

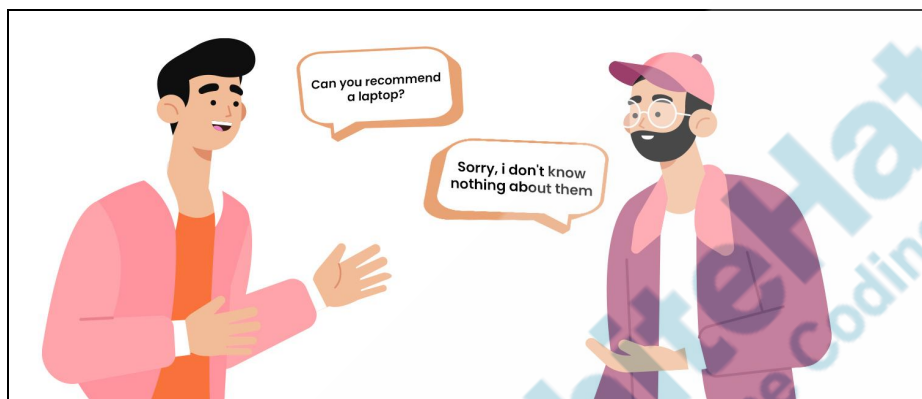


| Topic | MOVIE DATA ANALYSIS | |
|--|--|--|
| Class Description | The student will learn how to import a dataset from Kaggle, understand it and preprocess it. | |
| Class | PRO C138 | |
| Class time | 45 mins | |
| Goal | <ul style="list-style-type: none"> • Loading the dataset into google colab directly from Kaggle. • Understanding the data. • Preprocessing the data. | |
| Resources Required | <ul style="list-style-type: none"> • Teacher Resources: <ul style="list-style-type: none"> ○ Laptop with internet connectivity ○ Earphones with mic ○ Notebook and pen ○ Smartphone • Student Resources: <ul style="list-style-type: none"> ○ Laptop with internet connectivity ○ Earphones with mic ○ Notebook and pen | |
| Class structure | Warm-Up Teacher-Led Activity 1 Student-Led Activity 1 Wrap-Up | 05 mins 15 mins 20 mins 05 mins |
| WARM-UP SESSION - 5 mins | | |
| <div>  </div> <p>Teacher Starts Slideshow</p> <p>Slide 1 to 4</p> <p><Note> Only Applicable for Classes with VA></p> <p>Refer to speaker notes and follow the instructions on each slide.</p> | | |

| Teacher Action | Student Action |
|---|--|
| <p>Hey <student's name>. How are you? It's great to see you! Are you excited to learn something new today?</p> <p><i>Note: Encourage the student to give answers and be more involved in the discussion.</i></p> <p>Following are the WARM-UP session deliverables:</p> <ul style="list-style-type: none"> Greet the student. Revision of previous class activities. Quizzes. | <p>ESR: Hi, thanks! Yes, I am excited about it!</p> <p>Click on the slide show tab and present the slides</p> |
| <p>WARM-UP QUIZ Click on In-Class Quiz</p> | |
| <p></p> <p>Continue WARM-UP Session Slide 5 to 10 <Note: Only Applicable for Classes with VA></p> | |
| <p>Activity Details</p> <p>Following are the session deliverables:</p> <ul style="list-style-type: none"> Appreciate the student. Narrate the story by using hand gestures and voice modulation methods to bring in more interest in students. | |
| Teacher Action | Student Action |
| <p>What do you understand by the term recommendation or suggestion?</p> <p><i>Note: Encourage the student to give answers and connect the answer with today's topic.</i></p> | <p>ESR: Varied</p> |

Great answer. If you look through my perspective, a recommendation or a suggestion is a **selective** or **filtered piece of information** given to someone, provided you have ample amount of knowledge about that topic.

For example, you can't recommend or suggest a laptop to someone, if you don't know anything about laptops.



To understand this better, let's play a simple game, where your friend Alex, needs some recommendations related to certain topics of his interest.

Are you excited to play this quiz?

Note: Open [Teacher Activity 1](#) and play the quiz with the student.

Did you enjoy the quiz?

Great, let's discuss the first question!

ESR: Yes

ESR: Yes

ESR: Yes

Alex wants to loose some weight. Can you recommend one of the activities?

- ☐ Chess
- ☐ Carrom
- ☐ Running
- ☐ I don't know about sports!

Can you tell me the answer?

That is correct. You were able to recommend, because you have the **correct knowledge**, which one of them is a physical activity and which is a mental activity.

Now let's discuss the second question.

Alex is feeling nauseated, feverish , constipated and cold. Can you recommend a medicine?

- ☐ Paracetamol
- ☐ Disprin
- ☐ Digene
- ☐ I don't know, i am not a doctor or pharmacist!

Great, for me, the correct answer is option **D**, because I won't be able to recommend something unless I have proper medical knowledge. That is precisely why we go to the doctors!

Did you notice something peculiar about the last question?

ESR: Yes, he needs to run, if he wants to lose some weight.

