Mini-Project Report

Department:	Mathematical and Computational Sciences
Specialization:	Computational and Data Science
Subject:	Big Data and Analytics
Topic:	Analysis of Co-Authorship network using Scopus databases
Course Instructor:	Dr. Pushpraj Shetty
Date:	02/05/2021

Team Number 3

- Gavali Deshabhakt Nagnath - 202CD005
- Mohammad Ahsan - 202CD016
- Shimpi Mayur Anil - 202CD027

Index

Chapter No.	***Chapter***	***Page No.***
I	Introduction & Methodology	3
II	Data pre-processing	8
III	Generating data required for further analysis	13
IV	Finding Answers	19
V	Conclusion & References	30
	Entire Python Code	31

Chapter – I

Introduction & Methodology

Introduction:

This project is about analyzing the co-authorship network database gathered from Scopus. Scopus is a website which provide us many ways of filtering the data. As per the question requirements we need to filter data accordingly on Scopus itself and using python program when we required to do so. So, we have to go through numerous preprocessing steps that are required to generate a data frame before actually initializing our analyzing steps.

Now let's introduce you with the Scopus layout through which we actually got our database. Scopus is affiliate with our institute central library where we need to mention the name of document category we searching for, in our case it was 'Artificial Intelligence', which prompt us to various other subcategories such as open access, year, author name, subject area, document type, source title, publication stage, keyword, affiliation, funding sponsor, country, source type and language. Each of these categories will allow us for further refining of the data set.

So finally after filtration on Scopus we did some further database refining using 'pandas' library of python like dropping of some unnecessary columns, renaming of the columns for easy access and order in which they appear, which ultimately give us this final structure of database which includes every author available on Scopus from 15 different countries i.e. 'Australia' , 'Canada', 'China', 'France', 'Germany', 'India', 'Iran', 'Italy', 'Japan', 'Netherland', 'South Korea', 'Spain', 'Taiwan', 'UK', 'US', with around 59k records and around 120k unique authors. Our database is created by merging of database of 15 countries into one single database with column names as follows-

'Authors'- This column contains the name of multiple authors separated by comma in one single string format.

'Title'- It contains information of the topic on which authors work upon, each with unique name.

'Year'- Give information of year of publication.

'Cited by'- It shows how many have referred the author's publication for their works.

'Country'- It represent the country from where the publication is done.

'Sponsors'- It shows whether or not the authors got the sponsorship for their work.

With this information in our complete database, we are in good position to analyze our database the problem statements given to us. We use 'python' and some of the libraries such as 'pandas', 'matplotlib', 'NumPy', '_pickel' etc. for better representation and analysis.

Methodology:

Now we will see the methodology that we have adopted to complete this project.

Step 1: Downloading database from scopus. We downloaded database separately for each country and again for funding details.

Step 2: Preprocessing of Main database:

In this process we removed some unnecessary attributes which are not required for our analysis and then we merged all the separate databases of countries in to one single CSV file.

Step 3: Generating Necessary Data Required for Further Analysis

In this process we have done operations to generate necessary data for further analysis i.e., to solve questions.

This step includes sub steps

1. Generating Authors list
2. Creating Python dictionary with Author name as key and his/her corresponding database as value
3. Creating list of Indian authors
4. Generating Foreign Authors list
5. Creating a dictionary with foreign author as key and number of papers published by him with Indian authors as value

Step 4: Finding answers of given questions.

Chapter - II

Downloading Database

As discussed earlier we used Scopus repository to download database. Scopus repository allows user to download database with maximum 2000 entries directly from their website and for databases with more than 2000 entries, they send a link to provided email and address and then one can download the database from that link. Also, in Scopus repository 20000 is the cap on entries that can be downloaded. This means that the user cannot download database with more than 20000 entries. Also, as the number of entries in the selected database increases, the unavailability of corresponding data attributes decreases. This means that if we select a large database for downloading then we might not receive some of the selected data attributes in the download file. In our case this attribute was 'funding details'. When we selected 'funding details' attribute for large database then the repository did not give corresponding data in the file. To overcome this difficulty, we downloaded databases corresponding to only funding details separately and then merged it to our main database.

We considered 'artificial intelligence' as a keyword and 'engineering' as filed area for downloading database. So, our query was as below (without double quotes)

"TITLE-ABS-KEY(Artificial Intelligence) AND (LIMIT-TO (DOCTYPE,"ar") OR LIMIT-TO (DOCTYPE,"ENGI") OR LIMIT-TO (DOCTYPE, "Artificial Intelligence"))"

Now after analyzing the questions that we're supposed to answer, we concluded that the data should be downloaded country-wise, meaning if we download data country wise then we will be able to add a country column in respective countries' database and merge those together (row-wise) to get main database.

In country-wise databases, we found that If two authors publish a paper together and they are from different countries then the same paper will be present in databases of both countries'. This makes our work a lot simpler, because then we can analyze the database 'as per' country also.

So, with all this in mind we used following steps to download the database.

1. Registration on scopus with institute email id.
2. Verifying email id and logging on to scopus
3. Selecting search parameter as keyword (in our case it was 'Artificial Intelligence')
4. Selecting document type as 'Article'
5. Selecting subject area as 'Engineering'
6. Then we got the query as mentioned above.

7. Using above query, we downloaded database country-wise. E.g. if we're to download database of India then the query would be something like this

```
TITLE-ABS-KEY(Artificial Intelligence) AND ( LIMIT-TO ( AFFILCOUNTRY,"India" ) ) AND ( LIMIT-TO ( DOCTYPE,"ar" ) ) AND ( LIMIT-TO ( SUBJAREA,"ENGI" ) )
```

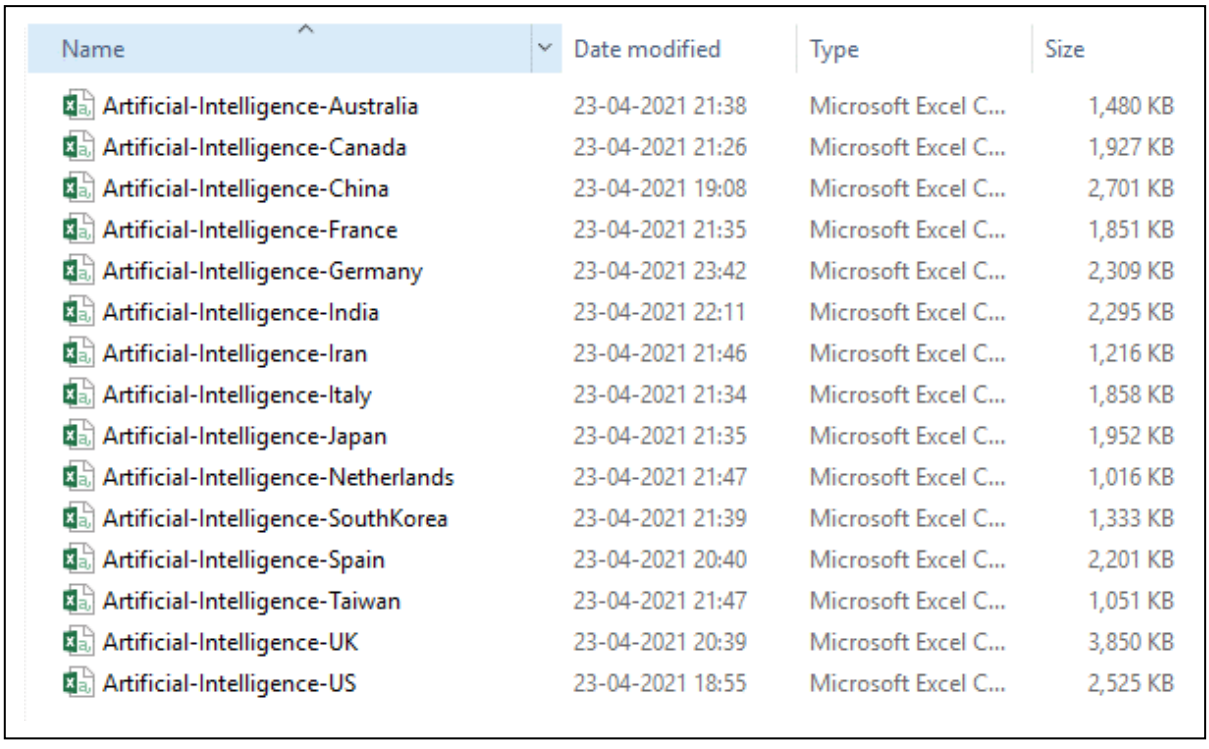
8. Changing country name and downloading corresponding data

We these steps we downloaded data country-wise but there was no 'funding details' attribute in any of the database. As we needed that attribute to find number of grants given to our filed 'Artificial Intelligence', we repeated same steps but used following query,

```
TITLE-ABS-KEY(Artificial Intelligence) AND ( LIMIT-TO ( DOCTYPE,"ar" ) OR LIMIT-TO ( DOCTYPE,"ENGI" ) OR LIMIT-TO ( DOCTYPE,"Artificial Intelligence" ) ) AND ( LIMIT-TO ( FUND-SPONSOR,"National Natural Science Foundation of China" ) OR LIMIT-TO ( FUND-SPONSOR,"National Institutes of Health" ) OR LIMIT-TO ( FUND-SPONSOR,"U.S. Department of Health and Human Services" ) OR LIMIT-TO ( FUND-SPONSOR,"National Science Foundation" ) OR LIMIT-TO ( FUND-SPONSOR,"European Commission" ) )
```

