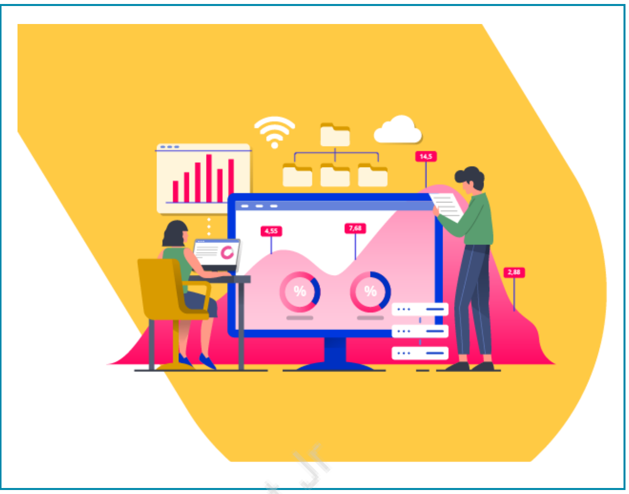
PRO-C140

Content Based Filtering



What is our GOAL for this MODULE?

The goal of this module is to understand how content based filtering is done and then perform it.

What did we ACHIEVE in the class TODAY?

- Understood how content based filtering is done
- Learned about Cosine Similarity
- Performed content based filtering

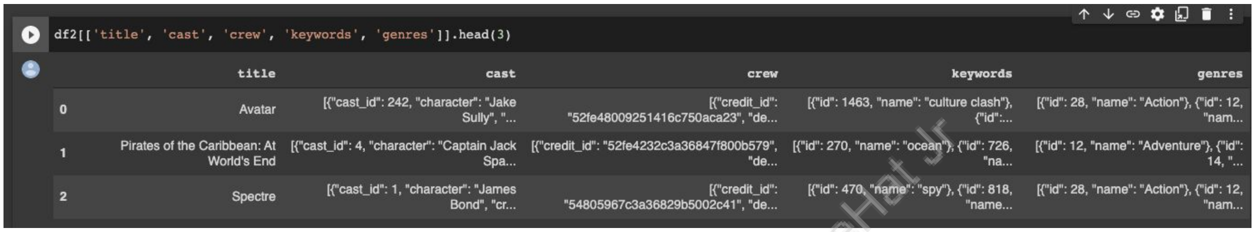
Which CONCEPTS/CODING BLOCKS did we cover today?

- Content Based Filtering
- Cosine Similarity
- Pandas DataFrame

How did we DO the activities?

1. We will work with the following features for our content based filtering:
    - **Cast**
    - **Crew**
    - **Keywords**
    - **Genres**

```
df2[['title', 'cast', 'crew', 'keywords', 'genres']].head(3)
```



```
df2[['title', 'cast', 'crew', 'keywords', 'genres']].head(3)
```

	title	cast	crew	keywords	genres
0	Avatar	[{"cast_id": 242, "character": "Jake Sully", "...	[{"credit_id": "52fe48009251416c750aca23", "de...	[{"id": 1463, "name": "culture clash"}, {"id":...	[{"id": 28, "name": "Action"}, {"id": 12, "nam...
1	Pirates of the Caribbean: At World's End	[{"cast_id": 4, "character": "Captain Jack Spa...	[{"credit_id": "52fe4232c3a36847f800b579", "de...	[{"id": 270, "name": "ocean"}, {"id": 726, "na...	[{"id": 12, "name": "Adventure"}, {"id": 14, "...
2	Spectre	[{"cast_id": 1, "character": "James Bond", "cr...	[{"credit_id": "54805967c3a36829b5002c41", "de...	[{"id": 470, "name": "spy"}, {"id": 818, "name...	[{"id": 28, "name": "Action"}, {"id": 12, "nam...

