Configuration Manual

MSc Research Project
Data Analytics

Subash Ayyalusamy
Student ID: X22162933

School of Computing
National College of Ireland

Supervisor: Teerath Kumar Menghwar

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Subash Ayyalusamy
Student ID:	X22162933
Programme:	Data Analytics **Year:** 2023
Module:	MSc Research Project
Lecturer:	Teerath Kumar Menghwar
Submission Due Date:	14th December 2023
Project Title:	Analysing Viewer Engagement and Preferences in Anime Streaming Platforms.
Word Count:	982 **Page Count:** 6

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Subash Ayyalusamy
Date:	14/12/2023

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Analyzing Viewer Engagement and Preferences in Anime Streaming Platforms.

Configuration Manual

Subash Ayyalusamy
x22162933

Introduction

The configuration manual has detailed instructions on how to set up a system or device that is needed for a research project. Its main goal is to go into great depth about how to start the study, including what kind of machines are needed to build and run the models. The manual goes over the most important parts of both a basic setup that makes sure the project works and the programs and apps that are needed.

Project Files Detail

The Jupyter Notebook app is the main tool used in this study for cleaning up data, exploring it, modeling it, and evaluating it. For the length of the project, all datasets used in this study are kept as.csv files in the Jupyter Notebook environment.

Datasets Used

The datasets for this research are taken from Kaggle which is a public domain. The dataset is split for training, validating, and testing the model. It is divided into 80% training and 20% testing.

System Specification

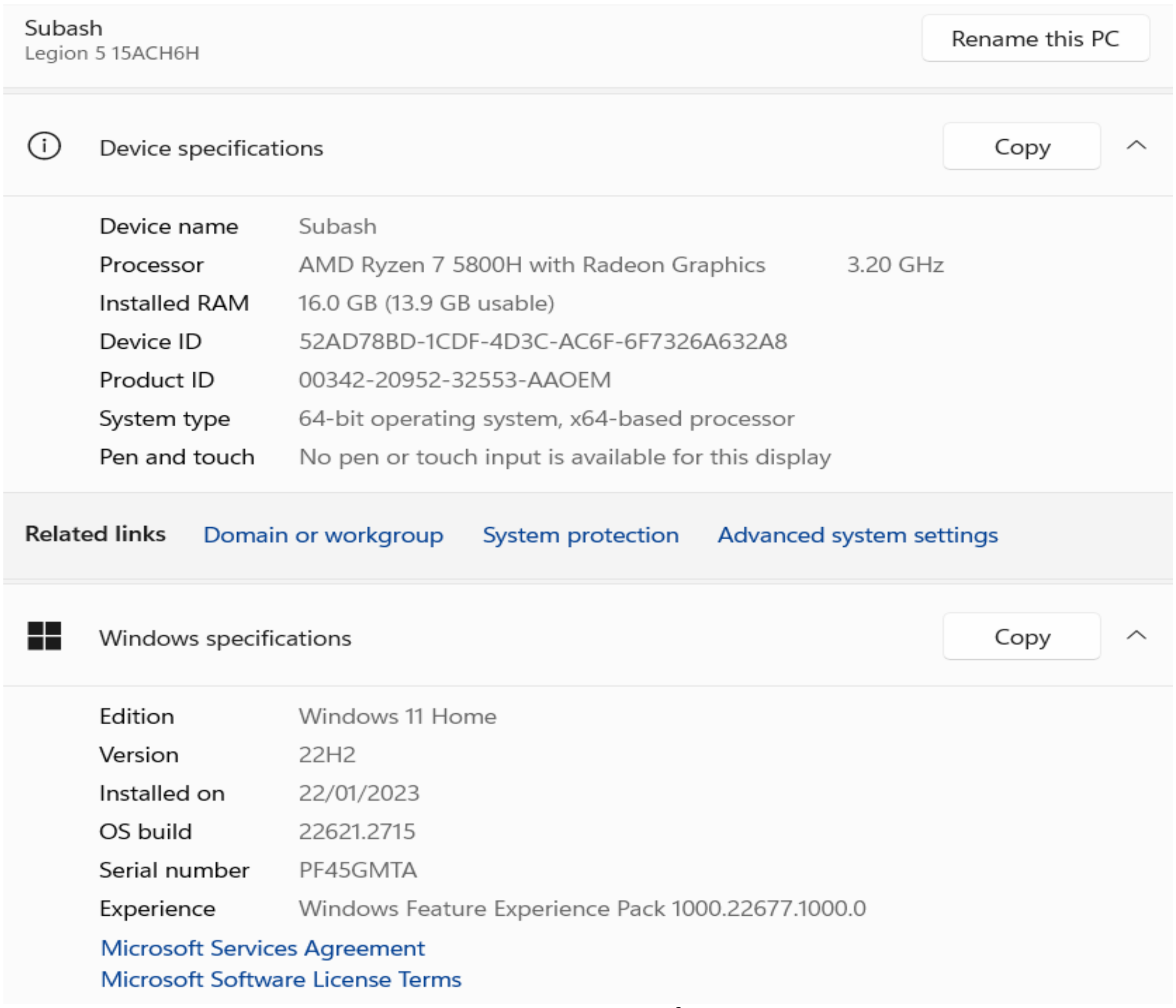
Figure 1 shows the system setup that was used for this project, and Table 1 shows the requirements for Jupyter Notebook.

