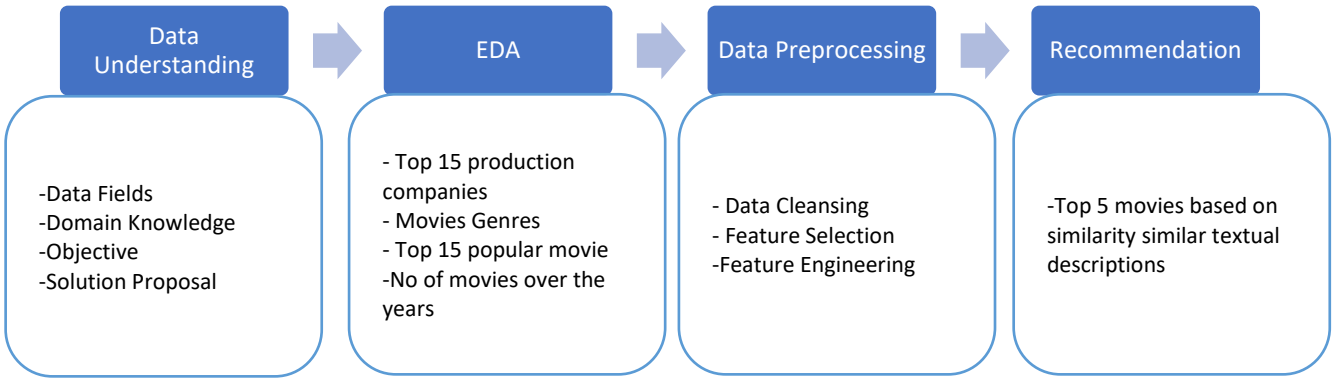
1.1 Details of Approach

- Content-based filtering

It is a recommendation technique that focuses on the characteristics of items to make recommendations. It analyzes the features and attributes of items, such as text, metadata, or user-generated content, to understand their properties and match them to user preferences.

1.2 Algorithms

The overall solution approach could be summarized as below flowchart:



1.3 Experimental Results and Analysis

1.3.1 Experimental Setup

1. Data Understanding

```
# First 5 movies
movies.head()
```

	budget	genres	homepage	id	keywords	original_language	original_title	overview	popularity	productio
0	237000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.avatarmovie.com/	19995	[{"id": 1463, "name": "culture clash"}, {"id":...	en	Avatar	In the 22nd century, a paraplegic Marine is di...	150.437577	[{"name" Parti
1	300000000	[{"id": 12, "name": "Adventure"}, {"id": 14, "...	http://disney.go.com/disneypictures/pirates/	285	[{"id": 270, "name": "ocean"}, {"id": 726, "na...	en	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be dead, ha...	139.082615	[{"nam Pictu
2	245000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.sonypictures.com/movies/spectre/	206647	[{"id": 470, "name": "spy"}, {"id": 818, "name...	en	Spectre	A cryptic message from Bond's past sends him o...	107.376788	[{"n Pictures",
3	250000000	[{"id": 28, "name": "Action"}, {"id": 80, "nam...	http://www.thedarkknightrises.com/	49026	[{"id": 849, "name": "dc comics"}, {"id": 853,...	en	The Dark Knight Rises	Following the death of District Attorney Harve...	112.312950	[{"na Pictures