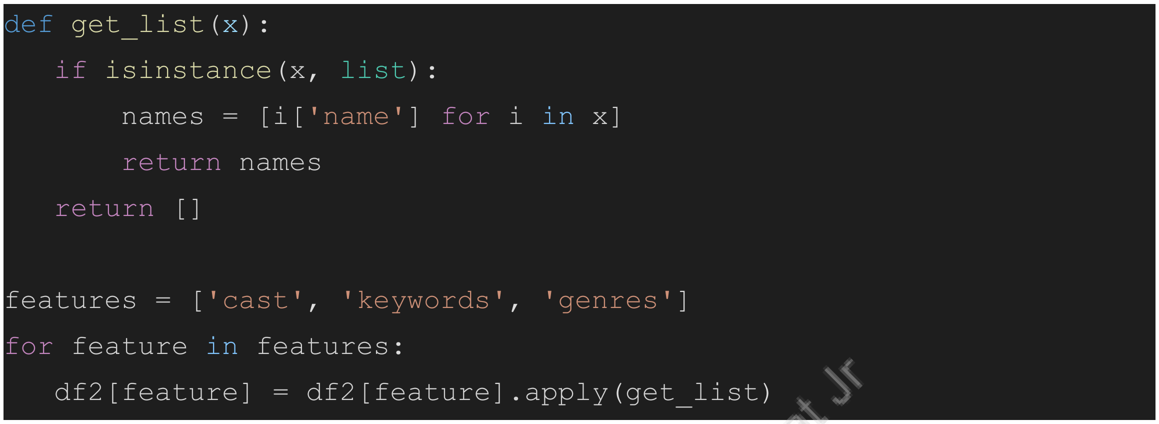
2. We observe the data is supposed to be in the form of a list of dictionaries. There might be some data that is a list of dictionaries too, but in a string. Convert all the rows into a list of dictionaries.

```
from ast import literal_eval

features = ['cast', 'crew', 'keywords', 'genres']
for feature in features:
   df2[feature] = df2[feature].apply(literal_eval)

df2.dtypes
```

3. Create a new column “director” and in this column, find the name of the director from the “crew” data:

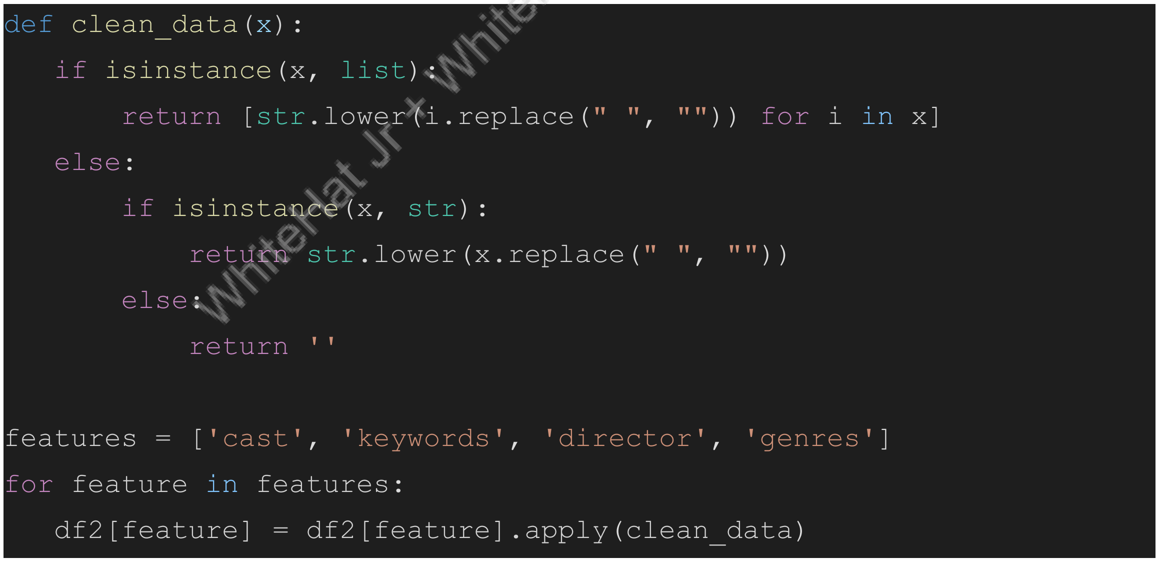
```
def get_director(x):
   for i in x:
       if i['job'] == 'Director':
           return i['name']
   return np.nan

df2['director'] = df2['crew'].apply(get_director)
```