A basic system requirement for effective operation is 8 GB of RAM. Though this amount is sufficient to achieve the expected results, it can result in longer processing times. Conversely, the recommended system configuration is tailored to ensure that the code runs smoothly, highlighting the importance of balancing system requirements with performance efficiency in technical projects.

Jupyter Notebook	
Processor	Core i3-530
The GPU Instance	4 GB
The Ram	12 GB
The Disk Space	128 GB

Max lifetime of VM	12 hrs

Table 1: Jupyter Notebook Specification



Subash
Legion 5 15ACH6H

Device specifications

Device name	Subash
Processor	AMD Ryzen 7 5800H with Radeon Graphics 3.20 GHz
Installed RAM	16.0 GB (13.9 GB usable)
Device ID	52AD78BD-1CDF-4D3C-AC6F-6F7326A632A8
Product ID	00342-20952-32553-AAOEM
System type	64-bit operating system, x64-based processor
Pen and touch	No pen or touch input is available for this display

Windows specifications

Edition	Windows 11 Home
Version	22H2
Installed on	22/01/2023
OS build	22621.2715
Serial number	PF45GMTA
Experience	Windows Feature Experience Pack 1000.22677.1000.0

Figure 1: System Specification

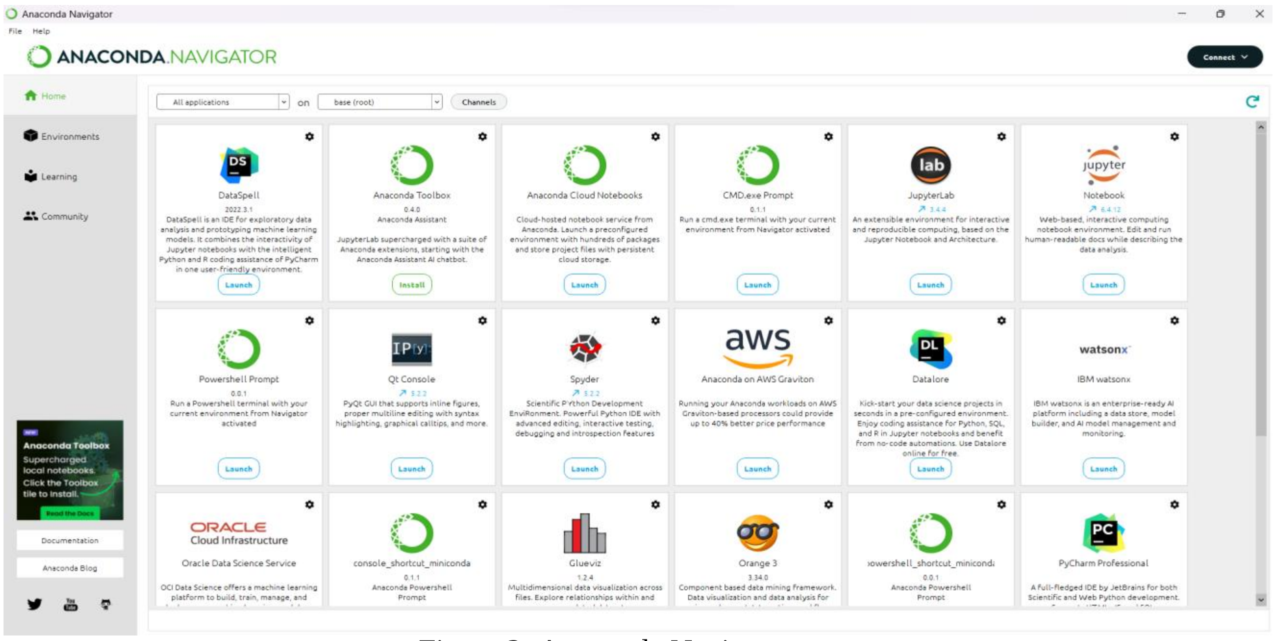
Software Used

- Microsoft Excel: Used for initial exploration into the .csv file.
- Jupyter Notebook: For the modelling and evaluation.
- Github: Used to store, test, and run the Code implementation.

Download and Install

Depending on the operating system, it is necessary to install Python, with the recommendation to use the latest version. For this project, Python 3.10.2 was downloaded and installed for Windows 11, representing the most current version at the time.

Following the installation of Python, a development environment is essential for writing, executing, and observing code output. Jupyter Notebook stands out as a popular and user-friendly option. It is included in the Anaconda Python distribution, which offers downloads tailored to different operating systems. Figure 2 illustrates the Anaconda dashboard, showcasing pre-installed packages, including Jupyter Notebook. To begin coding in Python, one must initiate the Jupyter Notebook and create a new Python file.



