TED TALK RECOMMENDATION SYSTEM Machine Learning Project

WORK FLOW Data -> Data-Preprocessing -> Feature Extraction (Strings into Numericals) -> User Input -> Cosine Similarity -> Output

Feature Extraction: Find the Similarity Score (Similarity Confidence Score) between the tedtalks with each other

Cosine Similarity: to find the similarity between the vector (Each talk is converted into a kind of a vector) => Take userInput and compare with scores.

It also helps us to find similarity between different data points

```
%%capture
# Above %%capture is to suppress unwanted output.
import numpy as np
import pandas as pd
import difflib

import nltk
import string
import warnings
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity

nltk.download('stopwords')
warnings.filterwarnings('ignore')
```

```
df = pd.read_csv('ted_main.csv')
df.head(2)
# df.shape
```

	description	duration	main_speaker	name	speaker_occupation	tags	
0	Sir Ken Robinson makes an entertaining and pro...	1164	Ken Robinson	Ken Robinson: Do schools kill creativity?	Author/educator	['children', 'creativity', 'culture', 'dance',...	s crea

```
df.shape
```

```
(2550, 9)
```

```
# data = df.drop(['comments', 'event', 'film_date', 'languages','num_speaker','ratings', '
```

```
data=df
data = data.drop(['duration', 'views'], axis=1)
data.shape

# Selecting the relevant features - X and Y i.e feature Selection
selected_features = ['main_speaker', 'description', 'speaker_occupation', 'tags']
print(selected_features)
```

```
['main_speaker', 'description', 'speaker_occupation', 'tags']
```

```
data.isnull().sum()
# Replace Null values in selected_features, if any
for feature in selected_features:
    data[feature] = data[feature].fillna('') #fill empty boxes with null string

data.head(1)
```