At the end of these steps we had following files at our hand,

a. Main database files



Name	Date modified	Type	Size
Artificial-Intelligence-Australia	23-04-2021 21:38	Microsoft Excel C...	1,480 KB
Artificial-Intelligence-Canada	23-04-2021 21:26	Microsoft Excel C...	1,927 KB
Artificial-Intelligence-China	23-04-2021 19:08	Microsoft Excel C...	2,701 KB
Artificial-Intelligence-France	23-04-2021 21:35	Microsoft Excel C...	1,851 KB
Artificial-Intelligence-Germany	23-04-2021 23:42	Microsoft Excel C...	2,309 KB
Artificial-Intelligence-India	23-04-2021 22:11	Microsoft Excel C...	2,295 KB
Artificial-Intelligence-Iran	23-04-2021 21:46	Microsoft Excel C...	1,216 KB
Artificial-Intelligence-Italy	23-04-2021 21:34	Microsoft Excel C...	1,858 KB
Artificial-Intelligence-Japan	23-04-2021 21:35	Microsoft Excel C...	1,952 KB
Artificial-Intelligence-Netherlands	23-04-2021 21:47	Microsoft Excel C...	1,016 KB
Artificial-Intelligence-SouthKorea	23-04-2021 21:39	Microsoft Excel C...	1,333 KB
Artificial-Intelligence-Spain	23-04-2021 20:40	Microsoft Excel C...	2,201 KB
Artificial-Intelligence-Taiwan	23-04-2021 21:47	Microsoft Excel C...	1,051 KB
Artificial-Intelligence-UK	23-04-2021 20:39	Microsoft Excel C...	3,850 KB
Artificial-Intelligence-US	23-04-2021 18:55	Microsoft Excel C...	2,525 KB

These files had following attributes,

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1	Authors	Author(s)	Title	Year	Source tit	Volume	Issue	Art. No.	Page start	Page end	Page coun	Cited by	DOI	Link	Documen	Publicatio	Open Acc	Source	EID	
2	Jayadeva,	55580544	Twin supp	2007	IEEE Trans	29	5		905	910		958	10.1109/T	https://w	Article	Final		Scopus	2-s2.0-34047225880	
3	Ravi, K., R	56715152	A survey c	2015	Knowledg	89			14	46		617	10.1016/j.	https://w	Article	Final		Scopus	2-s2.0-84944355103	
4	Varma, M.	24823171	A statistic	2009	IEEE Trans	31	11		2032	2047		479	10.1109/T	https://w	Article	Final		Scopus	2-s2.0-70349850950	
5	Nayak, P.(	12143137	A neuro-fi	2004	Journal of	291	01-Feb		52	66		466	10.1016/j.	https://w	Article	Final		Scopus	2-s2.0-1942490118	
6	Kale, A., S	36995883	Identifica	2004	IEEE Trans	13	9		1163	1173		445	10.1109/T	https://w	Article	Final		Scopus	2-s2.0-4344637549	
7	Singh, A.	3.53E+10	An artifici	2009	Applied Sc	9	2		625	631		373	10.1016/j.	https://w	Article	Final		Scopus	2-s2.0-58549117667	
8	Saha, S., R	55554063	AlgPred: F	2006	Nucleic Ac	34	WEB. SERV.	ISS.	W202	W209		348	10.1093/n	https://w	Article	Final	Open Acc	Scopus	2-s2.0-33747871140	
9	Aggarwal,	56493335	Electricity	2009	Internatio	31	1		13	22		321	10.1016/j.	https://w	Article	Final		Scopus	2-s2.0-57049127603	
10	Chakrabar	54790511	A functior	2006	Artificial I	19	2		113	132		317	10.1017/S	https://w	Article	Final		Scopus	2-s2.0-33744476708	
11	Bhandari,	25651606	Cuckoo se	2014	Expert Sys	41	7		3538	3560		313	10.1016/j.	https://w	Article	Final		Scopus	2-s2.0-84892371306	
12	Gupta, S.,	55581149	In Silico A	2013	PLoS ONE	8	9	e73957				298	10.1371/jc	https://w	Article	Final	Open Acc	Scopus	2-s2.0-84884141339	
13	Sun, Y., Ba	56346954	Majorizati	2017	IEEE Trans	65	3	7547360	794	816		295	10.1109/T	https://w	Article	Final		Scopus	2-s2.0-85002739348	
14	Rai, A., Up	57057017	A review c	2016	Tribology	96			289	306		287	10.1016/j.	https://w	Article	Final		Scopus	2-s2.0-84954319922	
15	Ahmad. S.	74019933	Analysis a	2004	Bioinform	20	4		477	486		277	10.1093/b	https://w	Article	Final	Open Acc	Scopus	2-s2.0-1542400269	

*We can see from above image that there is no 'funding details' column in downloaded database.

b. Database files with Funding Column containing funding column

Name	Date modified	Type	Size
AI-Australia	29-04-2021 00:11	Microsoft Excel C...	148 KB
AI-Canada	29-04-2021 00:09	Microsoft Excel C...	174 KB
AI-China	29-04-2021 00:04	Microsoft Excel C...	817 KB
AI-France	29-04-2021 00:11	Microsoft Excel C...	176 KB
AI-Germany	29-04-2021 00:07	Microsoft Excel C...	247 KB
AI-India	29-04-2021 00:01	Microsoft Excel C...	44 KB
AI-Iran	29-04-2021 00:13	Microsoft Excel C...	25 KB
AI-Italy	29-04-2021 00:10	Microsoft Excel C...	197 KB
AI-Japan	29-04-2021 00:10	Microsoft Excel C...	82 KB
AI-Netherlands	29-04-2021 00:14	Microsoft Excel C...	121 KB
AI-South_korea	29-04-2021 00:12	Microsoft Excel C...	76 KB
AI-Spain	29-04-2021 00:08	Microsoft Excel C...	375 KB
AI-Taiwan	29-04-2021 00:13	Microsoft Excel C...	49 KB
AI-UK	29-04-2021 00:07	Microsoft Excel C...	454 KB
AI-USA	29-04-2021 00:01	Microsoft Excel C...	788 KB

These files had following attributes

	A	B	C	D	E	F	G
1	Authors	Title	Year	Cited by	Link	Funding Details	Publication Stage
2	Varma M., Zisserman A.	A statistical approach to material classification using image patch exemplars	2009	479	http	University of Oxford	Final
3	Yang A.Y., Zhou Z., Balasubramanian A.G., Sastry S.S., Ma Y.	Fast l1-minimization algorithms for robust face recognition	2013	247	https://www.scopus.com/inward/re		Final
4	Kumar A., Zhang D.	Personal recognition using hand shape and texture	2006	216	http	National Natural Science Foundation of China	Final
5	Samuel O.W., Asogbon G.M., Sangaiah A.K., Fang P., Li G.	An integrated decision support system based on ANN and Fuzzy_AHP for heart failure risk pr	2017	158	http	Foundation of China	Final
6	Dou J., Yunus A.P., Tien Bui D., Merghadi A., Sahana M., Zhu Z	Assessment of advanced random forest and decision tree algorithms for modeling rainfall-in	2019	123	http	National Natural Science Foun	Final
7	Mitchell T., Cohen W., Hruschka E., Talukdar P., Yang B., Bette	Never-ending learning	2018	120	http		Final
8	Wang G.-G., Deb S., Gandomi A.H., Alavi A.H.	Opposition-based krill herd algorithm with Cauchy mutation and position clamping	2016	97	http	National Natural Science Foun	Final
9	Zhu F., Bosch M., Khanna N., Boushey C.J., Delp E.J.	Multiple hypotheses image segmentation and classification with application to dietary asses	2015	94	http		Final

*We can see from above image that there is a 'funding details' column in downloaded database.

Chapter – III

Data Pre-processing

A. Preprocessing of Main database

As discussed in earlier chapter we had downloaded database separately for each country and again for funding details. Also, the separated databases had some unnecessary attributes. So, these databases cannot be used directly for data analysis. That's why we have to do preprocessing on those.

For data-preprocessing and also data analysis, we used python programming language. Python has many libraries which makes data analysis much easier. We imported the database of each countries using 'read_csv()' function of pandas library and did pre-processing on it. Following is the block of code that we used to do so.

```
base_dir = '/mnt/General_Stuff/Study Stuff/Documents/CDS/Sem-II/BigDataAnalytics/BigData-Programs/Mini-Project/Data'
```

This base_dir variable contains a string which path to database directory.

```
dir_lst = []
for data_file in os.listdir(base_dir):
    dir_lst.append(os.path.join(base_dir,data_file))
```

Using this block of code merged filename.extension in database directory with base_dir variable and stored the resultant string into a python list.

e.g. merging 'Artificial-Intelligence-India.csv' with base_dir variable gives

"/mnt/General_Stuff/Study Stuff/Documents/CDS/Sem-II/BigDataAnalytics/BigData-Programs/Mini-Project/Data/Artificial-Intelligence-India.csv"

this string. We are doing this for every file in database directory and appending it to python list (i.e. dir_lst). Thus, the python list will contain strings which are paths of databases of each country.

After getting python to databases we imported those using pandas's read_csv() function and removed un-necessary columns from them using drop() function. Then we also added a country column in respective database containing corresponding country name.

We removed following columns/attributes from database,

'Author(s) ID', 'Source title', 'Volume', 'Issue', 'Art. No.', 'Page start', 'Page end', 'Page count', 'DOI', 'Link', 'Document Type', 'Publication Stage', 'Open Access', 'Source', 'EID'

Also, we filled all empty cells in database with 0.

For doing all this we have used following block of code.

```
country_lst =
['Australia','Canada','China','France','Germany','India','Iran','Italy','Japan','Netherlands','South
Korea','Spain','Taiwan','United Kingdom', 'United States']

df_lst = []
for data_file, country in zip(dir_lst, country_lst):
    print(country)
    df_tmp = pd.read_csv(data_file)
    df_tmp = df_tmp.drop(['Author(s) ID' ,'Source title','Volume', 'Issue', 'Art. No.', 'Page start', 'Page
end', 'Page count', 'DOI', 'Link', 'Document Type', 'Publication Stage','Open Access', 'Source',
'EID'],axis='columns')

    df_tmp = df_tmp.fillna(0)
    df_tmp['Country'] = country
    df_lst.append(df_tmp)
```

As explained above this code does following things,

1. Takes a database from database directory
2. Removes un-necessary attributes from it
3. Fills all empty cells with 0
4. Adds country column with corresponding country name
5. Appends the resulting database to python list.