DataSpell
2022.3.1
DataSpell is an IDE for exploratory data analysis and prototyping machine learning models. It combines the interactivity of Jupyter notebooks with the intelligent Python and R coding assistance of PyCharm in one user-friendly environment.

Anaconda Toolbox
0.4.0
Anaconda Assistant
JupyterLab supercharged with a suite of Anaconda extensions, starting with the Anaconda Assistant AI chatbot.

Anaconda Cloud Notebooks
Cloud-hosted notebook service from Anaconda. Launch a preconfigured environment with hundreds of packages and store project files with persistent cloud storage.

CMD.exe Prompt
0.1.1
Run a cmd.exe terminal with your current environment from Navigator activated

JupyterLab
3.4.4
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

Notebook
6.4.12
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

Powershell Prompt
0.0.1
Run a Powershell terminal with your current environment from Navigator activated

Qt Console
5.2.2
PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.

Spyder
5.2.2
Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features

Anaconda on AWS Graviton
Running your Anaconda workloads on AWS Graviton-based processors could provide up to 40% better price performance

Datalore
Kick-start your data science projects in seconds in a pre-configured environment. Enjoy coding assistance for Python, SQL, and R in Jupyter notebooks and benefit from no-code automations. Use Datalore online for free.

IBM watsonx
IBM watsonx is an enterprise-ready AI platform including a data store, model builder, and AI model management and monitoring.

Oracle Data Science Service
OCI Data Science offers a machine learning platform to build, train, manage, and

console_shortcut_miniconda
0.1.1
Anaconda Powershell Prompt

Glueviz
1.2.4
Multidimensional data visualization across files. Explore relationships within and

Orange 3
3.34.0
Component based data mining framework. Data visualization and data analysis for

owershell_shortcut_minicond
0.0.1
Anaconda Powershell Prompt

PyCharm Professional
A full-fledged IDE by JetBrains for both Scientific and Web Python development.