4. Convert the list of dictionaries into simple lists in columns **cast, keywords and genres:**



```python
def get_list(x):
    if isinstance(x, list):
        names = [i['name'] for i in x]
        return names
    return []

features = ['cast', 'keywords', 'genres']
for feature in features:
    df2[feature] = df2[feature].apply(get_list)
```

5. To perform content based filtering, we will create a string of metadata (info about keywords, actors, director and genres) and we will compare these strings to find similarity between them. Now to create these strings, we want to make sure our data is clean. This means that all things should be converted to lowercase and spaces between these elements should be removed.



```python
def clean_data(x):
    if isinstance(x, list):
        return [str.lower(i.replace(" ", "")) for i in x]
    else:
        if isinstance(x, str):
            return str.lower(x.replace(" ", ""))
        else:
            return ''

features = ['cast', 'keywords', 'director', 'genres']
for feature in features:
    df2[feature] = df2[feature].apply(clean_data)
```

6. Create the metadata string with keywords, cast, genres and director columns and save this string into a new column named “soup”.

```
def create_soup(x):
   return ' '.join(x['keywords']) + ' ' + ' '.join(x['cast']) + ' '
+ x['director'] + ' ' + ' '.join(x['genres'])
df2['soup'] = df2.apply(create_soup, axis=1)
```

7. Remove stop words from this metadata string (words like **and, but, the, etc.**) and count the number of occurrences of each of the word in the string (use count vectorizer from sklearn library).

```
from sklearn.feature_extraction.text import CountVectorizer
count = CountVectorizer(stop_words='english')
count_matrix = count.fit_transform(df2['soup'])
```

8. Import cosine similarity function from sklearn library and create a classifier.

```
from sklearn.metrics.pairwise import cosine_similarity
cosine_sim2 = cosine_similarity(count_matrix, count_matrix)
```