At the end of this block we will have a python list containing filtered database of all countries.

```
[ ]  1  print(country_lst[0])
     2  df_lst[0].head()
```

Australia

	Authors	Title	Year	Cited by	Country
0	Soares, J.V.B., Leandro, J.J.G., Cesar Jr., R....	Retinal vessel segmentation using the 2-D Gabo...	2006	1083.0	Australia
1	Scarselli, F., Gori, M., Tsoi, A.C., Hagenbuch...	The graph neural network model	2009	1031.0	Australia
2	Karantonis, D.M., Narayanan, M.R., Mathie, M.,...	Implementation of a real-time human movement c...	2006	908.0	Australia
3	Mirjalili, S.	Dragonfly algorithm: a new meta-heuristic opti...	2016	865.0	Australia
4	Naseem, I., Togneri, R., Bennamoun, M.	Linear regression for face recognition	2010	768.0	Australia

*here we can see the df_lst python list contains database of each country at different indices.

But we cannot use this python list for data analysis directly. We have to consolidate data in a single pandas dataframe. For that we used concat() function of pandas.

```
df = pd.concat(df_lst)
```

The ‘df’ variable now contains database of all countries in consolidated form. We can see that in below image.

```
[ ]  1  df.shape
```

```
(67694, 5)
```

```
[ ]  1  df.head()
```

	Authors	Title	Year	Cited by	Country
0	Soares, J.V.B., Leandro, J.J.G., Cesar Jr., R....	Retinal vessel segmentation using the 2-D Gabo...	2006	1083.0	Australia
1	Scarselli, F., Gori, M., Tsoi, A.C., Hagenbuch...	The graph neural network model	2009	1031.0	Australia
2	Karantonis, D.M., Narayanan, M.R., Mathie, M.,...	Implementation of a real-time human movement c...	2006	908.0	Australia
3	Mirjalili, S.	Dragonfly algorithm: a new meta-heuristic opti...	2016	865.0	Australia
4	Naseem, I., Togneri, R., Bennamoun, M.	Linear regression for face recognition	2010	768.0	Australia

So, our final database has **total 67694 entries**. It contains data of **15-countries**. The names of those countries are as follows:

1. Australia
2. Canada
3. China
4. France
5. Germany
6. India
7. Iran
8. Italy
9. Japan
10. Netherlands
11. South Korea
12. Spain
13. Taiwan
14. United Kingdom
15. United States

But this database does not have ‘Funding Details’ attribute in it. So, in order to add this attribute to our database, we used following procedure.

1. Changed base_dir variable to path of database with sponsorship details directory i.e.

```
base_dir = '/mnt/General_Stuff/Study Stuff/Documents/CDS/Sem-II/BigDataAnalytics/BigData-Programs/Mini-Project/Data_with_sponsor_details'
```

2. Used same data pipeline as above for filtering data.
3. At the end of second step we obtained a pandas dataframe with sponsorship column in it.

```
[ ]  1  df_addon.head()
```

	Authors	Title	Year	Cited by	Funding Details	Country
0	Tao F., Qi Q., Liu A., Kusiak A.	Data-driven smart manufacturing	2018	375.0	National Natural Science Foundation of China\n...	Australia
1	Zhang K., Gao X., Tao D., Li X.	Single image super-resolution with non-local m...	2012	362.0	National Natural Science Foundation of China\n...	Australia
2	Kristan M., Matas J., Leonardis A., Vojir T., ...	A Novel Performance Evaluation Methodology for...	2016	264.0	Seventh Framework Programme	Australia
3	Ding C., Choi J., Tao D., Davis L.S.	Multi-Directional Multi-Level Dual-Cross Patte...	2016	243.0	National Science Foundation\n \nAustralian Rese...	Australia
4	Celebi M.E., Kingravi H.A., Iyatomi H., Asland...	Border detection in dermoscopy images using st...	2008	241.0	National Cancer Institute	Australia

4. Then we added a 'Funding Details' column in our main database and initialized it to zero using following code.

```
df['Funding_Details'] = 0
```

5. After adding funding details column, we compared two databases by title and changed funding details accordingly.

```
titles_old = list(df.Title)
titles_new = list(df_addon.Title)
count = 1
for i in range(len(titles_new)):
    for j in range(len(titles_old)):
        if titles_new[i]==titles_old[j]:
            df.iloc[j,-1] = df_addon.iloc[i,-2]
    print(f'{count}\t{titles_new[i]}')
    count += 1
```

This block of code creates 2- python lists containing titles from main and additional database and compares those titles one by one. If we find a match then we just replacing value at funding details column in main database with value at funding details column in additional database corresponding to matched title

After completing all the above steps we generated a dataframe which suits our need. This dataframe was then stored in csv file using to_csv() function of panadas library, so that it can be used for further ananlysis easily.

```
df.to_csv('Complete_database.csv')
```

The resulting csv file had following attributes in it.

	A	B	C	D	E	F	G
1		Authors	Title	Year	Cited by	Country	Funding_Details
2	0	Soares, J.V.B., Leandro, J.J.G., Cesar Jr., R.M., Jel	Retinal vessel segmentation using the 2-D Gabor wavelet and supervi	2006	1083	Australia	0
3	1	Scarselli, F., Gori, M., Tsoi, A.C., Hagenbuchner,	The graph neural network model	2009	1031	Australia	0
4	2	Karantonis, D.M., Narayanan, M.R., Mathie, M., I	Implementation of a real-time human movement classifier using a tri	2006	908	Australia	0
5	3	Mirjalili, S.	Dragonfly algorithm: a new meta-heuristic optimization technique fo	2016	865	Australia	0
6	4	Naseem, I., Togneri, R., Bennamoun, M.	Linear regression for face recognition	2010	768	Australia	0
7	5	Geng, X., Zhou, Z.-H., Smith-Miles, K.	Automatic age estimation based on facial aging patterns	2007	675	Australia	0
8	6	Chen, Y., Zhao, X., Jia, X.	Spectral-Spatial Classification of Hyperspectral Data Based on Deep Be	2015	599	Australia	0
9	7	Phung, S.L., Bouzerdoum, A., Chai, D.	Skin segmentation using color pixel classification: Analysis and compa	2005	594	Australia	0
10	8	Li, X., Yao, X.	Cooperatively coevolving particle swarms for large scale optimization	2012	508	Australia	0
11	9	Dissanayake, S.D., Armstrong, J.	Comparison of ACO-OFDM, DCO-OFDM and ADO-OFDM in IM/DD syst	2013	427	Australia	0
12	10	Ong, Y.-S., Lim, M.-H., Zhu, N., Wong, K.-W.	Classification of adaptive memetic algorithms: A comparative study	2006	425	Australia	0
13	11	Mian, A.S., Bennamoun, M., Owens, R.	An efficient multimodal 2D-3D hybrid approach to automatic face rec	2007	403	Australia	0
14	12	Tao, F., Qi, Q., Liu, A., Kusiak, A.	Data-driven smart manufacturing	2018	373	Australia	National Natural Science Foundation of
15	13	Mian, A.S., Bennamoun, M., Owens, R.	Three-dimensional model-based object recognition and segmentatio	2006	364	Australia	0
16	14	Zhang, K., Gao, X., Tao, D., Li, X.	Single image super-resolution with non-local means and steering ker	2012	362	Australia	National Natural Science Foundation of
17	15	Hong, C., Yu, J., Wan, J., Tao, D., Wang, M.	Multimodal Deep Autoencoder for Human Pose Recovery	2015	354	Australia	0
18	16	Tournier, J.-D., Yeh, C.-H., Calamante, F., Cho, K.	Resolving crossing fibres using constrained spherical deconvolution:	2008	349	Australia	0
19	17	Zhang, D., Islam, M.M., Lu, G.	A review on automatic image annotation techniques	2012	332	Australia	0
20	18	Lu, J., Behbood, V., Hao, P., Zuo, H., Xue, S., Zhai	Transfer learning using computational intelligence: A survey	2015	329	Australia	0
21	19	LÃª Cao, K.-A., Boitard, S., Besse, P.	Sparse PLS discriminant analysis: Biologically relevant feature selecti	2011	329	Australia	0
22	20	Shawe-Taylor, J., Bartlett, P.L., Williamson, R.C.,	Structural risk minimization over data-dependent hierarchies	1998	310	Australia	0
23	21	Yao, X.	A review of evolutionary artificial neural networks	1993	309	Australia	0

Chapter – III

Generating Data Required for Further Analysis

Now we'll discuss about the files that we generated during further data-preprocessing phase. We imported the previously created csv file and stored it in panadas data-frame. One thing to note here is that, if two authors from different countries work together on a same research paper then that paper will be present corresponding to both countries. Thus, we need to tackle with these duplicate entries. For this we used drop_duplicate() function of pandas. Here's code of that,

```
[ ]  1  df_without_countries = df.drop('Country',axis='columns')

[ ]  1  df_without_countries = df_without_countries.drop_duplicates(subset=['Title'],keep='first')

[ ]  1  df_without_countries.shape

     (59215, 5)
```

Here we can see that the unique entries are 59,215 instead of 67,694. Now we'll use this dataframe (i.e. df_without_countries) for further analysis.

Now we'll generate files/data required for further analysis.