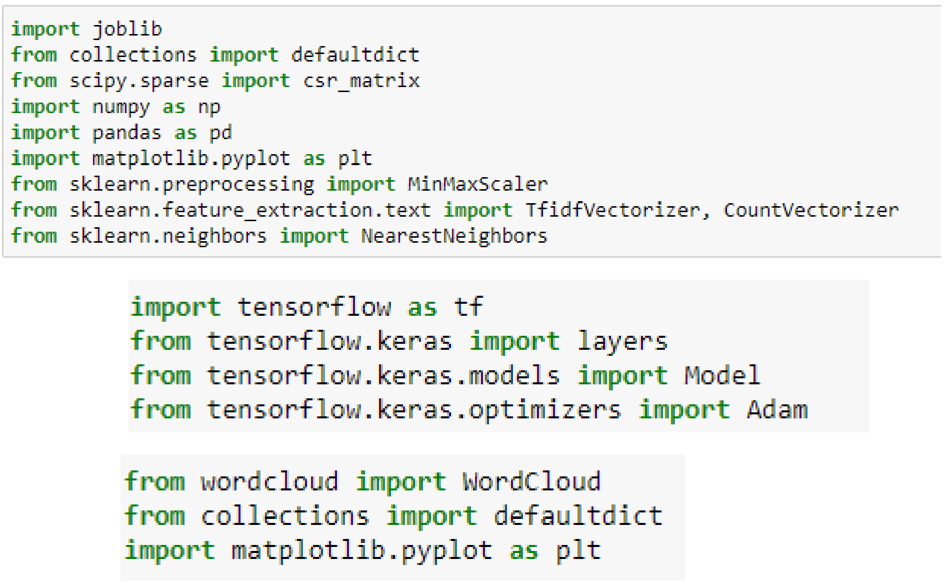
Figure 2: Anaconda Navigator

Project Development

Once these procedures are finished, you can start the Jupyter notebook, select the 'new' button located at the top of the file explorer, and open the scripted file by using the file path given in the code section. This allows you to execute all the cells at once or run them separately. If there's a need to install a package, the command "pip install package-name" should be employed.

Importing Library

The packages used in the project are displayed in Figure 3. The Jupyter Notebook comes with several necessary libraries already installed. If necessary, additional libraries should be imported.



```python
import joblib
from collections import defaultdict
from scipy.sparse import csr_matrix
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
from sklearn.neighbors import NearestNeighbors
```

```python
import tensorflow as tf
from tensorflow.keras import layers
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
```

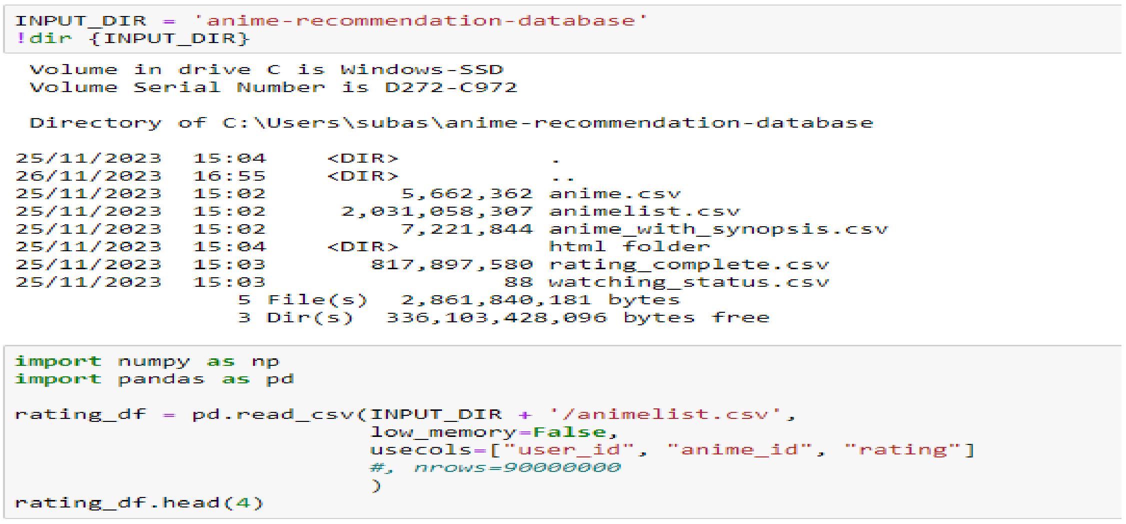
```python
from wordcloud import WordCloud
from collections import defaultdict
import matplotlib.pyplot as plt
```

Figure 3: Packages used in the Project

Various libraries were installed and used for both the models built during the course of the research in Jupyter Notebook such as

- `pandas` - for data processing and reading CSV files.
- `numpy` - for linear algebra operations.
- `seaborn` - for data visualization.
- `matplotlib.pyplot` - for plotting and visualizations.
- `NearestNeighbors` from `sklearn.neighbors` - for implementing nearest neighbors algorithm.
- `cosine_similarity` from `sklearn.metrics.pairwise` - for computing cosine similarity between vectors.
- `train_test_split` from `sklearn.model_selection` - for splitting the dataset into training and testing sets.
- `csr_matrix` from `scipy.sparse` - for creating compressed sparse row matrix.