ESR: Varied. (Most students might choose option D)

ESR: Yes

ESR: Yes, the last question didn't have

Alex needs a doctor? Can you recommend one?

- ☐ Dietician
- ☐ Skin specialist
- ☐ Physiotherapist
- ☐ Pshycologist

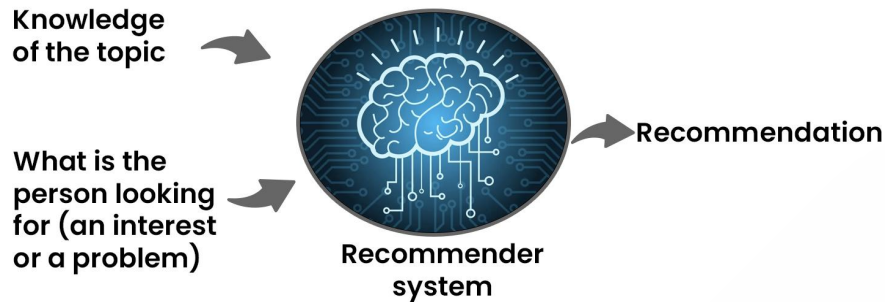
any proper context, so I couldn't give a recommendation.

Great observation, if you observe the last question, even if you have the **proper knowledge** about what kind of doctor one has to consult for a problem, it's meaningless, **if you don't know the person's problem**.

This leads us to a very important conclusion, that even if you have proper knowledge about a particular topic, you can't recommend it unless you don't know what the person is looking for.

So we can summarize that in order to give a recommendation:

- 1) Proper knowledge of the topic is required.
- 2) One should know about the **interest** or the **problem** of the person.



ESR: Varied.

Now, let's try the same principle to a machine.

Have you seen any example where a **machine** or a **software application** is recommending something to a person?

Note: Encourage the student to try and come up with some examples.

Great examples!

Almost every major tech company uses a recommender system in some form or the other:

- a) Amazon uses it to suggest products to customers.
- b) YouTube uses it to decide which video to play next on autoplay.
- c) Facebook uses it to recommend pages to like and people to follow.
- d) Companies like Netflix and Spotify depend highly on the effectiveness of their recommendation engines for their business and success.

Do you use Youtube?

Let's see how **Youtube** uses recommender algorithms to give recommendations to the users.

ESR: Varied. (Mostly Yes)

[Play gif](#)

Before we see how the YouTube recommender algorithm works, let me ask you a simple question. Do you like chocolate cakes?

ESR : Yes

Great. Now, If you see the list of the videos which are being recommended by youtube, there is no video which says '**how to make chocolate cake**', because i have never searched it on youtube, so there are no recommendations around it.

Let's search it on youtube and see if youtube recommends similar videos or not.

ESR: Yes

So open the youtube search bar and write '**how to make chocolate cakes at home**'.

You will see a list of videos on youtube which will show you how to make a chocolate cake at home. Watch some of those videos by fast forwarding them.

Once you have watched some videos, close the youtube and open it again.

Look at the list of the videos which are being recommended by youtube. I am sure that you will find some of them on '**how to make chocolate cakes**'.

To get the results better, make sure you clear your **search** and **watch history** before searching 'how to make a chocolate cake' on youtube.

Isn't that interesting?

ESR : Yes

Now, if we want to make our recommender application, we should keep in mind that our machine should have:

- a) **Proper knowledge** : In case of machines or software **proper knowledge** refers to the '**Details of the item**' or the '**metadata of the item**' which is being **recommended**.
 - **For example**, if a machine wants to recommend movies to the user, it should have information about the '**details**' or the '**metadata**' of the movies, such as **genre, cast, crew, overview** etc.
- b) **Interest or problem of the person getting recommendations** :
In the case of machines or software, the interest or problem of the person refers to **some kind of information about the users**, which will help the machine to give the person better recommendations.
 - **For example**, if a machine wants to recommend movies to a user, knowing about the **age, country, gender, genre that the user likes, actors that the user likes** etc, will help the machine to give proper recommendations.

Can you tell me how to provide the **details of the item which is being recommended**, to the machine?

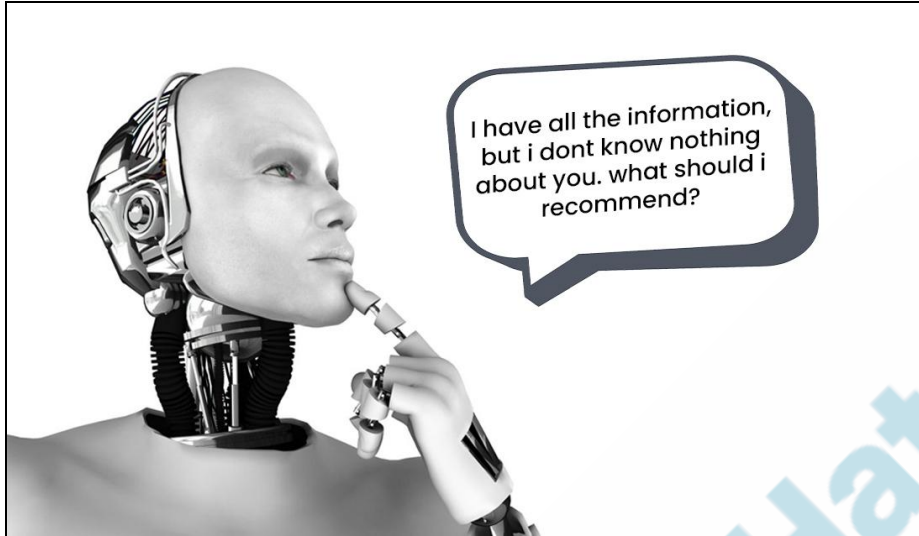
ESR : Varied

Note: Let the student try and answer.