1. **Generating Authors list**

   In database that we have almost all research papers are published by authors working together. So, authors column corresponding to each contains names of all the others who worked on that particular paper. But for data analysis we need names of individual authors. So, here we we'll generate those names and store it in python list and in turn in a text file. For this we have used following code

   a. First, we'll extract collective authors from database

```
[ ]  1  authors_lst = list(df_without_countries.loc[:,'Authors'].values)

[ ]  1  print(authors_lst[0])

     Soares, J.V.B., Leandro, J.J.G., Cesar Jr., R.M., Jelinek, H.F., Cree, M.J.

[ ]  1  len(authors_lst)

     59215
```

   b. As we can see in output of above code, the authors_lst holds authors names with end name and initial separted by commas. We'll now generate individual authors name from authors_lst python list.

```
set_authors = ['Jayadeva', 'Khemchandani, R.', 'Chandra, S.']
for i in range(len(authors_lst)):
    authors_sub_lst = authors_lst[i].split(',')
    authors_sub_lst_mod = []
    if authors_sub_lst == 'Jayadeva, Khemchandani, R., Chandra, S.'.split(','):
        continue
    for j in range(0,len(authors_sub_lst)-1,2):
        authors_sub_lst_mod.append(authors_sub_lst[j].strip()+','+authors_sub_lst[j+1])

    for author in authors_sub_lst_mod:
        if(author not in set_authors):
            set_authors.append(author)
            print(f'{i}\t{author}')
```

The above code works as follow:

1. Create a python list with name set_authors
2. Go through the authors_lst one by one, pick one string from it.
3. Split the string at ‘,’ and then merge the split string with increment of 2. (because end name and initials of author are also separated by ‘,’)
4. If the merged names are not in set_authors then append it to set_authors (this avoids repetitions of author names).

At the end of this code we’d have a python list containing names of all authors in our database.

As creating this list takes lot of time, so to avoid repetitive computation of this list we stored it in a text file using following code,

```
with open('Authors_list.txt','w') as filehandle:
    filehandle.writelines("%s\n" % author for author in set_authors)
filehandle.close()
```

Also, we can re-use it whenever it is required in further analysis.
This file can be read using following code,

```
set_authors = []
with open('Authors_list.txt','r') as filehandle:
    filecontents = filehandle.readlines()

    for line in filecontents:
        # remove linebreak which is the last character of the string
        author = line[:-1]

        # add item to the list
        set_authors.append(author)
```

Note:

Here we initialized the set_authors with "'Jayadeva', 'Khemchandani, R.', 'Chandra, S.'" because this entry does not follow our assumption that the end name and initials are separated by ',' (Mr. Jayadeva did not provide his initials)

2. Creating Python dictionary with Author name as key and his/her corresponding database as value

For doing this we used following code,

```
dct_author_database = {}
count=0
for author in set_authors:
    print(f'{count}\t{author}')
    df_auth = pd.DataFrame(columns = ['Authors', 'Title', 'Year', 'Cited_by','Funding Details'] )
    filt= df_without_countries['Authors'].str.contains(author, na=False)
    df_auth= df_without_countries.loc[filt,'Authors':].reset_index(drop=True)
    dct_author_database[author] = df_auth
    count += 1
```

This code does following things:

1. Creates a python dictionary with name dct_author_database
2. Go through each author in set_authors list
3. Create an empty dataframe say df_tmp
4. Go through df_wihtout_countries dataframe and append all rows containing authors name (picked at step 2) to the empty dataframe ie. df_tmp.
5. And then store author name and df_tmp as key value pair in dct_author_database dictionary.

At the end of this code we would have database of each other separated by author name. Storing it in dictionary makes it easy to fetch.

This thing can also be done using following code.

```
dct_author_database = {}
count = 0
for author in set_authors:
    print(f'{count}\t{author}')
    df_auth = pd.DataFrame(columns = ['Authors', 'Title', 'Year', 'Cited_by'] )

    for authors in authors_lst:
        if author in authors:
            df_tmp = df_without_countries[df_without_countries.Authors==authors]
            df_auth = df_auth.append(df_tmp,ignore_index=True)
    dct_author_database[author] = df_auth
    count += 1
```

Generating this dictionary takes around 50-60 minutes. To avoid this repetitive computation, we'll store this library to a txt file. For that we'll use 'pickle' library.
Here's code for that,

```
with open('Author_database_dictonary.txt','wb') as file:
   file.write(pickle.dumps(dct_author_database))
file.close()
```

To read this file and get back our dictionary we can use following code,

```
dct_author_database_from_file = {}
with open('Author_database_dictonary.txt','rb') as file:
   dct_author_database_from_file = pickle.load(file)
file.close()
```

With this code we can load the dictionary at any point in further data analysis, without much computation and save our time.

3. Creating list of Indian authors

```
authors_from_ind_database = list(df[df.Country=='India']['Authors'].unique())

set_of_authors_from_indian_database = ['Jayadeva', 'Khemchandani, R.', 'Chandra, S.']
count = 0
for authors in authors_from_ind_database:
   authors_sub_lst = authors.split(',')
   if authors_sub_lst == 'Jayadeva, Khemchandani, R., Chandra, S.'.split(','):
      continue

   authors_sub_lst_mod = []
   for i in range(0,len(authors_sub_lst)-1,2):
         authors_sub_lst_mod.append(authors_sub_lst[i].strip()+','+authors_sub_lst[i+1])

   for author in authors_sub_lst_mod:
      if(author not in set_of_authors_from_indian_database):
         set_of_authors_from_indian_database.append(author)
   print(f'{count}\t{authors}')
   count += 1
```

This code follows same algorithm as code used for generating set_authors list in 1st bullet.

We then saved this file to text file using following code,

```
with open('India_authors_list.txt','w') as filehandle:

      filehandle.writelines("%s\n" % author for author in set_of_authors_from_indian_database)

filehandle.close()
```

But, as we know that this list might contain names of some foreign authors, that's why we checked list manually and removed foreign authors names from it. (it took time because the list generated had around 11K names in it, so we divided it among our team)

After removing foreign authors names from the text file, we can load it using following command.

```
set_of_indian_authors_from_file = []
with open('Indian_authors_list.txt','r') as filehandle:
    filecontents = filehandle.readlines()
    for line in filecontents:
        # remove linebreak which is the last character of the string
        author = line[:-1]
        # add item to the list
        set_of_indian_authors_from_file.append(author)
```

4. Generating Foreign Authors list

As we have a Indian authors list and a list of all authors it's fairly easy to generate foreign authors list from it using following code.

```
set_of_foreign_authors = []
count = 0
for author in set_authors:
    if author not in set_of_indian_authors_from_file:
        set_of_foreign_authors.append(author)
        print(f'{count}\t{author}')
        count += 1
```

Storing it in a text file:

```
with open('Foreign_authors_list.txt','w') as filehandle:
        filehandle.writelines("%s\n" % author for author in set_of_foreign_authors_from_file)
filehandle.close()
```

Reading from Text file:

```
set_of_foreign_authors_from_file = []
with open('Foreign_authors_list.txt','r') as filehandle:
    filecontents = filehandle.readlines()
    for line in filecontents:
        # remove linebreak which is the last character of the string
        author = line[:-1]
        # add item to the list
        set_of_foreign_authors_from_file.append(author)
```

5. **Creating a dictionary with foreign author as key and number of papers published by him with Indian authors as value**

We used this dictionary to find out author with highest co-authorship with Indian author. The dictionary can be generated using following code,

```
dct_foreign_author_coauth_count = {}
for foreign_author in set_of_foreign_authors_from_file:
    dct_foreign_author_coauth_count[foreign_author] = 0

count = 0
for foreign_author in set_of_foreign_authors_from_file:
    row_foreign_auth_list = dct_author_database_from_file[foreign_author].iloc[:,0].values
    for indian_author in set_of_indian_authors_from_file:
        for authors in row_foreign_auth_list:
            if indian_author in authors:
                dct_foreign_author_coauth_count[foreign_author] += 1
    print(f'{count}    {foreign_author}       {dct_foreign_author_coauth_count[foreign_author]}')
    count += 1
```

Storing it in file:

```
with open('Foreign_auth_and_their_publication_count_with_india_authors_dct.txt','wb') as file:
    file.write(pickle.dumps(dct_foreign_author_coauth_count))
file.close()
```

Reading it from file:

```
dct_foreign_author_coauth_count_from_file = {}
with open('Foreign_auth_and_their_publication_count_with_india_authors_dct.txt','rb') as file:
    dct_foreign_author_coauth_count_from_file = pickle.load(file)
file.close()
```

These are all the files that we'll require in finding answers to our questions. In next chapter we'll use these files for further analysis.

Chapter – IV

Finding Answers

In this chapter we've found answers of all the questions asked in our min-project pdf. For this we've used all the previously generated files. First, we created a function to sort dictionaries. This function is used in lot of next steps. Its definition is as follows,

```
def sort_dict(dct,parameter,order='Ascending'):
    if parameter=='key':
        if(order=='reverse'):
            sorted_tuples = sorted(dct.items(),key=lambda item:item[0],reverse=True)
            return {k:v for k,v in sorted_tuples}
        else:
            sorted_tuples = sorted(dct.items(),key=lambda item:item[0])
            return {k:v for k,v in sorted_tuples}
    else:
        if(order=='reverse'):
            sorted_tuples = sorted(dct.items(),key=lambda item:item[1],reverse=True)
            return {k:v for k,v in sorted_tuples}
        else:
            sorted_tuples = sorted(dct.items(),key=lambda item:item[1])
            return {k:v for k,v in sorted_tuples}
```

This function takes 3 arguments as input,

1. Python dictionary
2. Parameter for sorting (either key or value)
3. Order of sorting (by default its ascending)

This function returns a sorted python dictionary as per provided parameters.