Importing Files



```
INPUT_DIR = 'anime-recommendation-database'
!dir {INPUT_DIR}
```

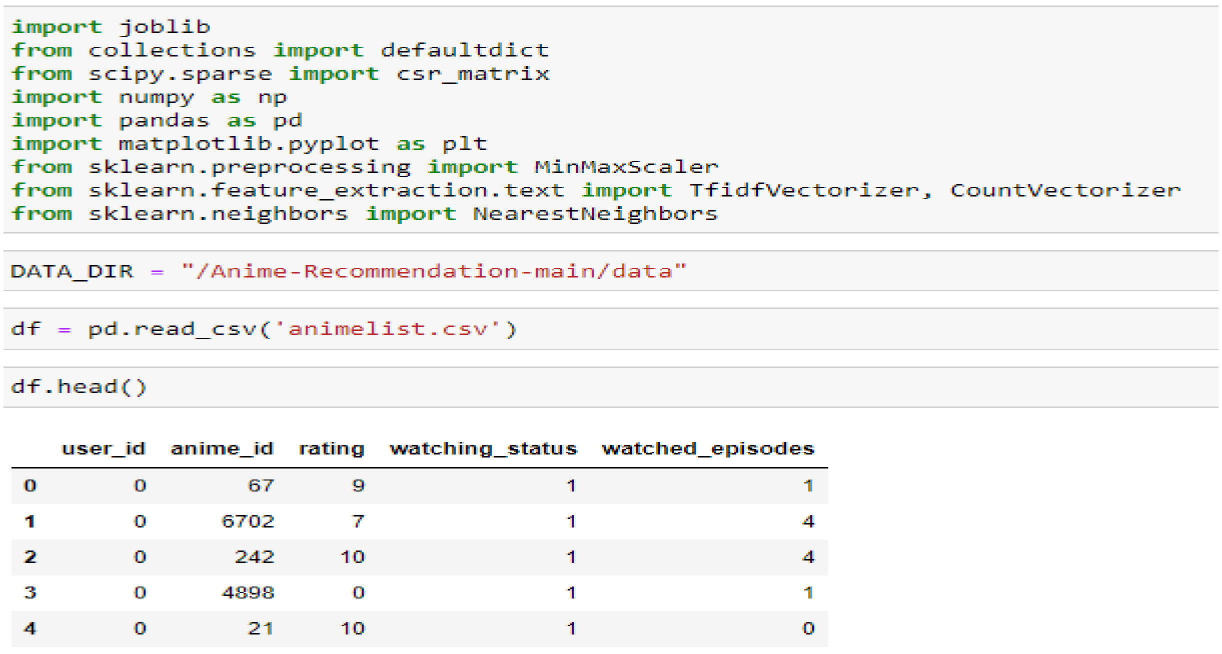
```
 Volume in drive C is Windows-SSD
 Volume Serial Number is D272-C972

 Directory of C:\Users\subas\anime-recommendation-database

25/11/2023  15:04    <DIR>          .
26/11/2023  16:55    <DIR>          ..
25/11/2023  15:02         5,662,362 anime.csv
25/11/2023  15:02     2,031,058,307 animelist.csv
25/11/2023  15:02         7,221,844 anime_with_synopsis.csv
25/11/2023  15:04    <DIR>          html folder
25/11/2023  15:03       817,897,580 rating_complete.csv
25/11/2023  15:03                88 watching_status.csv
               5 File(s)  2,861,840,181 bytes
               3 Dir(s)  336,103,428,096 bytes free
```

```python
import numpy as np
import pandas as pd

rating_df = pd.read_csv(INPUT_DIR + '/animelist.csv',
                        low_memory=False,
                        usecols=["user_id", "anime_id", "rating"]
                        #, nrows=90000000
                        )
rating_df.head(4)
```



```python
import joblib
from collections import defaultdict
from scipy.sparse import csr_matrix
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
from sklearn.neighbors import NearestNeighbors
```

```python
DATA_DIR = "/Anime-Recommendation-main/data"
```

```python
df = pd.read_csv('animelist.csv')
```

```python
df.head()
```

```
   user_id  anime_id  rating  watching_status  watched_episodes
0        0        67       9                1                 1
1        0      6702       7                1                 4
2        0       242      10                1                 4
3        0      4898       0                1                 1
4        0        21      10                1                 0
```