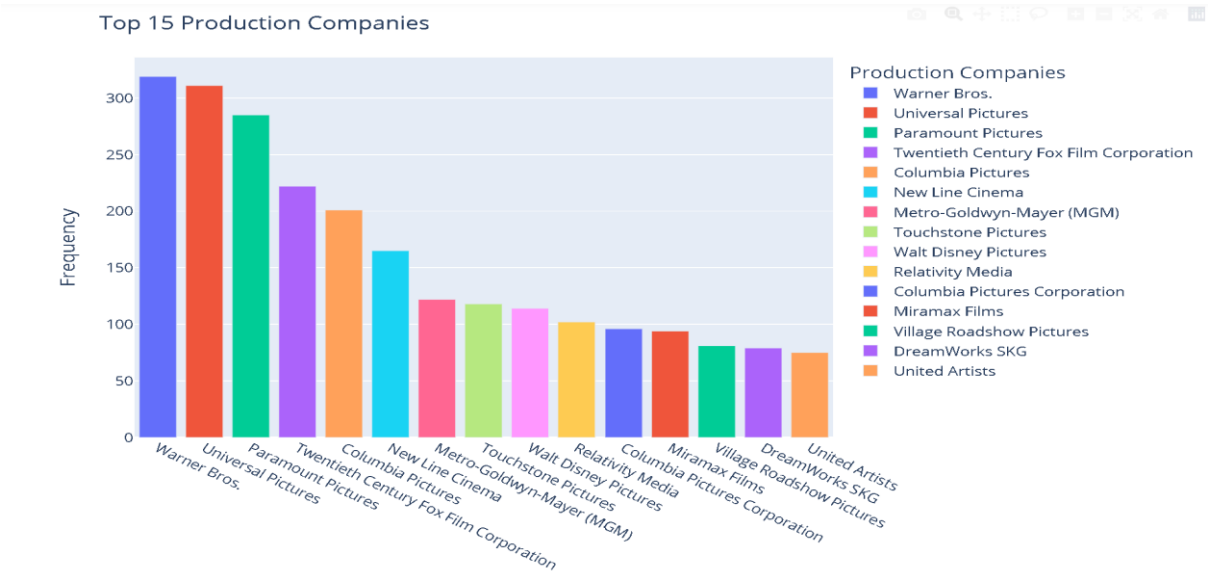
There are 4803 rows and 23 columns (data type: 16 strings, 4 int, 3 float). There are missing values from homepage, overview, release_date, runtime and tagline.

```
merge.info()
```