Now coming to questions,

a. Highest cited author and his h-index (from the world)

For finding this we've used following code,

```
author_with_highest_citations = ""
max_citations = 0
for author in set_authors:
  cites = dct_author_database[author]['Cited_by'].sum()
  if max_citations<cites:
    author_with_highest_citations = author
    max_citations = cites

df_of_highest_cited_author = dct_author_database[author_with_highest_citations]
rows,cols = df_of_highest_cited_author.shape

# h_index = min(rows,df_of_highest_cited_author['Cited_by'].min())

avg_citations_of_author_with_highest_citations = max_citations/rows
h_index = min(rows,avg_citations_of_author_with_highest_citations)
```

Its output is as follows,

```
[ ]  1  print(f'Max cited author    = {author_with_highest_citations}')
     2  print(f'Total cited by      = {max_citations}')
     3  print(f'His h-index         = {h_index}')

Max cited author    = Hassabis, D.
Total cited by      = 17466.0
His h-index         = 13
```

Here we can see that highest cited author is **'Hassabis, D.'** and his total citations and h-index are as in output.

b. Highest publication author

For finding this we've used following code,

```
author_with_highest_publication = ""
max_publication_count = 0
for author in set_authors:
  rows, columns = dct_author_database[author].shape
  if rows>max_publication_count:
    max_publication_count=rows
    author_with_highest_publication = author
    # print(f'{author}              \t{rows}')
```

Its output is a below,

```
[ ]  1  print(f'Author\t\t= {author_with_highest_publication}\nPublications\t= {max_publication_count}')
     2  dct_author_database[author_with_highest_publication]
```

```
Author          = Wang, Y.
Publications    = 439
```

	Authors	Title	Year	Cited_by	Funding Details
0	Wang, Y., Hu, J., Phillips, D.	A fingerprint orientation model based on 2D fo...	2007	233.0	0
1	Chen, C., Wang, Y., Zhang, J., Xiang, Y., Zhou...	Statistical Features-Based Real-Time Detection...	2017	63.0	Australian Research Council\n\nNational Natura...
2	Chen, G., Wang, Y., Li, S., Cao, W., Ren, H., ...	Spatiotemporal patterns of PM<inf>10</inf> con...	2018	54.0	0
3	Yin, J., Wang, Y., Hu, J.	A new dimensionality reduction algorithm for h...	2012	53.0	National Natural Science Foundation of China\n...
4	Dwivedi, Y.K., Ismagilova, E., Hughes, D.L., C...	Setting the future of digital and social media...	2020	47.0	0
...	...	...	...	...	...
434	Wang, Y., Chen, T., Zeng, D.	Support vector hazards machine: A counting pro...	2016	5.0	0
435	Cheng, Y., Wang, Y., Camps, O., Sznaier, M.	The Interplay Between Big Data and Sparsity in...	2015	2.0	0
436	Usher, J.M., Wang, Y.C.	Judging the value of additional information on...	2000	2.0	0
437	Wang, Y., Wu, S., Yu, B.	Unique Sharp Local Minimum in `<inf>1</inf>-mi...	2020	1.0	0
438	Wang, Y.-P., Ragib, H., Huang, C.-M.	A wavelet approach for the identification of a...	2007	1.0	0

439 rows × 5 columns

From output we can see that '**Wang Y.'** has highest publications **(total 439)**.

c. Highest cited authors avg. citations, and the country name

We already found highest cited author in first question. Now we can simply make use of our dct_author_database dictionary to find out his average citations and for finding country we can simply use google.

```
[ ]  1  rows, cols = dct_author_database[author_with_highest_citations].shape
     2
     3  avg_citations_of_author_with_highest_citations = max_citations/rows
     4  print(f'Highest Cited Author \t= {author_with_highest_citations}')
     5  print(f'His Average Citations \t= {avg_citations_of_author_with_highest_citations}')
     6  print(f'His Total Publications \t= {rows}')
```

```
Highest Cited Author    = Hassabis, D.
His Average Citations   = 1343.5384615384614
His Total Publications  = 13
```

From output we can see that **the total publications of our highest cited author are 13** and his average citations are **around 1344.**
And from google we found that the author is from **United Kingdom.**

d. Total number of publications of the highest cited author

Answered in previous question itself.

```
[ ]  1  print(f'Highest Cited Author \t= {author_with_highest_citations}')
     2  print(f'Total Publications \t= {rows}')
     3  dct_author_database[author_with_highest_citations]
```

```
Highest Cited Author    = Hassabis, D.
Total Publications      = 13
```

e. Total publication in year

Our database contains information from 1964 to 2021. The total publication year-wise can be found using following code.

```
year_lst = sorted(list(df['Year'].unique()))
country_lst = list(df['Country'].unique())

df_without_duplicates = df.drop_duplicates(subset=['Authors','Title'],keep='first')

dct_df_per_year_publications = {}
for year in year_lst:
   dct_df_per_year_publications[year], cols =
df_without_duplicates[df_without_duplicates.Year==year].shape

dct_df_per_year_publications = sort_dict(dct_df_per_year_publications,'Values','reverse')
```

Its output is as below,

```
{2020: 6222,
 2019: 4688,
 2018: 4122,
 2016: 3341,
 2014: 3159,
 2015: 3117,
 2017: 3049,
 2021: 2688,
 2013: 2277,
 2004: 2191,
 2012: 2010,
 2008: 1933,
 2009: 1565,
 2011: 1559,
 2010: 1542,
 2006: 1464,
 2007: 1401,
 2003: 1357,
 2005: 1346,
 2000: 793,
 1989: 734,
 1999: 730,
 2001: 726,
 1997: 718,
 1996: 713,
 1994: 703,
 2002: 702,
 1998: 697,
 1995: 601,
 1988: 531,
 1993: 485,
 1990: 457,
 1991: 394,
 1992: 357,
 1987: 351,
 1986: 165,
 1985: 117,
 1984: 70,
 1983: 25,
 1982: 23,
 1977: 17,
 1980: 13,
 1973: 12,
 1978: 12,
 1979: 12,
 1981: 11,
 1974: 8,
 1975: 8,
 1976: 8,
 1971: 7,
 1972: 6,
 1970: 3,
 1962: 1,
 1963: 1,
 1964: 1,
 1965: 1,
 1968: 1,
 1969: 1}
```

f. Total citation per year

We can use same algorithm as above to find total cites per year,

```
dct_citations_per_year = {}
for year in year_lst:
   dct_citations_per_year[year]=df_without_duplicates[df_without_duplicates.Year==year]['Cited_by
   '].sum()
   dct_citations_per_year = sort_dict(dct_citations_per_year,'Values','reverse')
dct_citations_per_year
```

Its output is as below,

```
{2008: 86548.0,
 2016: 85426.0,
 2015: 81343.0,
 2005: 78325.0,
 2004: 75544.0,
 2007: 73896.0,
 2006: 73652.0,
 2014: 69065.0,
 2009: 68433.0,
 2018: 66871.0,
 2013: 63245.0,
 2017: 61390.0,
 2010: 61118.0,
 2012: 56435.0,
 2011: 55280.0,
 2019: 42281.0,
 2000: 36085.0,
 2003: 32783.0,
 1999: 28128.0,
 1998: 26193.0,
 2002: 25418.0,
 2001: 25176.0,
 1997: 21638.0,
 2020: 21386.0,
 1995: 18774.0,
 1994: 18694.0,
 1996: 16943.0,
 1989: 12072.0,
 1992: 10540.0,
 1990: 9800.0,
 1993: 8807.0,
 1991: 8453.0,
 1988: 8385.0,
 1987: 5439.0,
 1986: 3652.0,
 1980: 2616.0,
 1985: 1930.0,
 1977: 1644.0,
 2021: 1568.0,
 1979: 997.0,
 1984: 812.0,
 1971: 446.0,
 1973: 256.0,
 1978: 252.0,
 1976: 215.0,
 1983: 208.0,
 1982: 202.0,
 1975: 186.0,
 1970: 83.0,
 1981: 76.0,
 1972: 49.0,
 1974: 33.0,
 1963: 17.0,
 1962: 16.0,
 1969: 4.0,
 1964: 2.0,
 1965: 2.0,
 1968: 0.0}
```

g. Author(country) having highest co-authorship with Indian authors

Its code as below,

```
mx_pubs_with_indian_authors = 0
foreign_auth_corr_to_mx_pubs_with_indian_authors = ""
for author in set_of_foreign_authors:
  pubs = dct_foreign_author_coauth_count[author]
  if mx_pubs_with_indian_authors < pubs:
    mx_pubs_with_indian_authors = pubs
    foreign_auth_corr_to_mx_pubs_with_indian_authors = author
```

Its output is as below,

```
[ ]  1  print(f'Foreign author with Highest Co-authorship with Indian Authors =  {foreign_auth_corr_to_mx_pubs_with_indian_authors}')
     2  print(mx_pubs_with_indian_authors)

Foreign author with Highest Co-authorship with Indian Authors =  Nicolaides, A.
127
```

It turns out that author **Nicolaides, A.** has highest co-authorship with Indian authors and he is from **"Vascular Screening and Diagnostic Centre, University of Nicosia, Nicosia, Cyprus"** (from google)

h. Highest cited author from India and the university

For this we make use of our previously created Indian authors list and dct_author_database dictionary. Its code goes as below,

```
max_cites_of_indian_author = 0
highest_cited_indian_author = ""

for author in set_of_indian_authors:
  cites = dct_author_database[author]['Cited_by'].sum()
  if max_cites_of_indian_author < cites:
    max_cites_of_indian_author = cites
    highest_cited_indian_author = author
```

Its output is as follow,

```
[ ]  1  print(f'Highest Cited Author from India = {highest_cited_indian_author}')
     2  print(f'His Total Citations = {max_cites_of_indian_author}')
     3  dct_author_database[highest_cited_indian_author].head()

Highest Cited Author from India = Raghava, G.P.
His Total Citations = 3132.0
```

	Authors	Title	Year	Cited_by	Funding Details
0	Saha, S., Raghava, G.P.S.	AlgPred: Prediction of allergenic proteins and...	2006	348.0	0
1	Gupta, S., Kapoor, P., Chaudhary, K., Gautam, ...	In Silico Approach for Predicting Toxicity of ...	2013	298.0	0
2	Bhasin, M., Raghava, G.P.S.	ESLpred: SVM-based method for subcellular loca...	2004	252.0	0
3	Bhasin, M., Raghava, G.P.S.	Prediction of CTL epitopes using QM, SVM and A...	2004	239.0	0
4	Kumar, M., Gromiha, M.M., Raghava, G.P.S.	Prediction of RNA binding sites in a protein u...	2008	202.0	0
5	Bhasin, M., Garg, A., Raghava, G.P.S.	PSLpred: Prediction of subcellular localizatio...	2005	161.0	0

i. Comparative year wise article publication analysis of India, China and USA

For this question we first separated each countries database from our main database using following code.

```
df_india = df[df.Country=='India'].copy().reset_index(drop=True)
df_china = df[df.Country=='China'].copy().reset_index(drop=True)
df_usa = df[df.Country=='United States'].copy().reset_index(drop=True)
```

Then we created dictionary with country name as key and publication count as value. For this we use following code.