Figure 4: Procedure to Mount and Fetch Document in Jupyter Notebook

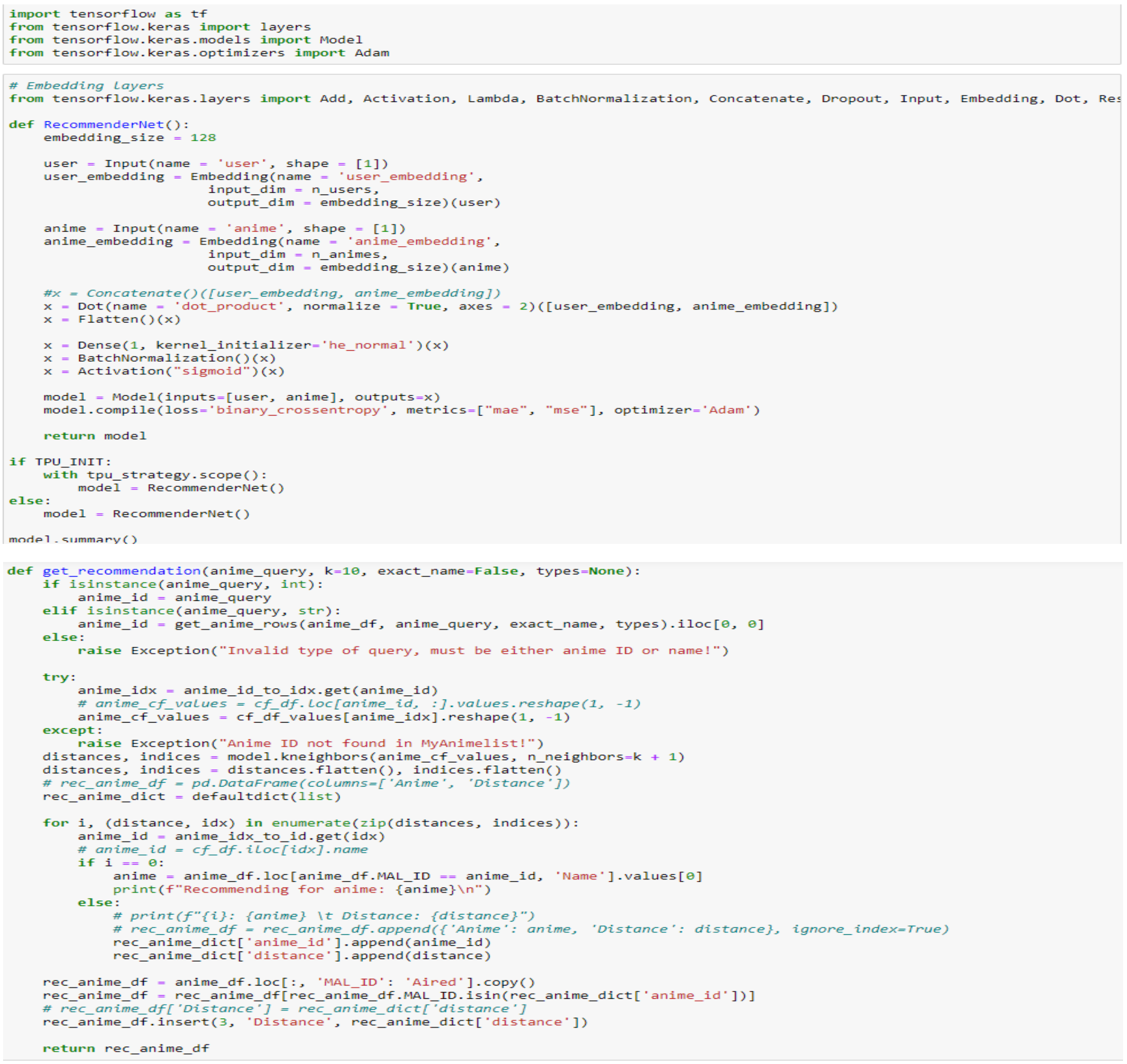
In this research, data processing is exclusively carried out in Jupyter Notebook using .csv files. The process begins by setting the path to the local directory where this file is stored. The file, which is in a .csv format, is then imported into Jupyter Notebook for modeling and evaluation. Figure 4 in the documentation demonstrates how to load the .csv file into the notebook, detailing the essential steps for accessing and effectively utilizing the data within the Jupyter environment.

Processing

- Treatment of Missing Values: Missing values were identified and subsequently removed from the dataset.
- Handling Null Values: The functions IsNull() and sum() are employed to identify null values, while the duplicated() function is utilized to detect and address duplicates in the data.
- Outliers Treatment: Box plots were used to identify the most extreme outliers, which were then removed, with careful consideration given to minimizing data loss.
- Age Restriction: Any data entries with values indicating reviews of Content that are above 18 were excluded from the analysis.
- Selection Criteria for Samples: For this research, only data samples representing users who had reviewed at least 200 anime were included. All other samples were removed from the dataset.