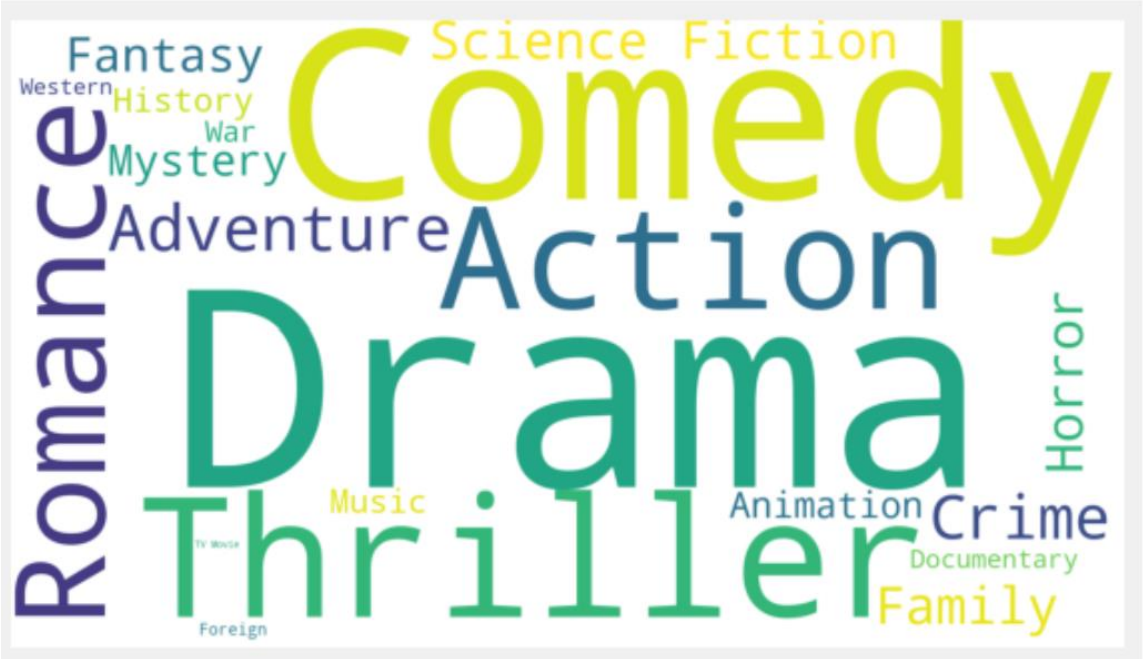
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4803 entries, 0 to 4802
Data columns (total 23 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   budget                4803 non-null   int64
 1   genres                4803 non-null   object
 2   homepage              1712 non-null   object
 3   id                    4803 non-null   int64
 4   keywords              4803 non-null   object
 5   original_language     4803 non-null   object
 6   original_title        4803 non-null   object
 7   overview              4800 non-null   object
 8   popularity            4803 non-null   float64
 9   production_companies  4803 non-null   object
 10  production_countries  4803 non-null   object
 11  release_date          4802 non-null   object
 12  revenue               4803 non-null   int64
 13  runtime               4801 non-null   float64
 14  spoken_languages      4803 non-null   object
 15  status                4803 non-null   object
 16  tagline               3959 non-null   object