Then we made x and y variables as follow,

Now coming to comparison part, we used line and bar graphs for comparison.

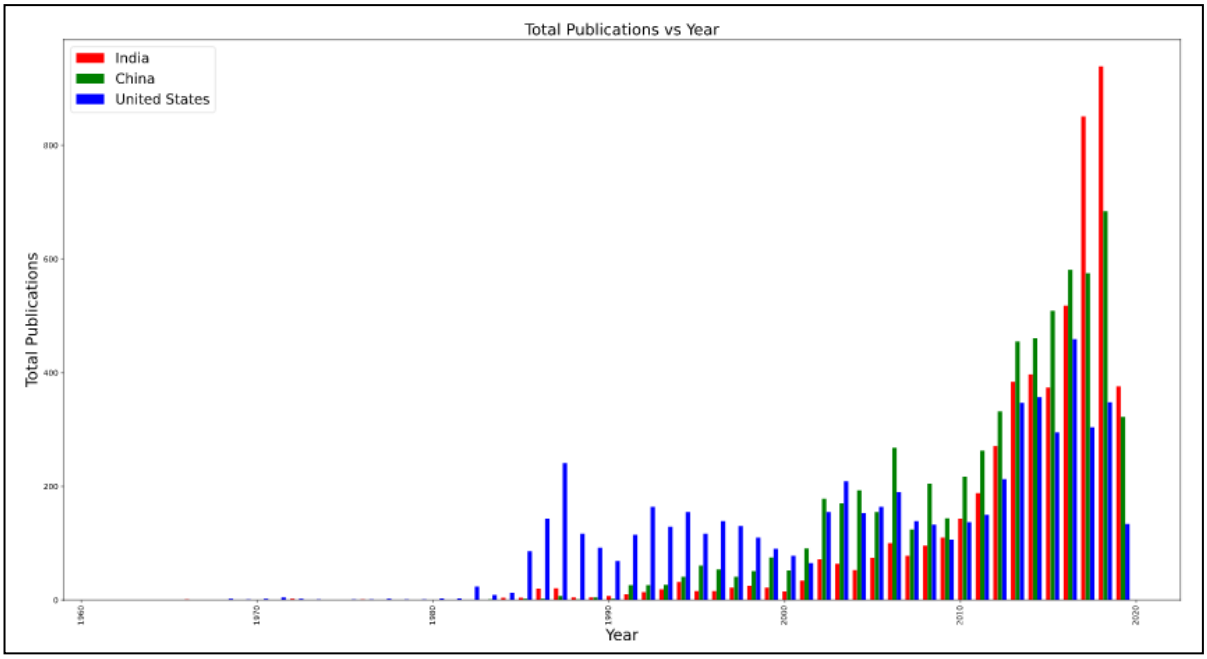
```
x_data =
[list(dct_india_year_publications.keys()),list(dct_china_year_publications.keys()),list(dct_usa_yea
r_publications.keys())]

y_data =
[list(dct_india_year_publications.values()),list(dct_china_year_publications.values()),list(dct_usa
_year_publications.values())]
```

1. Bar graph – Publication count vs Year

Code for plotting bar graph is as follow,

```
# plt.rcParams['figure.figsize'] = [20,10]
fig = plt.figure(figsize=[20,10])

X = np.arange(len(year_lst))
X = X + year_lst[0]

fig = fig.add_axes([0,0,1,1])

fig.bar(X + 0.00, list(dct_india_year_publications.values()), label='India', color='r', width
=0.25)
fig.bar(X + 0.25, list(dct_china_year_publications.values()), label='China', color='g', width
=0.25)
fig.bar(X + 0.50, list(dct_usa_year_publications.values()), label='United States', color='b',
width =0.25)

fig.legend(loc='upper left',fontsize=18)

plt.xticks(rotation = 'vertical')

plt.title('Total Publications vs Year',fontsize=20)
plt.xlabel('Year',fontsize=20)
plt.ylabel('Total Publications',fontsize=20)
plt.show()
```

We get following bar chart as output,

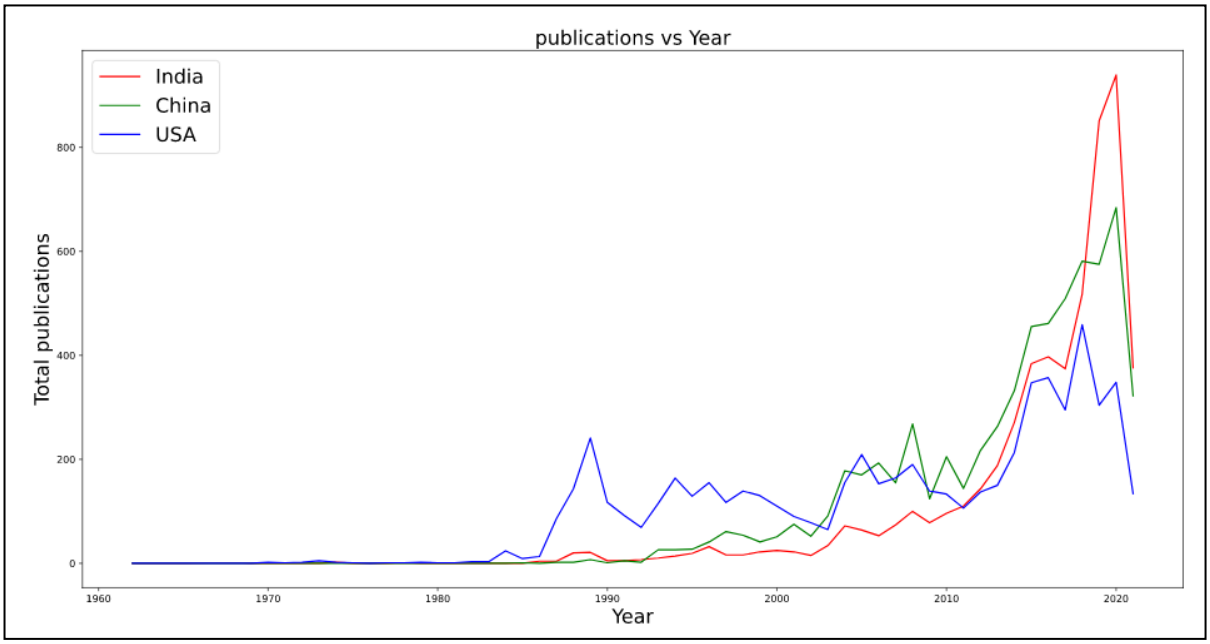


From bar graph we can see variation in publication year-wise for all 3-countries. India has highest publications in year 2019, China in 2017 and USA in 1988.

2. **Line Graph – Publication Count vs Year**
   a. **Rough curve**
      We first plotted the x and y data directly and got below curve
      Code:

```
fig = plt.figure(figsize=[20,10])

plt.plot(x_data[0], y_data[0],   label='India',  color='r')
plt.plot(x_data[1], y_data[1],   label='China',  color='g')
plt.plot(x_data[2], y_data[2],   label='USA',    color='b')

plt.xlabel('Year',fontsize=20)
plt.ylabel('Total publications',fontsize=20)
plt.title('Publications vs Year',fontsize=20)
plt.legend(loc='upper left',fontsize=20)
plt.show()
```

Output:



As we can see, the plot is not at all smooth, but it follows same trajectory as bar graph.

To generate smoother plot, we use interpolate library from scipy package.

b. Smooth plot

Code:

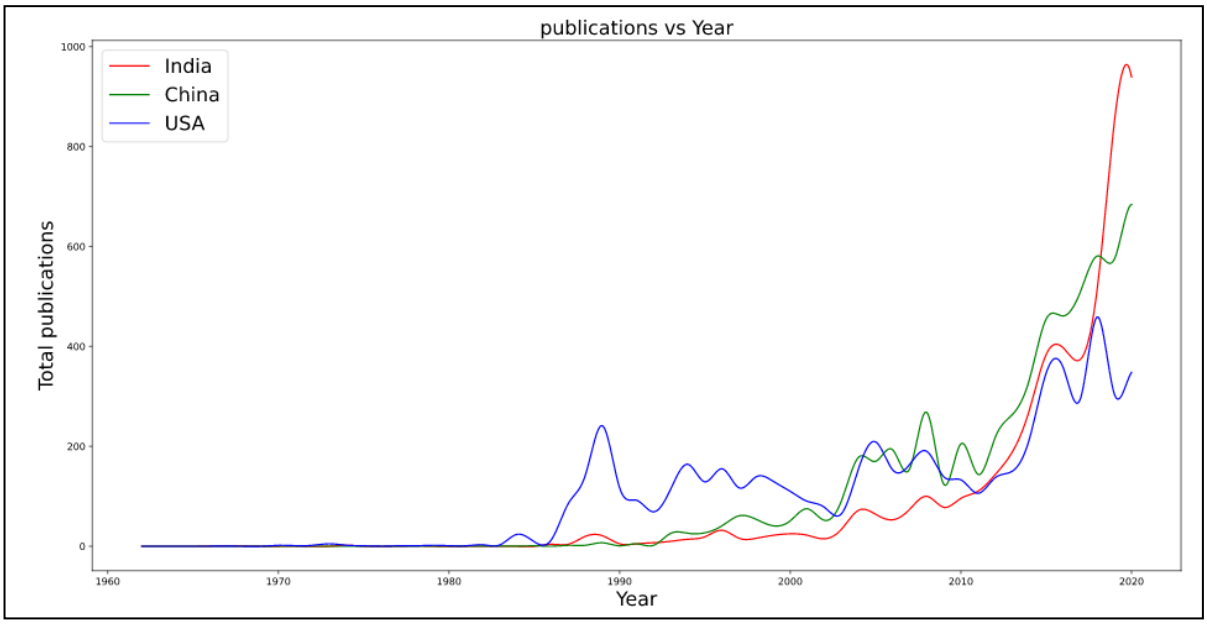
```
y_new = []
x_new = []
for i in range(3):
   x_new_tmp = np.linspace(year_lst[0],year_lst[0]+len(x_data[0]),1000)
   x_new.append(x_new_tmp)

   spline = interpolate.make_interp_spline(x_data[i], y_data[i])
   y_new.append(spline(x_new_tmp))
```

This code generates around 1000 points in the give interval using interpolation function. Using these new x and y we plotted new line graph.