Modeling

The Jupyter Notebook notebook involves preprocessing data, conducting exploratory data analysis (EDA), and then utilizing this data to train and evaluate both models. Further, the dataset was anonymized using Dot product layers and spread out into matrices for better efficiency. After, the preprocessed dataset is now split into two parts, as Test and Train data.



```python
import tensorflow as tf
from tensorflow.keras import layers
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
```

```python
# Embedding layers
from tensorflow.keras.layers import Add, Activation, Lambda, BatchNormalization, Concatenate, Dropout, Input, Embedding, Dot, Res

def RecommenderNet():
    embedding_size = 128

    user = Input(name = 'user', shape = [1])
    user_embedding = Embedding(name = 'user_embedding',
                               input_dim = n_users,
                               output_dim = embedding_size)(user)

    anime = Input(name = 'anime', shape = [1])
    anime_embedding = Embedding(name = 'anime_embedding',
                               input_dim = n_animes,
                               output_dim = embedding_size)(anime)

    #x = Concatenate()([user_embedding, anime_embedding])
    x = Dot(name = 'dot_product', normalize = True, axes = 2)([user_embedding, anime_embedding])
    x = Flatten()(x)

    x = Dense(1, kernel_initializer='he_normal')(x)
    x = BatchNormalization()(x)
    x = Activation("sigmoid")(x)

    model = Model(inputs=[user, anime], outputs=x)
    model.compile(loss='binary_crossentropy', metrics=["mae", "mse"], optimizer='Adam')

    return model

if TPU_INIT:
    with tpu_strategy.scope():
        model = RecommenderNet()
else:
    model = RecommenderNet()

model.summary()
```

```python
def get_recommendation(anime_query, k=10, exact_name=False, types=None):
    if isinstance(anime_query, int):
        anime_id = anime_query
    elif isinstance(anime_query, str):
        anime_id = get_anime_rows(anime_df, anime_query, exact_name, types).iloc[0, 0]
    else:
        raise Exception("Invalid type of query, must be either anime ID or name!")

    try:
        anime_idx = anime_id_to_idx.get(anime_id)
        # anime_cf_values = cf_df.loc[anime_id, :].values.reshape(1, -1)
        anime_cf_values = cf_df_values[anime_idx].reshape(1, -1)
    except:
        raise Exception("Anime ID not found in MyAnimelist!")
    distances, indices = model.kneighbors(anime_cf_values, n_neighbors=k + 1)
    distances, indices = distances.flatten(), indices.flatten()
    # rec_anime_df = pd.DataFrame(columns=['Anime', 'Distance'])
    rec_anime_dict = defaultdict(list)

    for i, (distance, idx) in enumerate(zip(distances, indices)):
        anime_id = anime_idx_to_id.get(idx)
        # anime_id = cf_df.iloc[idx].name
        if i == 0:
            anime = anime_df.loc[anime_df.MAL_ID == anime_id, 'Name'].values[0]
            print(f"Recommending for anime: {anime}\n")
        else:
            # print(f"{i}: {anime} \t Distance: {distance}")
            # rec_anime_df = rec_anime_df.append({'Anime': anime, 'Distance': distance}, ignore_index=True)
            rec_anime_dict['anime_id'].append(anime_id)
            rec_anime_dict['distance'].append(distance)

    rec_anime_df = anime_df.loc[:, 'MAL_ID': 'Aired'].copy()
    rec_anime_df = rec_anime_df[rec_anime_df.MAL_ID.isin(rec_anime_dict['anime_id'])]
    # rec_anime_df['Distance'] = rec_anime_dict['distance']
    rec_anime_df.insert(3, 'Distance', rec_anime_dict['distance'])

    return rec_anime_df
```

Figure 5: Code Snippet of Model building.

This study employs a Jupyter notebook for data preparation and the training of a collaborative filtering (CF) embedding model using neural networks. In this study, there is another notebook that explains how to train a different Collaborative Filtering (CF) model using K-Nearest Neighbor and cosine similarity. These are not the same as neural network-based methods.

Implementation of Code

The code can be downloaded from Github

The dataset can be downloaded here