 17  title_x               4803 non-null   object
 18  vote_average          4803 non-null   float64
 19  vote_count            4803 non-null   int64
 20  title_y               4803 non-null   object
 21  cast                  4803 non-null   object
 22  crew                  4803 non-null   object
dtypes: float64(3), int64(4), object(16)
```

Features	Missing values
homepage	**3,091**
overview	**3**
release_date	**1**
runtime	**2**
tagline	**124**

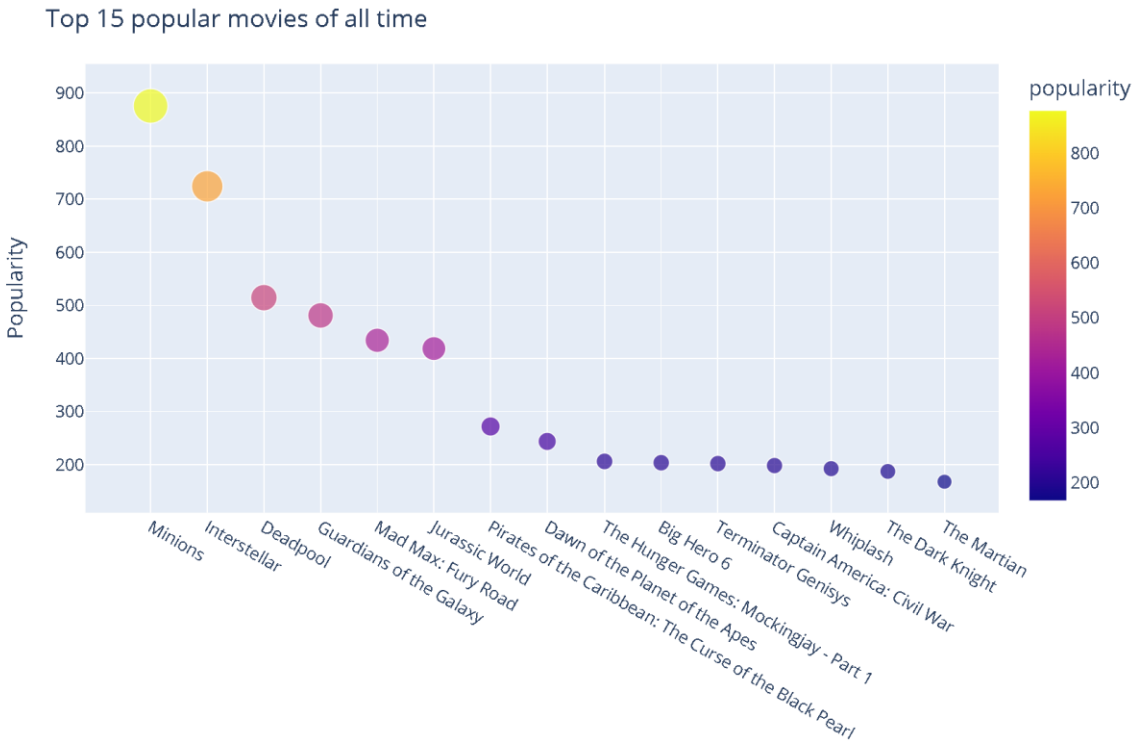
2. EDA



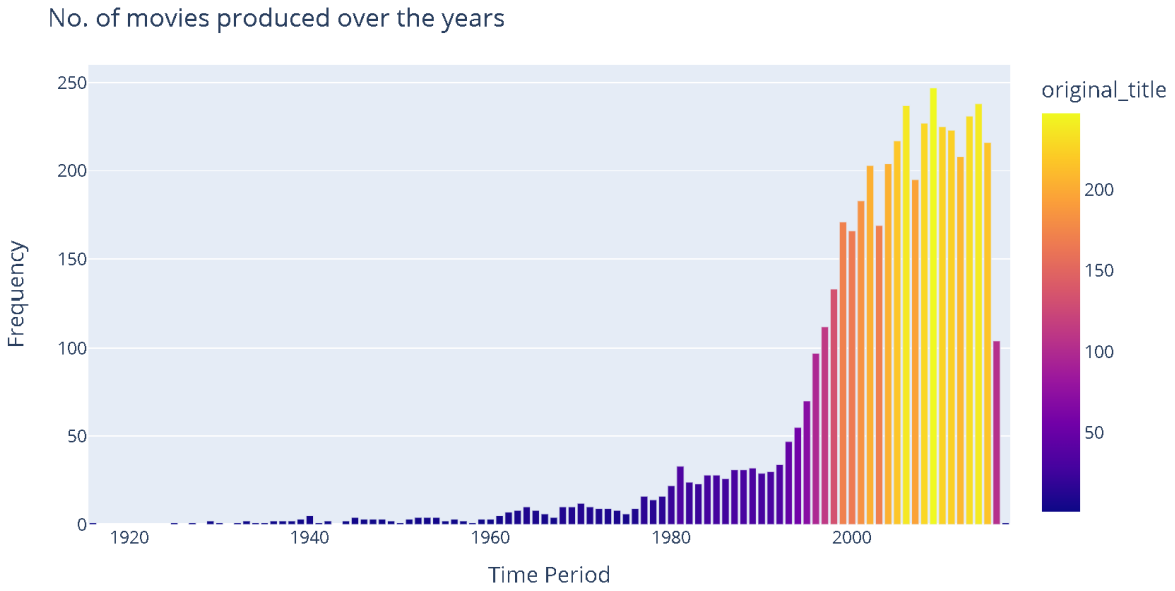
TMDB movies top 15 production companies bar graph in descending order. Warner bros, Universal Pictures and paramount pictures produce the most movies.



The TMDB movies dataset predominantly consists of movies belonging to genres such as Comedy, Drama, Thriller, and Romance.



TMDB movies top 15 most popular movies of all time.



Most of the movies were produced after the year 2000.

3. Data Pre-processing

- Data cleansing

➢ Check if any duplicated rows