Code:

```
fig = plt.figure(figsize=[20,10])

plt.plot(x_new[0], y_new[0],   label='India',  color='r')
plt.plot(x_new[1], y_new[1],   label='China',  color='g')
plt.plot(x_new[2], y_new[2],   label='USA',    color='b')

plt.xlabel('Year',fontsize=20)
plt.ylabel('Total publications',fontsize=20)
plt.title('Publications vs Year',fontsize=20)
plt.legend(loc='upper left',fontsize=20)
plt.show()
```

Output:



Thus we obtained a smoother curve and here we can clearly see that it follows same trajectory as bar graph.

j. Total number of grants given to the field

Grant is basically receiving funding from external organizations or institutions. In our case it was very easy to calculate grants, because we had already added funding column in our database. We then simply compared that 'funding details' column with zero and find out number of grants.

Code:

```
grants, cols = df_without_duplicates[df_without_duplicates['Funding_Details']!='0'].shape
```

Output:

```
[ ]  1  print(f'Grants given to field = {grants}')

     Grants given to field = 4771
```

As we can see from output, the grants received by field are **4,771**.

k. Country wise total number of publications

This is again simply because we had list of countries and also in our database there is country column. Using these two things we generated a dictionary with key as country name and value as publication count. For this we use following code,

Code:

```
dct_country_publications = {}
for country in country_lst:
   rows, columns = df[df.Country==country].shape
   dct_country_publications[country] = rows

dct_country_publications = sort_dict(dct_country_publications,'Value','reverse')
```

Output:

```
{'United Kingdom': 8994,
 'China': 6401,
 'United States': 6104,
 'India': 5383,
 'Germany': 5186,
 'Spain': 4759,
 'Canada': 4486,
 'Japan': 4324,
 'Italy': 4214,
 'France': 4133,
 'Australia': 3361,
 'South Korea': 3026,
 'Iran': 2720,
 'Taiwan': 2430,
 'Netherlands': 2173}
```

Chapter – V

Conclusion and References

Conclusion:

1. We this mini-project we had hands-on experience of data analysis using python.
2. We learned following things,
   a. How to download database from scopus
   b. What should we keep in mind while downloading database
   c. How to pursue problem properly so that it can be solved easily
3. We also had hands on experience on python libraries such as matplotlib, pandas, NumPy, SciPy, _pickle, etc.
4. Also, we got to know what tasks a person has to perform as a data scientist.

References:

1. Scopus
2. Pandas Documentation
3. Numpy Documentation
4. Matplotlib Documentation
5. GeeksForGeeks
6. StackOverFlow

1-Data Preprocessing

(Creating Complete Database file)

Importing Necessary Libraries

In [1]:
```
import pandas as pd
import numpy as np
import os

base_dir = '/mnt/General_Stuff/Study Stuff/Documents/CDS/Sem-II/BigDataAn
alytics/BigData-Programs/Mini-Project/Data'
```

Storing Country-wise Database file location into python list

In [2]:
```
dir_lst = []
for data_file in os.listdir(base_dir):
    dir_lst.append(os.path.join(base_dir,data_file))
```

In [3]:
```
dir_lst
```

Out[3]:
```
['/mnt/General_Stuff/Study Stuff/Documents/CDS/Sem-II/BigDataAnalytics/Bi
gData-Programs/Mini-Project/Data/Artificial-Intelligence-Australia.csv',
 '/mnt/General_Stuff/Study Stuff/Documents/CDS/Sem-II/BigDataAnalytics/Bi
gData-Programs/Mini-Project/Data/Artificial-Intelligence-Canada.csv',
 '/mnt/General_Stuff/Study Stuff/Documents/CDS/Sem-II/BigDataAnalytics/Bi
gData-Programs/Mini-Project/Data/Artificial-Intelligence-China.csv',
 '/mnt/General_Stuff/Study Stuff/Documents/CDS/Sem-II/BigDataAnalytics/Bi
gData-Programs/Mini-Project/Data/Artificial-Intelligence-France.csv',
 '/mnt/General_Stuff/Study Stuff/Documents/CDS/Sem-II/BigDataAnalytics/Bi
gData-Programs/Mini-Project/Data/Artificial-Intelligence-Germany.csv',
 '/mnt/General_Stuff/Study Stuff/Documents/CDS/Sem-II/BigDataAnalytics/Bi
gData-Programs/Mini-Project/Data/Artificial-Intelligence-India.csv',
 '/mnt/General_Stuff/Study Stuff/Documents/CDS/Sem-II/BigDataAnalytics/Bi
gData-Programs/Mini-Project/Data/Artificial-Intelligence-Iran.csv',
 '/mnt/General_Stuff/Study Stuff/Documents/CDS/Sem-II/BigDataAnalytics/Bi
gData-Programs/Mini-Project/Data/Artificial-Intelligence-Italy.csv',
 '/mnt/General_Stuff/Study Stuff/Documents/CDS/Sem-II/BigDataAnalytics/Bi
gData-Programs/Mini-Project/Data/Artificial-Intelligence-Japan.csv',
 '/mnt/General_Stuff/Study Stuff/Documents/CDS/Sem-II/BigDataAnalytics/Bi
gData-Programs/Mini-Project/Data/Artificial-Intelligence-Netherlands.csv
',
 '/mnt/General_Stuff/Study Stuff/Documents/CDS/Sem-II/BigDataAnalytics/Bi
gData-Programs/Mini-Project/Data/Artificial-Intelligence-SouthKorea.csv',
 '/mnt/General_Stuff/Study Stuff/Documents/CDS/Sem-II/BigDataAnalytics/Bi
gData-Programs/Mini-Project/Data/Artificial-Intelligence-Spain.csv',
 '/mnt/General_Stuff/Study Stuff/Documents/CDS/Sem-II/BigDataAnalytics/Bi
gData-Programs/Mini-Project/Data/Artificial-Intelligence-Taiwan.csv',
 '/mnt/General_Stuff/Study Stuff/Documents/CDS/Sem-II/BigDataAnalytics/Bi
gData-Programs/Mini-Project/Data/Artificial-Intelligence-UK.csv',
 '/mnt/General_Stuff/Study Stuff/Documents/CDS/Sem-II/BigDataAnalytics/Bi
gData-Programs/Mini-Project/Data/Artificial-Intelligence-US.csv']
```

Creating a Python list with country names

In [4]:
```
country_lst = ['Australia','Canada','China','France','Germany','India','I
ran','Italy','Japan','Netherlands','South Korea','Spain','Taiwan','United
Kingdom', 'United States']
```

Crossing checking number of database files and number of country names

In [5]:
```
len(dir_lst) == len(country_lst)
```

Out[5]:
```
True
```

Reading database of each country, removing unnecessary columns, adding country name column and storing those as a pandas dataframe into python list

In [6]:
```
df_lst = []

for data_file, country in zip(dir_lst, country_lst):
        print(country)
        df_tmp = pd.read_csv(data_file)

        df_tmp = df_tmp.drop(['Author(s) ID' ,'Source title','Volume', 'I
ssue', 'Art. No.', 'Page start', 'Page end', 'Page count', 'DOI', 'Link',
'Document Type', 'Publication Stage','Open Access', 'Source', 'EID'],axis
='columns')

        df_tmp = df_tmp.fillna(0)
        df_tmp['Country'] = country
        df_lst.append(df_tmp)
```

```
Australia
Canada
China
France
Germany
India
Iran
Italy
Japan
Netherlands
South Korea
Spain
Taiwan
United Kingdom
United States
```

In [7]:
```
print(country_lst[0])
df_lst[0].head()
```

```
Australia
```

Out[7]:

	Authors	Title	Year	Cited by	Country
0	Soares, J.V.B., Leandro, J.J.G., Cesar Jr., R....	Retinal vessel segmentation using the 2-D Gabo...	2006	1083.0	Australia
1	Scarselli, F., Gori, M., Tsoi, A.C., Hagenbuch...	The graph neural network model	2009	1031.0	Australia
2	Karantonis, D.M., Narayanan, M.R., Mathie, M.,...	Implementation of a real-time human movement c...	2006	908.0	Australia
3	Mirjalili, S.	Dragonfly algorithm: a new meta-heuristic opti...	2016	865.0	Australia
4	Naseem, I., Togneri, R., Bennamoun, M.	Linear regression for face recognition	2010	768.0	Australia

Merging coutry-databases from python list into a single pandas dataframe

In [8]:
```
df = pd.concat(df_lst)
```

In [9]:
```
df.shape
```

Out[9]:
```
(67694, 5)
```

In [10]:
```
df.head()
```

Out[10]:

	Authors	Title	Year	Cited by	Country
0	Soares, J.V.B., Leandro, J.J.G., Cesar Jr., R....	Retinal vessel segmentation using the 2-D Gabo...	2006	1083.0	Australia
1	Scarselli, F., Gori, M., Tsoi, A.C., Hagenbuch...	The graph neural network model	2009	1031.0	Australia
2	Karantonis, D.M., Narayanan, M.R., Mathie, M.,...	Implementation of a real-time human movement c...	2006	908.0	Australia
3	Mirjalili, S.	Dragonfly algorithm: a new meta-heuristic opti...	2016	865.0	Australia
4	Naseem, I., Togneri, R., Bennamoun, M.	Linear regression for face recognition	2010	768.0	Australia

