ratings = pd.read\_csv('file path', sep='\t', encoding='latin-1', usecols=['', ''])

users = pd.read\_csv('file path', sep='\t', encoding='latin-1', usecols=[''])

movies = pd.read\_csv('file path', sep='\t',encoding='latin-1', usecols=[''])

--------------------

genre\_labels = set()

for s in movies['genres'].str.split('|').values:

genre\_labels = genre\_labels.union(set(s))

# Function that counts the number of times each of the genre keywords appear

def count\_word(dataset, ref\_col, census):

keyword\_count = dict()

for s in census:

keyword\_count[s] = 0

for census\_keywords in dataset[ref\_col].str.split('|'):

if type(census\_keywords) == float and pd.isnull(census\_keywords):

continue

for s in [s for s in census\_keywords if s in census]:

if pd.notnull(s):

keyword\_count[s] += 1

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# convert the dictionary in a list to sort the keywords by frequency

keyword\_occurences = []

for k,v in keyword\_count.items():

keyword\_occurences.append([k,v])

keyword\_occurences.sort(key = lambda x:x[1], reverse = True)

return keyword\_occurences, keyword\_count

# Calling this function gives access to a list of genre keywords which are sorted by decreasing frequency

keyword\_occurences, dum = count\_word(movies, 'genres', genre\_labels)

keyword\_occurences[:5]

-----------------------------------

# Break up the big genre string into a string array

movies['genres'] = movies['genres'].str.split('|')

# Convert genres to string value

movies['genres'] = movies['genres'].fillna("").astype('str')

-----------------------------------------------

from sklearn.feature\_extraction.text import TfidfVectorizer

tf = TfidfVectorizer(analyzer='word',ngram\_range=(1, 2),min\_df=0, stop\_words='english')

tfidf\_matrix = tf.fit\_transform(movies['genres'])

tfidf\_matrix

-------------------------------------------------

from sklearn.metrics.pairwise import linear\_kernel

cosine\_sim = linear\_kernel(tfidf\_matrix, tfidf\_matrix)

cosine\_sim[:4, :4]

-----------------------------------

# Build a 1-dimensional array with movie titles

titles = movies['title']

indices = pd.Series(movies.index, index=movies['title'])

-----------------------------------------------------------

# Function that get movie recommendations based on the cosine similarity score of movie genres

def genre\_recommendations(title):

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[1:21]

movie\_indices = [i[0] for i in sim\_scores]

return titles.iloc[movie\_indices]

-----------------------------------------------------------

genre\_recommendations('movie name').head(10)