```
# Check if any duplicated row
movies.loc[movies.duplicated(),:]
```

	id	original_title	overview	genres	cast	keywords	crew

➢ Feature selection

```
#Selecting only the useful features
movies = movies[['id','original_title','overview','genres','cast','keywords','crew']]
```

```
movies.head(3)
```

	id	original_title	overview	genres	cast	keywords	crew
0	19995	Avatar	In the 22nd century, a paraplegic Marine is di...	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	[{"cast_id": 242, "character": "Jake Sully", "...	[{"id": 1463, "name": "culture clash"}, {"id":...	[{"credit_id": "52fe48009251416c750aca23", "de...
1	285	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be dead, ha...	[{"id": 12, "name": "Adventure"}, {"id": 14, "...	[{"cast_id": 4, "character": "Captain Jack Spa...	[{"id": 270, "name": "ocean"}, {"id": 726, "na...	[{"credit_id": "52fe4232c3a36847f800b579", "de...
2	206647	Spectre	A cryptic message from Bond's past sends him o...	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	[{"cast_id": 1, "character": "James Bond", "cr...	[{"id": 470, "name": "spy"}, {"id": 818, "name...	[{"credit_id": "54805967c3a36829b5002c41", "de...

➢ Check if any missing values

```
# Checking null values
movies.isnull().sum()
```

```
id                0
original_title    0
overview          3
genres            0
cast              0
keywords          0
crew              0
dtype: int64
```

```
movies.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4800 entries, 0 to 4802
Data columns (total 8 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   id              4800 non-null   int64 
 1   original_title  4800 non-null   object
 2   overview        4800 non-null   object
 3   genres          4800 non-null   object
 4   cast            4800 non-null   object
 5   keywords        4800 non-null   object
 6   crew            4800 non-null   object
 7   tags            4800 non-null   object
dtypes: int64(1), object(7)
memory usage: 337.5+ KB
```

As shown above, there are 3 missing values in overview. The use of dropNA() method will drop rows that contain missing values.

- Replace square brackets with empty strings

```
# Replace square brackets with empty strings
movies2 = movies2.replace({'\\[|\\]': ''}, regex=True)
movies2.head()
```

- Spilt text in overview column

```
# Splitting the text in the overwiew column
movies['overview'] = movies['overview'].apply(lambda x:x.split())
```

- Apply a transformation to remove spaces between words for genres, keywords, cast and crew features

```
# Applying a transformation to remove spaces between words

movies['genres'] = movies['genres'].apply(lambda x:[i.replace(" ","") for i in x])
movies['keywords'] = movies['keywords'].apply(lambda x:[i.replace(" ","") for i in x])
movies['cast'] = movies['cast'].apply(lambda x:[i.replace(" ","") for i in x])
movies['crew'] = movies['crew'].apply(lambda x:[i.replace(" ","") for i in x])
```

- Tags feature derived from overview, genres, cast, keywords, and crew

```
# Making tags column by combining other 5 columns
movies['tags'] = movies['overview'] + movies['genres'] + movies['keywords'] + movies['cast'] + movies['crew']
```

- New data frame movies2 consist only id, original_title and tags for recommendation system

```
movies2.head()
```

	id	original_title	tags
0	19995	Avatar	in the 22nd century, a paraplegic marine is di...
1	285	Pirates of the Caribbean: At World's End	captain barbossa, long believed to be dead, ha...
2	206647	Spectre	a cryptic message from bond's past sends him o...
3	49026	The Dark Knight Rises	following the death of district attorney harve...
4	49529	John Carter	john carter is a war-weary, former military ca...

First 5 rows after data pre-processing

4. Feature Engineering

- New feature tags derived from overview, genres, keywords, cast and crew
- Text Vectorization: CountVectorizer

It converts a collection of text documents into a matrix of word counts.

- PorterStemmer

Stemming is a text processing technique that reduces words to their root or base form, which can help to improve text analysis by reducing the number of unique words in the dataset and grouping together words with similar meanings.

5. Recommendation System – Content Based Filtering

Function to get recommendations for a movie based on the tags

Step 1: Text Vectorization: The CountVectorizer technique is used to convert the 'tags' column of the movies2 dataframe into a feature matrix of token counts.