Great. We can use a **dataset** in order to provide the necessary information to the machine.

Can you guess what kind of information about people is helpful for a machine, so that it can give a recommendation or

suggestion to them?



Depending upon what kind of information a machine has, we have 3 types of recommender systems,

1) Demographic filtering method:

- a) This system uses the **demographic data** of a user, which includes **age, race, sex, education, etc.** to decide which movies may be appropriate for recommendations.
- b) Since each **user** is different, this approach is considered **to be too simple**.
- c) The **basic idea** behind this system is that movies that **are more** popular and critically acclaimed will have a **higher** probability of being liked by the average audience.

| SNo. | Movie Name | Popularity Score | Genre |
|------|------------|------------------|--------|
| 1 | Avatar | 100 | Action |

| | | | |
|---|----------|----|--------|
| 2 | Batman | 95 | Action |
| 3 | Avengers | 90 | Action |
| 4 | Iron Man | 85 | Scifi |


- d) **For example:** Consider the following table above, no matter what genre movies you like, this system will recommend **Avatar** as the first movie to watch.

2) Content-based filtering:

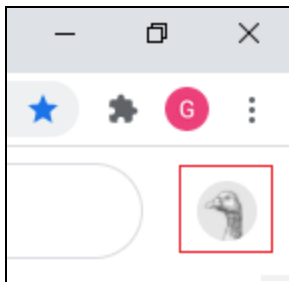
- This system uses movie metadata, such as genre, director, description, actors, etc. for movies, to make these recommendations.
- The general idea behind these recommender systems is that if a person likes a particular movie, he or she will also like a movie that is similar to it.
- For example, if a person likes **Iron Man (Genre: Scifi)**, they will probably like **Iron Man 2** and **Iron Man 3** as well.

3) Collaborative filtering:

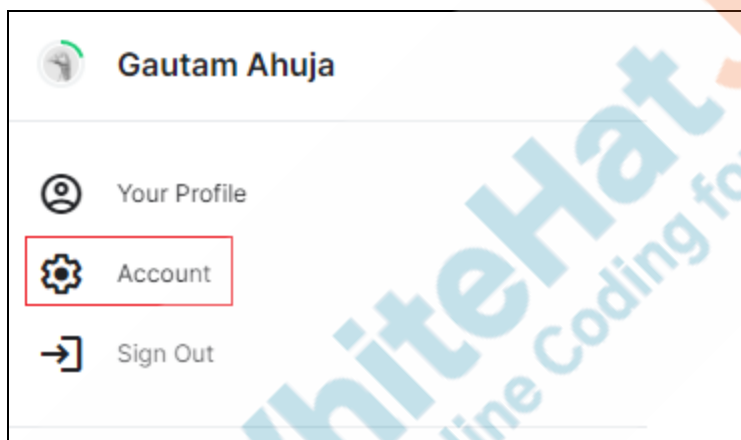
- This system matches persons with similar interests and provides recommendations based on this matching.
- Collaborative filters do not require item metadata like its content-based counterparts.
- For example, if person A likes **Iron Man (Genre: Scifi)** and person B likes **Avatar (Genre: Action)**, it is very likely that person A will get a recommendation for the **Avatar** movie and person B will get a recommendation for the **Iron Man movie**, as both of them have a likeability towards **Action** movies.

| <p>Let's jump into work now! First thing is that we are going to deal with big datasets. It would be a pain to download them first and then re-upload them! How could we directly add the dataset in a Google colab from Kaggle?</p> | |
|--|---------------------------|
| <div>  Teacher Ends Slideshow </div> | |
| TEACHER-LED ACTIVITY - 15 mins | |
| Teacher Initiates Screen Share | |
| <p align="center"><u>ACTIVITY</u></p> <ul style="list-style-type: none"> • Loading data to google colab from Kaggle. | |
| Teacher Action | Student Action |
| <p>In this project, we will be using the following dataset from Kaggle!</p> <p>https://www.kaggle.com/tmdb/tmdb-movie-metadata</p> <p>Do you have any idea what Kaggle is?</p> <p>Kaggle is a platform where you can find and publish any kind of dataset you want.</p> <p>This link contains TMDB's data for 5000 movies.</p> <p>The question still comes: How do we upload this directly into Google colab?</p> <p>The first thing that we want to do is install the dependencies!</p> <p>So, open google colab and in a new notebook, write the command :</p> | <p>ESR: Varied</p> |