9. Next, change the index of our movie data to the name of the movies.

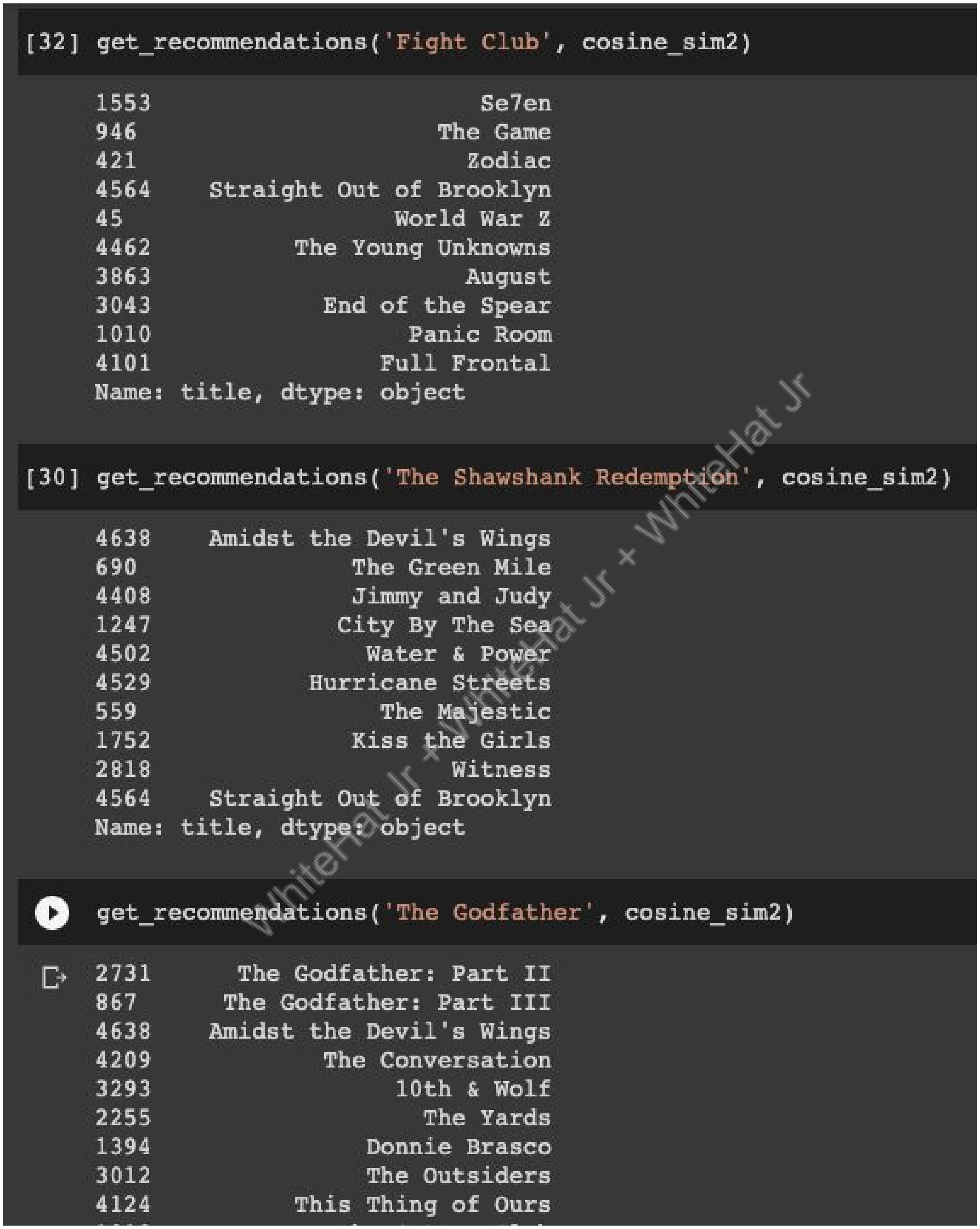
```
df2 = df2.reset_index()
indices = pd.Series(df2.index, index=df2['title'])
```

10. Finally, create the function to get top 10 most recommended movies based on similarity scores stored in our classifier:

```
def get_recommendations(title, cosine_sim):
   idx = indices[title]
   sim_scores = list(enumerate(cosine_sim[idx]))
   sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
   sim_scores = sim_scores[1:11]
   movie_indices = [i[0] for i in sim_scores]
   return df2['title'].iloc[movie_indices]
```

11. Test your code:

```
get_recommendations('Fight Club', cosine_sim2)
get_recommendations('The Shawshank Redemption', cosine_sim2)
get_recommendations('The Godfather', cosine_sim2)
```



```
[32] get_recommendations('Fight Club', cosine_sim2)

     1553                       Se7en
     946                     The Game
     421                       Zodiac
     4564    Straight Out of Brooklyn
     45                   World War Z
     4462          The Young Unknowns
     3863                      August
     3043            End of the Spear
     1010                  Panic Room
     4101               Full Frontal
     Name: title, dtype: object
```

```
[30] get_recommendations('The Shawshank Redemption', cosine_sim2)

     4638    Amidst the Devil's Wings
     690               The Green Mile
     4408              Jimmy and Judy
     1247             City By The Sea
     4502               Water & Power
     4529          Hurricane Streets
     559                 The Majestic
     1752              Kiss the Girls
     2818                     Witness
     4564    Straight Out of Brooklyn
     Name: title, dtype: object
```

```
     get_recommendations('The Godfather', cosine_sim2)

     2731       The Godfather: Part II
     867       The Godfather: Part III
     4638    Amidst the Devil's Wings
     4209            The Conversation
     3293                 10th & Wolf
     2255                   The Yards
     1394               Donnie Brasco
     3012             The Outsiders
     4124          This Thing of Ours
```

What’s NEXT?

In the next class, we will begin the cap-stone project where we will be using all these analysis and create a mobile app that can recommend great movies to our users based on their preferences!