```
#Text Vectorization
from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer(max_features=5000,stop_words='english')

vectors = cv.fit_transform(movies2['tags']).toarray()
```

Step 2: The PorterStemmer technique is used to stem the words in the 'tags' column to improve the quality of the features.

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

#defining the stemming function
def stem(text):
    a=[]
    for i in text.split():
        a.append(ps.stem(i))
    return " ".join(a)

movies2['tags'] = movies2['tags'].apply(stem)
```

Step 3: Similarity Measurement: The cosine_similarity function from the sklearn.metrics.pairwise module is used to calculate the pairwise cosine similarity between the feature vectors of all the items in the dataset.

```
# Measuring similarity between Movies using cosine distance
from sklearn.metrics.pairwise import cosine_similarity
similarity = cosine_similarity(vectors)
```

```
similarity
```

```
array([[1.        , 0.08980265, 0.05986843, ..., 0.02457366, 0.02777778,
        0.        ],
       [0.08980265, 1.        , 0.06451613, ..., 0.02648136, 0.        ,
        0.        ],
       [0.05986843, 0.06451613, 1.        , ..., 0.02648136, 0.        ,
        0.        ],
       ...,
       [0.02457366, 0.02648136, 0.02648136, ..., 1.        , 0.07372098,
        0.04721922],
       [0.02777778, 0.        , 0.        , ..., 0.07372098, 1.        ,
        0.05337605],
       [0.        , 0.        , 0.        , ..., 0.04721922, 0.05337605,
        1.        ]])
```

Step 4: Recommendation: The recommend_me function takes a movie name as input, finds the index of the movie in the movies2 dataframe, calculates the cosine similarity between the feature vector of the movie and the feature vectors of all other movies in the dataset, and returns the top 5 most similar movies as recommendations

```
def recommend_me(movie):
    movie_index = movies2[movies2['original_title'] == movie].index[0]
    distances = similarity[movie_index]
    movies_list = sorted(list(enumerate(distances)),reverse=True, key=lambda x:x[1])[1:6] # Top 5 movies similar

    for i in movies_list:
        print(movies2.iloc[i[0]].original_title)
```

```
recommend_me('Pirates of the Caribbean: On Stranger Tides')
```

```
Pirates of the Caribbean: Dead Man's Chest
Pirates of the Caribbean: The Curse of the Black Pearl
The Pirates! In an Adventure with Scientists!
Pirates of the Caribbean: At World's End
20,000 Leagues Under the Sea
```

```
recommend_me('Batman')
```

```
Batman & Robin
The Dark Knight Rises
Batman Begins
Batman Returns
Batman v Superman: Dawn of Justice
```

The recommendation system will recommend top 5 movies based on similarity similar textual descriptions as the input movie and recommending those as potential movies that the user might be interested in.

3.0 Discussion and Conclusions

- Summary of project achievements

➢ TMDB 5000 Movie dataset

It does not rely heavily on user history or past behavior. It can provide recommendations even for new or first-time users based on the properties of items and their similarity to user preferences. It does not require a large user base or rely heavily on data from other users. It primarily focuses on the characteristics of items, making it useful in situations where user data is unavailable.

- Future Direction for improvement

We considered the four options below for future improvements. To begin, we can communicate more with business users to better understand their business concepts and, hopefully, improve model prediction performance by leveraging their experience. Second, we can use improved techniques to continue fine-tuning the parameters. Third, we can search for more efficient algorithms to run the datasets.

Lastly, we can try to gather additional data to augment existing dataset. More data can help increase the diversity and representation of our samples, improving the model's generalization capabilities. If the data test set is small, it may not adequately represent the entire population or the true distribution of the data. The results may be more prone to random variations and may not provide a reliable estimate of model performance.

4.0 Reference

1. Convolutional Neural Network with Implementation in Python (analyticsvidhya.com)
2. 10 steps to build and optimize a ML model - DEV Community
3. Getting Started with a Movie Recommendation System | Kaggle
4. Text Vectorization and Word Embedding | Guide to Master NLP (Part 5) (analyticsvidhya.com)
5. Cosine Similarity - Understanding the math and how it works? (with python) (machinelearningplus.com)