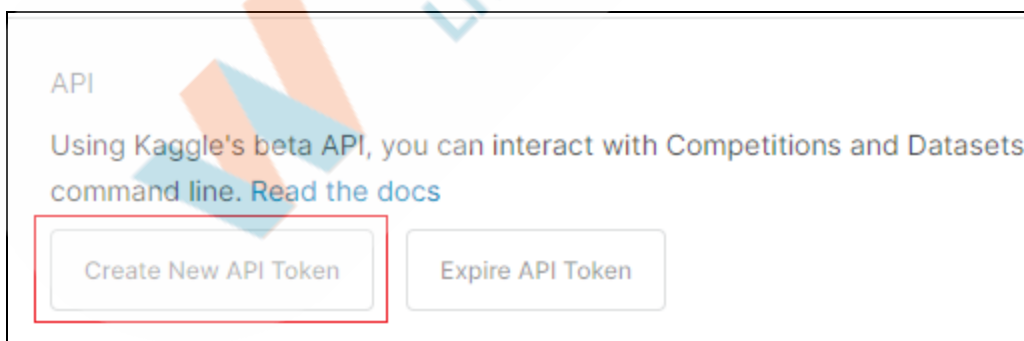
| | |
|--|---|
| <div data-bbox="477 310 750 369" data-label="Code-Block"> <pre>pip install kaggle</pre> </div> <p>Did you notice something strange?</p> <p>Perfect, which means this command is not being recognized by Google Colab. Can you tell me why?</p> <p><i>Note: Encourage the student to give answers as the student has already covered these concepts in earlier classes.</i></p> <p>Great, that's right, if we want to run a terminal command in Google Colab, we use an '!' sign before it. So instead of pip install kaggle, we will write !pip install kaggle</p> | <p>ESR: Yes, there is a red line under the command.</p> <p>ESR: Varied.</p> |
| <div data-bbox="354 905 1218 1203" data-label="Code-Block"> <pre>!pip install kaggle Requirement already satisfied: kaggle in /usr/local/lib/ Requirement already satisfied: urllib3 in /usr/local/lib Requirement already satisfied: python-slugify in /usr/lo Requirement already satisfied: requests in /usr/local/li Requirement already satisfied: certifi in /usr/local/lib Requirement already satisfied: six>=1.10 in /usr/local/l</pre> </div> | |
| <p>Awesome! Now we have installed the kaggle module and the next step is to generate the API key from the kaggle website.</p> <p>The API key will authenticate your request for downloading a dataset from kaggle, just like your Gmail login details authenticate an access request to your account.</p> <p>For that, sign up with the Kaggle website. Once your account is created, click on the icon which is on the top right corner.</p> | |




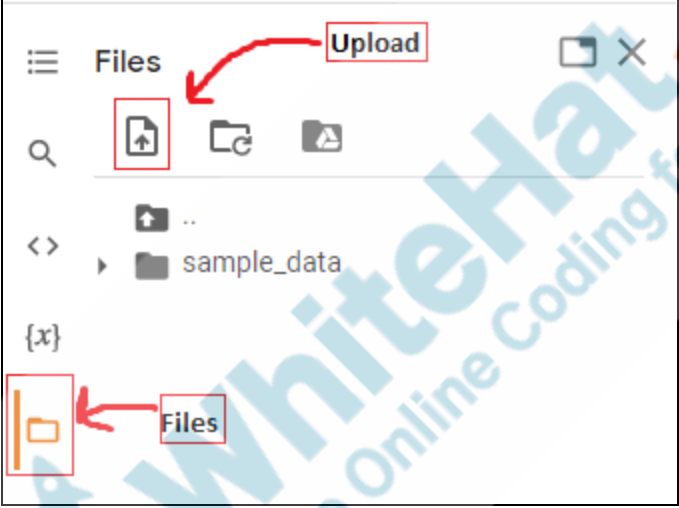
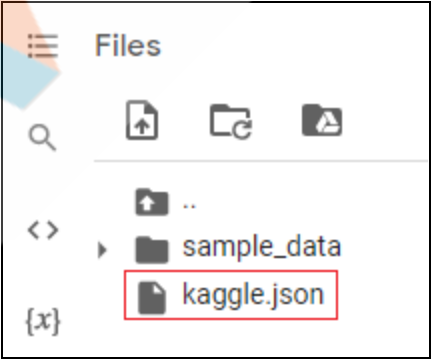
A menu will pop up. Click on **Account**.