	description	main_speaker	name	speaker_occupation	tags	t
0	Sir Ken Robinson makes an	Ken Robinson	Ken Robinson:	Author/educator	['children', 'creativity',	Do sc

```
# Combining Selected Featured
combined_features = data['main_speaker']+' '+data['description']+' '+data['speaker_occupat
print(combined_features)
```

```
0       Ken Robinson Sir Ken Robinson makes an enterta...
1       Al Gore With the same humor and humanity he ex...
2       David Pogue New York Times columnist David Pog...
3       Majora Carter In an emotionally charged talk, ...
4       Hans Rosling You've never seen data presented ...
                              ...
2545    Duarte Geraldino Between 2008 and 2016, the Un...
2546    Armando Azua-Bustos How can you study Mars wit...
2547    Radhika Nagpal Science fiction visions of the ...
2548    Theo E.J. Wilson In an unmissable talk about r...
2549    Karoliina Korppoo With more than half of the w...
Length: 2550, dtype: object
```

```
# Converting Textual Data into Numericals(Feature Vectors)

# Create an instance of it
vectorizer = TfidfVectorizer()

# Now we convert it into numericals
feature_vectors = vectorizer.fit_transform(combined_features)
print(feature_vectors)
```

```
  (0, 13893)    0.1463897967059414
  (0, 10110)    0.15021304703047728
  (0, 3423)     0.1604668980761989
  (0, 3330)     0.08007339290370094
  (0, 2459)	0.11713953419514435
  (0, 4371)	0.14588037058947498
  (0, 1103)	0.12208987331684158
  (0, 3232)	0.2182602972238796
  (0, 14698)	0.251213265875426
  (0, 14067)	0.11135520786490397
  (0, 11312)	0.166741732028129
  (0, 9656)	0.251213265875426
  (0, 14073)	0.05558940684495165
  (0, 13765)	0.1391043942406136
  (0, 4369)	0.22894890873345705
  (0, 3226)	0.15457865623865016
  (0, 5363)	0.061951420019824864
  (0, 2243)	0.1360198642486211
  (0, 9217)	0.14298891611285441
  (0, 10966)	0.2229735718780267
  (0, 678)	0.03524196974627706
  (0, 4629)	0.18712575436867
  (0, 652)	0.15943418646396665
  (0, 8427)	0.11835625450290148
  (0, 12768)	0.21260363942070637
  :	:
  (2549, 8175)	0.08768172220366721
  (2549, 14886)	0.07797611544461426
  (2549, 14883)	0.04736881780640794
  (2549, 7268)	0.04290537944766699
  (2549, 15252)	0.04346031219927643
  (2549, 11738)	0.11168576540089778
  (2549, 9778)	0.0599801726748963
  (2549, 9775)	0.04318075323751909
  (2549, 15502)	0.048855695139686635
  (2549, 8093)	0.061872195773012326
  (2549, 2547)	0.36365177782788033
  (2549, 14870)	0.2679318863872361
  (2549, 12967)	0.08799616790561401
  (2549, 9724)	0.04730082236261857
  (2549, 3723)	0.04962707128968222
  (2549, 13809)	0.08053778903722988
  (2549, 14244)	0.07333364136079845
  (2549, 5521)	0.043830208860036905
  (2549, 6868)	0.026701513428969943
  (2549, 14074)	0.04181939125972599
  (2549, 15442)	0.039887228050872386
  (2549, 3232)	0.06513418859504073
  (2549, 14067)	0.06646221234339154
  (2549, 678)	0.021034124237081696
  (2549, 652)	0.04757904438748618
```

```python
# Now we find cosine similarity
similarity = cosine_similarity(feature_vectors)

# Now it will go through all feature_vectors(numerical values of selected_features) and fi
# - how it works is it will first compare  first tedtalk with all the other tedtalk and th

print(similarity)
similarity.shape
```

```
[[1.         0.02272407 0.00377821 ... 0.01530766 0.01216128 0.02994413]
 [0.02272407 1.         0.02378765 ... 0.01267135 0.0344802  0.01339022]
 [0.00377821 0.02378765 1.         ... 0.01108547 0.01713572 0.03289073]
 ...
 [0.01530766 0.01267135 0.01108547 ... 1.         0.04072367 0.03681623]
 [0.01216128 0.0344802  0.01713572 ... 0.04072367 1.         0.05772986]
 [0.02994413 0.01339022 0.03289073 ... 0.03681623 0.05772986 1.        ]]
(2550, 2550)
```

```
# create a list of all movies names of dataset
list_of_all_titles = data['title'].tolist()
print(list_of_all_titles)
```

```
['Do schools kill creativity?', 'Averting the climate crisis', 'Simplicity sells', '(
```

```
# User Input Movie Name
tedtalk_name = input("Enter Movie Name: ")
```

```
Enter Movie Name: why we do what we do
```

```
# Finding close match for user input
find_close_match = difflib.get_close_matches(tedtalk_name , list_of_all_titles)
print(find_close_match)
```

```
['Why we do what we do', 'Why I do theater']
```

```
# Now take the first movie from find_close_match and compare the rest of movies for simila
close_match = find_close_match[0]
print(close_match)
```

```
Why we do what we do
```

```
# and now find the index of the movie
index = data[data.title == close_match]['index'].values[0]
print(index)
```

```
6
```

```
# getiing list of similar tedtalk based on index number
similarity_score = list(enumerate(similarity[index]))
# the tedtalks having simmilairyt score value which is equal to tedtalk index(why we do wh
print(similarity_score)
# len(similarity_score) = 2550
```

```
[(0, 0.004760346323887444), (1, 0.019004020293594524), (2, 0.014066979082299879), (3,
```

```
# find the highest similarity score - sort it now
sorted_similar_scores = sorted(similarity_score, key=lambda x:x[1], reverse=True)
```

```
print(sorted_similar_scores)
```

```
[(6, 1.0000000000000002), (685, 0.4427996820225012), (193, 0.3434023414632892), (1762
```

```
# User Input Movie Name
tedtalk_name = input("Enter Movie Name: ")
```

```
Enter Movie Name: WHY WE DO WHAT WE DO
```

```
# Display names

print("Suggested Ted Talks for "+tedtalk_name+" are: \n")
i = 1;
for tedtalk in sorted_similar_scores:
    index = tedtalk[0]
    title_from_index = data[data.index == index]['title'].values[0]
    if (i<11):
        print(i, ' - ', title_from_index)
        i+=1
```

```
Suggested Ted Talks for WHY WE DO WHAT WE DO are:

1  -  Letting go of God
2  -  It's time for "The Talk"
3  -  The Jill and Julia Show
4  -  I'm not your inspiration, thank you very much
5  -  A life of purpose
6  -  Four American characters
7  -  Why you think you're right -- even if you're wrong
8  -  Let's teach religion -- all religion -- in schools
9  -  Pay attention to nonviolence
10  -  How I'm working for change inside my church
```