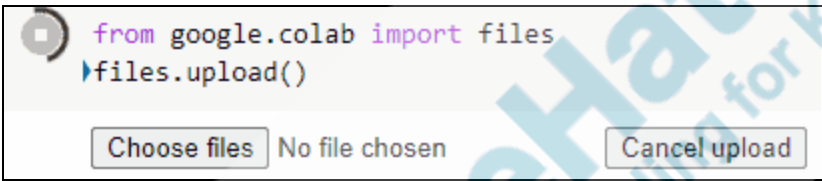



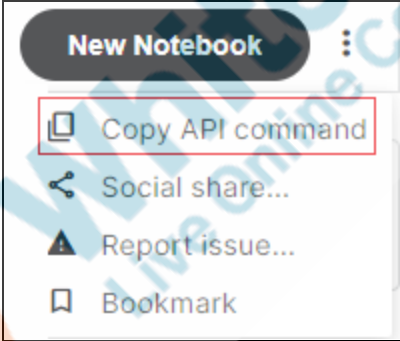
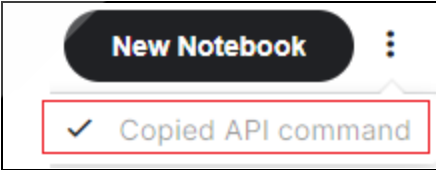
A new page will load. Look for the API header and once you find it, click on **Create New API Token** button.



A file named '**kaggle.json**' will download which has the API key required for authentication.

| | | | |
|---|------------------|------------------|------|
|  kaggle | 22-03-2022 16:45 | JSON Source File | 1 KB |
| <p>Once you have the 'kaggle.json' file, you can upload it by using one of the two methods.</p> | | | |
| <p>Method 1:</p> <ol style="list-style-type: none"> Click on the Files option. Click on the upload button. A window will open. Select 'kaggle.json' file. | | | |
|  | | | |
| <p>Your file will be uploaded.</p> | | | |
|  | | | |
| <p>You can also upload the 'kaggle.json' file by writing a small code as,</p> | | | |

| | |
|--|-------------------------------------|
| <ul style="list-style-type: none"> • Import the files object from google.colab directory. • Use the .upload() method, to upload a file into Google colab. | |
| <pre>from google.colab import files files.upload()</pre> | |
| <p>Once you run this code, it will give you an option to Choose files from your local system or to Cancel upload.</p> <p><i>Note : Until and unless you select one of the options, it's going to wait there forever.</i></p> | |
|  | |
| <p>Select Choose files.</p> <p>A window will pop up. Select 'kaggle.json' file and click on OK. Your file will be uploaded.</p> | |
| <pre>from google.colab import files files.upload()</pre> <p>Choose files kaggle.json</p> <ul style="list-style-type: none"> • kaggle.json(application/json) - 67 bytes, last modified: 08/02/2022 - 100% done <p>Saving kaggle.json to kaggle.json</p> <pre>{'kaggle.json': b'{"username": "gautamahuja", "key": "1ab834b6cef531b5d851831f7b020b2e"}'}</pre> | |
| <p>After uploading the 'kaggle.json' file, we can use a simple terminal command to download the dataset:</p> <p>!kaggle datasets download -d [DATASET_NAME]</p> <p>Can you try and frame the command for our dataset?</p> <p><i>Note: Let the student try and frame the command by themselves.</i></p> | <p>ESR: Sure, I can try.</p> |

| | |
|--|--|
| <p>To download the dataset into colab, run this terminal command : ! kaggle datasets download -d tmdb/tmdb-movie-metadata.</p> <p>There might be a small chance of writing incorrect dataset names, which might lead to some unexpected problems. So instead of writing this command by yourself, you can copy it by following some simple steps: Open the dataset link : https://www.kaggle.com/tmdb/tmdb-movie-metadata And click on more options button, depicted by 3 vertical dots.</p> | |
|  | |
| <p>A dropdown window will appear from which we have to select the Copy API command.</p> | |
|  | |
| <p>Once you have selected it, it will show that you have Copied API command.</p> | |
|  | |
| <p>Once you have copied the command, switch back to colab and paste the command in a new cell.</p> | |


```
kaggle datasets download -d tmdb/tmdb-movie-metadata
```

Don't forget to put the **‘!’ sign** at the beginning while running a terminal command.

```
!kaggle datasets download -d tmdb/tmdb-movie-metadata
```

Traceback (most recent call last):

File "/usr/local/bin/kaggle", line 5, in <module>

from kaggle.cli import main

File "/usr/local/lib/python2.7/dist-packages/kaggle/__init__.py", line 23, in <module>

api.authenticate()

File "/usr/local/lib/python2.7/dist-packages/kaggle/api/kaggle_api_extended.py", line 146, in authenticate

self.config_file, self.config_dir))

IOError: Could not find kaggle.json. Make sure it's located in /root/.kaggle. Or use the environment method.

Snap! Does it show an error? Can you try and tell me what went wrong?

ESR: Varied.

If you can read the error, it says **Could not find kaggle.json**. **Make sure it's located in /root/.kaggle. Or use the environment method**, which means our program looks for the authentication file, the **“kaggle.json”**, in a very specific folder. So in order to avoid the error, let's create the required directory and move our file into that folder.

To make a new directory, use the terminal command :

! mkdir -p ~/.kaggle

If we put the **‘.’ operator** before a directory name, it means we are trying to create a **hidden directory**.



```
!mkdir -p ~/.kaggle
```

After creating the directory, it's time to copy our **“kaggle.json”** file into our hidden directory using the terminal command:

! cp kaggle.json ~/.kaggle/

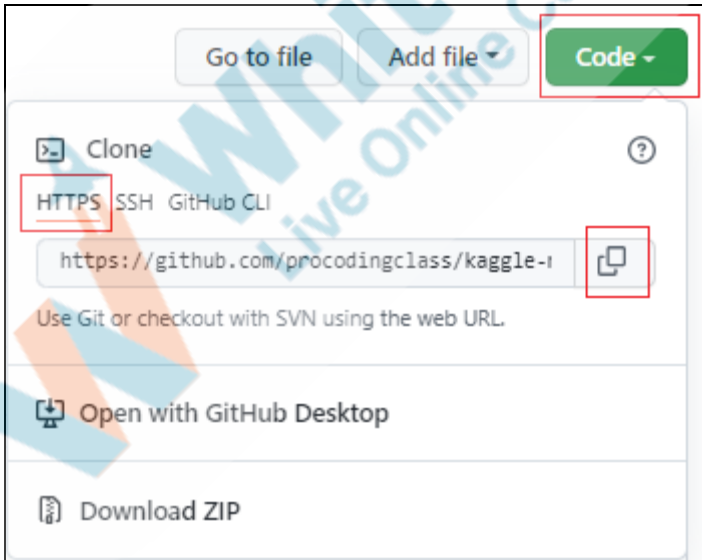
```
!mkdir -p ~/.kaggle
!cp kaggle.json ~/.kaggle/
```

| | |
|--|----------------------------|
| <p>Our kaggle directory is a very private folder which holds our authentication details and it shouldn't be accessible to anyone. So let's change the mode (chmod) of that directory using the terminal command:</p> <p>! chmod 600 ~/.kaggle/kaggle.json</p> <p>Here 600 permission code means that only the owner of the file has full read and write access to it. Once file permission is set to 600, no one else can access the file.</p> | |
| <pre>!mkdir -p ~/.kaggle !cp kaggle.json ~/.kaggle/ !chmod 600 ~/.kaggle/kaggle.json</pre> | |
| <p>After moving our authentication file into the required directory, run the terminal command to download the dataset.</p> | |
| <pre>!kaggle datasets download -d tmdb/tmdb-movie-metadata Downloading tmdb-movie-metadata.zip to /content 56% 5.00M/8.89M [00:00<00:00, 11.4MB/s] 100% 8.89M/8.89M [00:00<00:00, 18.1MB/s]</pre> | |
| <p>After downloading the dataset, let's list all the files, using the terminal command ! ls, so that we can see all the files in our working directory.</p> | |
| <pre>!ls kaggle.json sample_data tmdb-movie-metadata.zip</pre> | |
| <p>Our dataset is in zipped format. To unzip it, use the terminal command ! unzip [FILE_NAME]</p> <p>Can you try and frame this command?</p> <p>To unzip it, use the command :</p> <p>! unzip tmdb-movie-metadata.zip</p> | <p>ESR: Varied.</p> |

| | |
|---|--|
| <pre>!unzip tmdb-movie-metadata.zip Archive: tmdb-movie-metadata.zip inflating: tmdb_5000_credits.csv inflating: tmdb_5000_movies.csv</pre> | |
| Our unzipped dataset has 2 files, tmdb_5000_credits.csv and tmdb_5000_movies.csv . | |
| We have successfully imported our dataset into colab. | |
| Teacher Stops Screen Share | |
| So now it's your turn. Please share your screen with me. | |
| <p>Teacher Starts Slideshow </p> <p>Slide 11 to 14</p> <p><Note: Only Applicable for Classes with VA></p> <p>Refer to speaker notes and follow the instructions on each slide.</p> | |
| We have one more class challenge for you. Can you solve it? | |
| Let's try. I will guide you through it. | |
| <p>Teacher Ends Slideshow </p> | |
| STUDENT-LED ACTIVITY - 20 mins | |
| <ul style="list-style-type: none"> • Ask the student to press the ESC key to come back to the panel. • Guide the student to start Screen Share. • The teacher gets into Full Screen. | |
| Student Initiates Screen Share | |

ACTIVITY

- Students will understand the data.
- Students will preprocess the data.
- Students will merge and clean the data.

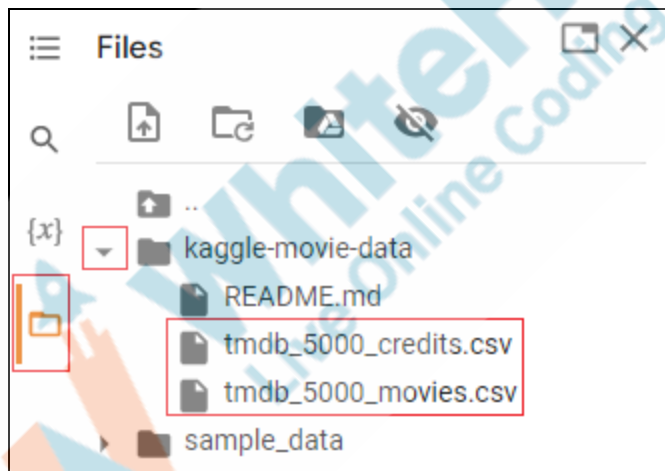
| Teacher Action | Student Action |
|--|----------------|
| <p>Now, you can follow the same process to get tmdb_5000_credits.csv and tmdb_5000_movies.csv files directly from Kaggle.</p> <p>But for your convenience, we have added these two files in a GitHub repository so that you don't have to go over this process again and again.</p> <p>Open the above GitHub link and click on the Code button. Click on the HTTPS tab, and finally to copy the link, click on the copy button.</p> | |
|  | |
| <p>Once you have copied the link, open the Student boilerplate link, and in a new cell, write the command,</p> <p>!git clone https://github.com/procodingclass/kaggle-movie-data.git,</p> | |

and run it.

```
# downloading files from github repo or from kaggle
!git clone https://github.com/procodingclass/kaggle-movie-data.git

Cloning into 'kaggle-movie-data'...
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 7 (delta 0), reused 4 (delta 0), pack-reused 0
Unpacking objects: 100% (7/7), done.
```

To check if the files are downloaded or not, go to the **files** option, **expand** the **kaggle-movie-data** directory and you will see that **tmdb_5000_credits.csv** and **tmdb_5000_movies.csv** files are downloaded.



Now, it's time to create a DataFrame.

Do you remember how to create a DataFrame?

To create a **DataFrame**, in a new cell, import the **pandas** module and write the command,

DataFrame = pd.read_csv[FILE_PATH]

Can you try and frame the command?

ESR: Varied.

ESR: Sure.

To create a dataframe, use the command:

```
credits_df = pd.read_csv('tmdb_5000_credits.csv')
movies_df = pd.read_csv('tmdb_5000_movies.csv')
```

```
import pandas as pd
credits_df = pd.read_csv('tmdb_5000_credits.csv')
movies_df = pd.read_csv('tmdb_5000_movies.csv')
```

Now, let's print the first 5 rows of the dataframe using the `.head()` function.

```
credits_df.head()
```

| credits_df.head() | | |
|-------------------|----------|--|
| | movie_id | title |
| 0 | 19995 | Avatar |
| 1 | 285 | Pirates of the Caribbean: At World's End |
| 2 | 206647 | Spectre |
| 3 | 49026 | The Dark Knight Rises |
| 4 | 49529 | John Carter |

Let's have a look at our **credits** dataset. It contains the following features:

- **movie_id** - A unique identifier for each movie.
- **title** - Name of the movie.
- **cast** - The name of lead and supporting actors.
- **crew** - The name of Director, Editor, Composer, Writer, etc.

Similarly, for movies dataframe, write the command,
movies_df.head()

```
movies_df.head()
```

| | budget | genres | homepage | id |
|---|-----------|--|--|--------|
| 0 | 237000000 | [{"id": 28, "name": "Action"}, {"id": 12, "name": "Adventure"}] | http://www.avatarmovie.com/ | 19995 |
| 1 | 300000000 | [{"id": 12, "name": "Adventure"}, {"id": 14, "name": "Fantasy"}] | http://disney.go.com/disneypictures/pirates/ | 285 |
| 2 | 245000000 | [{"id": 28, "name": "Action"}, {"id": 12, "name": "Adventure"}] | http://www.sonypictures.com/movies/spectre/ | 206647 |

Our movies dataset has the following columns:

- **budget** - The budget in which the movie was made.
- **genre** - The genre of the movie, Action, Comedy, Thriller, etc.
- **homepage** - A link to the homepage of the movie.
- **id** - This is the **movie_id** as in the first dataset.
- **keywords** - The keywords or tags related to the movie.
- **original_language** - The language in which the movie was made.
- **original_title** - The title of the movie before translation or adaptation.
- **overview** - A brief description of the movie.

| | |
|--|----------------------------|
| <ul style="list-style-type: none"> ● popularity - A numeric quantity specifying the movie's popularity. ● production_companies - The production house of the movie. ● production_countries - The country in which it was produced. ● release_date - The date on which it was released. ● revenue - The worldwide revenue generated by the movie. ● runtime - The running time of the movie in minutes. ● status - "Released" or "Rumored". ● tagline - Movie's tagline. ● title - Title of the movie. ● vote_average - average ratings the movie received. ● vote_count - the count of votes received. | |
| <p>It is important to learn about the dimensions of our datasets. Can you tell me how we can calculate the total number of rows and columns in our dataset?</p> <p><i>Note: Let the student try and answer.</i></p> <p>Great, we can use the .shape property of the dataframe, which will return the shape or dimension of a dataframe in the form of a tuple as,</p> <pre>print("Credits dataframe shape : ", credits_df.shape) print("Movies dataframe shape : ", movies_df.shape)</pre> <p>We can see that our credits dataframe has 4803 rows and 4 columns, whereas our movies dataframe has 4803 rows and 20 columns.</p> | <p>ESR: Varied.</p> |


```
print("Credits dataframe : " , credits_df.shape)
print("Movies dataframe : " , movies_df.shape)

Credits dataframe : (4803, 4)
Movies dataframe : (4803, 20)
```

For easy processing, we have to combine both the dataframes. Can you look for a common column in both the dataframes?

Great, you spotted it correctly. But, there is a challenge, the column in the credits dataframe is named as **movie_id** and the column in the movies dataframe is named as **id**.

To merge them we need to have a common column name, so let's rename the column of credits dataframe as **id**.

To do so, use the command, **credits_df.rename(columns = {'movie_id' : 'id'}, inplace = True)**, which will rename the column name from **movie_id** to **id**.

Note: This command will make the changes in the original dataset itself (as we have written `inplace = True`).

To verify, let's print the first **5** rows of our credits dataframe using the **.head()** function.

ESR: Yes, they both have a common **movie_id** column.

```
credits_df.rename(columns = {'movie_id' : 'id'} , inplace = True)
credits_df.head()
```

| | id | title | cast |
|---|--------|--|---|
| 0 | 19995 | Avatar | [{"cast_id": 242, "character": "Jake Sully", "... |
| 1 | 285 | Pirates of the Caribbean: At World's End | [{"cast_id": 4, "character": "Captain Jack Spa... |
| 2 | 206647 | Spectre | [{"cast_id": 1, "character": "James Bond", "cr... |
| 3 | 49026 | The Dark Knight Rises | [{"cast_id": 2, "character": "Bruce Wayne / Ba... |
| 4 | 49529 | John Carter | [{"cast_id": 5, "character": "John Carter", "c... |

After changing the column name, let's merge both the datasets as,
result_df = pd.merge(movies_df , credits_df , on = 'id'),
 which will combine both the datasets with reference to a common column named 'id'.

```
result_df = pd.merge(movies_df , credits_df , on = 'id')
```

One quick way to check if both the dataframes have merged or not, is to check the column count. So let's print the shape of our dataset using the **result_df.shape** command.



Great, we can see that our column count has increased from **20** to **23**, which means we have successfully merged our datasets.

```
result_df.shape
(4803, 23)
```

Great! Now, it looks like our data is ready!

Teacher Guides Student to Stop Screen Share

WRAP-UP SESSION - 05 mins

| | |
|---|-----------------------|
| <div>  <p>Teacher Starts Slideshow Slide 15 to 20 <Note: Only Applicable for Classes with VA></p> </div> | |
| <p>Activity details</p> <p>Following are the WRAP-UP session deliverables:</p> <ul style="list-style-type: none"> • Appreciate the student. • Revise the current class activities. • Discuss the quizzes. | |
| <p>WRAP-UP QUIZ Click on In-Class Quiz</p> | |
| <div>  <p>Continue WRAP-UP Session Slide 21 to 27 <Note: Only Applicable for Classes with VA></p> </div> | |
| <p>Activity Details</p> <p>Following are the session deliverables:</p> <ul style="list-style-type: none"> • Explain the facts and trivia • Next class challenge • Project for the day • Additional Activity (Optional) | |
| <p><u>FEEDBACK</u></p> <ul style="list-style-type: none"> • Appreciate and compliment the student for trying to learn a difficult concept. • Get to know how they are feeling after the session. • Review and check their understanding. | |
| Teacher Action | Student Action |

You get “hats-off” for your excellent work!

In the next class, we'll be making our own Demographic filtering recommender system.

Make sure you have given at least 2 hats-off during the class for:

Creatively Solved Activities  +10

Great Question  +10

Strong Concentration  +10

PROJECT OVERVIEW DISCUSSION

Refer the document below in Activity Links Sections

Teacher Clicks

✕ End Class

ACTIVITY LINKS

| Activity Name | Description | Links |
|--------------------|----------------|---|
| Teacher Activity 1 | Quiz link | https://docs.google.com/forms/d/e/1FAIpQLSfXjqirQoKjf-7UtZII0BNvdQWdxL0bCeokfmLK7eMh2BvJ0Q/viewform |
| Teacher Activity 2 | Colab link | https://colab.research.google.com/?utm_source=scs-index |
| Teacher Activity 3 | Reference Code | https://colab.research.google.com/drive/1j_fCoOVXq2Wb5rYSeNYVEnP66loSWVEv?usp=sharing |

| | | |
|---------------------|------------------|---|
| Teacher Reference 1 | Project | https://s3-whjr-curriculum-uploads.whjr.online/a71566fc-12cf-4c68-ba79-4ccb0c7ff52.pdf |
| Teacher Reference 2 | Project Solution | https://colab.research.google.com/drive/1zERpGGvswm6lbiUVkJbELbyaJ3qHqvgb?usp=sharing |
| Teacher Reference 3 | Visual-Aid | Will be added after VA creation |
| Teacher Reference 4 | In-Class Quiz | https://s3-whjr-curriculum-uploads.whjr.online/2c328251-db12-41fe-aede-1bfa5e7e839a.pdf |
| Student Activity 1 | Boilerplate Code | https://colab.research.google.com/drive/1Hwg-krLw4aPDW1PXnSQHYVbREUu9_QR1?usp=sharing |