Applying same data pipeline as above for databases with funding column

In [11]:
```
# Changin base_dir variable to path of database with funding column

base_dir = '/mnt/General_Stuff/Study Stuff/Documents/CDS/Sem-II/BigDataAn
alytics/BigData-Programs/Mini-Project/Data_with_sponsor_details'
```

In [12]:
```
dir_lst = []
for data_file in os.listdir(base_dir):
    dir_lst.append(os.path.join(base_dir,data_file))
```

In [13]:
```
df_lst = []

for data_file, country in zip(dir_lst, country_lst):

        print(country)

        df_tmp = pd.read_csv(data_file)

        df_tmp = df_tmp.drop(['Link','Publication Stage'],axis='columns')

        df_tmp = df_tmp.fillna(0)

        df_tmp['Country'] = country

        df_lst.append(df_tmp)
```

```
Australia
Canada
China
France
Germany
India
Iran
Italy
Japan
Netherlands
South Korea
Spain
Taiwan
United Kingdom
United States
```

In [14]:
```
df_addon = pd.concat(df_lst)
```

In [15]:
```
df_addon.head()
```

Out[15]:

	Authors	Title	Year	Cited by	Funding Details	Country
0	Tao F., Qi Q., Liu A., Kusiak A.	Data-driven smart manufacturing	2018	375.0	National Natural Science Foundation of China\n...	Australia
1	Zhang K., Gao X., Tao D., Li X.	Single image super-resolution with non-local m...	2012	362.0	National Natural Science Foundation of China\n...	Australia
2	Kristan M., Matas J., Leonardis A., Vojir T., ...	A Novel Performance Evaluation Methodology for...	2016	264.0	Seventh Framework Programme	Australia
3	Ding C., Choi J., Tao D., Davis L.S.	Multi-Directional Multi-Level Dual-Cross Patte...	2016	243.0	National Science Foundation\n\nAustralian Rese...	Australia
4	Celebi M.E., Kingravi H.A., Iyatomi H., Asland...	Border detection in dermoscopy images using st...	2008	241.0	National Cancer Institute	Australia

Adding Funding Column to previously merged database

In [16]:
```
df['Funding_Details'] = 0
```

Extracting titles from df and df_addon

In [17]:
```
titles_old = list(df.Title)
titles_new = list(df_addon.Title)
```

Adding funding values to df

In [18]:
```
count = 1
for i in range(len(titles_new)):
    for j in range(len(titles_old)):
        if titles_new[i]==titles_old[j]:
            df.iloc[j,-1] = df_addon.iloc[i,-2]
    # print(f'{count}\t{titles_new[i]}')
    count += 1
```

In [19]:
```
df.head()
```

Out[19]:

	Authors	Title	Year	Cited by	Country	Funding_Details
0	Soares, J.V.B., Leandro, J.J.G., Cesar Jr., R....	Retinal vessel segmentation using the 2-D Gabo...	2006	1083.0	Australia	0
1	Scarselli, F., Gori, M., Tsoi, A.C., Hagenbuch...	The graph neural network model	2009	1031.0	Australia	0
2	Karantonis, D.M., Narayanan, M.R., Mathie, M.,...	Implementation of a real-time human movement c...	2006	908.0	Australia	0
3	Mirjalili, S.	Dragonfly algorithm: a new meta-heuristic opti...	2016	865.0	Australia	0
4	Naseem, I., Togneri, R., Bennamoun, M.	Linear regression for face recognition	2010	768.0	Australia	0

Saving final database to 'csv' file

In [20]:
```
df.to_csv('Complete_database.csv')
```

2-Generating data required for further analysis

Importing Necessary Libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import _pickle as pickle
```

Importing Database

In [2]:
```
df = pd.read_csv('Complete_database.csv')
```

In [3]:
```
df[df.Country=='India']
```

Out[3]:

	Unnamed: 0	Authors	Title	Year	Cited by	Country	Funding_Details
23567	0	Jayadeva, Khemchandani, R., Chandra, S.	Twin support vector machines for pattern class...	2007	958.0	India	0
23568	1	Ravi, K., Ravi, V.	A survey on opinion mining and sentiment analy...	2015	617.0	India	0
23569	2	Varma, M., Zisserman, A.	A statistical approach to material classificat...	2009	479.0	India	University of Oxford\n \nEuropean Commission
23570	3	Nayak, P.C., Sudheer, K.P., Rangan, D.M., Rama...	A neuro-fuzzy computing technique for modeling...	2004	466.0	India	0
23571	4	Kale, A., Sundaresan, A., Rajagopalan, A.N., C...	Identification of humans using gait	2004	445.0	India	0
...	...	...	...	...	...	...	...
28945	5378	Narasimhan, R.	Artificial intelligence in fifth generation co...	1986	0.0	India	0
28946	5379	Ramani, S., Chandrasekar, R.	Partitioning computations and parallel processing	1986	0.0	India	0
28947	5380	Krishna, M.H., Murty, N.M.	A conceptual clustering scheme for frame-based...	1986	0.0	India	0
28948	5381	Krishnamurthy, E.V., Subramanian, K., Mahadeva...	Formal description, compression and transforma...	1974	0.0	India	0
28949	5382	Aggarwal, G.K.	On negative character of information	1968	0.0	India	0

5383 rows × 7 columns

In [4]:
```
df.head()
```

Out[4]:

	Unnamed: 0	Authors	Title	Year	Cited by	Country	Funding_Details
0	0	Soares, J.V.B., Leandro, J.J.G., Cesar Jr., R....	Retinal vessel segmentation using the 2-D Gabo...	2006	1083.0	Australia	0
1	1	Scarselli, F., Gori, M., Tsoi, A.C., Hagenbuch...	The graph neural network model	2009	1031.0	Australia	0
2	2	Karantonis, D.M., Narayanan, M.R., Mathie, M.,...	Implementation of a real-time human movement c...	2006	908.0	Australia	0
3	3	Mirjalili, S.	Dragonfly algorithm: a new meta-heuristic opti...	2016	865.0	Australia	0
4	4	Naseem, I., Togneri, R., Bennamoun, M.	Linear regression for face recognition	2010	768.0	Australia	0

In [5]:
```
df.drop('Unnamed: 0',axis='columns',inplace=True)
```

In [6]:
```
df = df.rename(columns={'Cited by':'Cited_by'})
```

In [7]:
```
df.head()
```

Out[7]:

	Authors	Title	Year	Cited_by	Country	Funding_Details
0	Soares, J.V.B., Leandro, J.J.G., Cesar Jr., R....	Retinal vessel segmentation using the 2-D Gabo...	2006	1083.0	Australia	0
1	Scarselli, F., Gori, M., Tsoi, A.C., Hagenbuch...	The graph neural network model	2009	1031.0	Australia	0
2	Karantonis, D.M., Narayanan, M.R., Mathie, M.,...	Implementation of a real-time human movement c...	2006	908.0	Australia	0
3	Mirjalili, S.	Dragonfly algorithm: a new meta-heuristic opti...	2016	865.0	Australia	0
4	Naseem, I., Togneri, R., Bennamoun, M.	Linear regression for face recognition	2010	768.0	Australia	0

In [8]:
```
df_without_countries = df.drop('Country',axis='columns')
```

In [9]:
```
df_without_countries = df_without_countries.drop_duplicates(subset=['Title'],keep='first')
```

In [10]:
```
df_without_countries.shape
```

Out[10]:
```
(59215, 5)
```

Generating Author:database dictonary

In [11]:
```
authors_lst = list(df_without_countries.loc[:,'Authors'].values)
```

In [12]:
```
print(authors_lst[0])
```

```
Soares, J.V.B., Leandro, J.J.G., Cesar Jr., R.M., Jelinek, H.F., Cree, M.J.
```

In [13]:
```
len(authors_lst)
```

Out[13]:
```
59215
```

In [14]:
```
set_authors = ['Jayadeva', 'Khemchandani, R.', 'Chandra, S.']
for i in range(len(authors_lst)):
    authors_sub_lst = authors_lst[i].split(',')
    authors_sub_lst_mod = []
    if authors_sub_lst == 'Jayadeva, Khemchandani, R., Chandra, S.'.split(','):
        continue
    for j in range(0,len(authors_sub_lst)-1,2):
        authors_sub_lst_mod.append(authors_sub_lst[j].strip()+','+authors_sub_l
st[j+1])

    for author in authors_sub_lst_mod:
        if(author not in set_authors):
            set_authors.append(author)
            # print(f'{i}\t{author}')
```

Storing Authors Names to a file so that it can be easily availabe afterwards

In [15]:
```
with open('Authors_list.txt','w') as filehandle:
        filehandle.writelines("%s\n" % author for author in set_authors)
filehandle.close()
```

Reading Authors from Above file and storing into a python list

In [16]:
```
set_authors = []
with open('Authors_list.txt','r') as filehandle:
    filecontents = filehandle.readlines()

    for line in filecontents:
        # remove linebreak which is the last character of the string
        author = line[:-1]

        # add item to the list
        set_authors.append(author)
```

In [17]:
```
len(set_authors)
```

Out[17]:
```
120161
```

Creating Author:Database Dictonary

In [18]:
```
dct_author_database = {}
count=0
for author in set_authors:
    # print(f'{count}\t{author}')

    df_auth = pd.DataFrame(columns = ['Authors', 'Title', 'Year', 'Cited_by','F
unding Details'] )

    filt= df_without_countries['Authors'].str.contains(author, na=False)
    df_auth= df_without_countries.loc[filt,'Authors':].reset_index(drop=True)

    dct_author_database[author] = df_auth

    count += 1
```

```
/home/deshabhakt/.local/lib/python3.8/site-packages/pandas/core/strings/accesso
r.py:101: UserWarning: This pattern has match groups. To actually get the group
s, use str.extract.
  return func(self, *args, **kwargs)
```

Alterative to above code

''' dct_author_database = {}

count = 0 for author in set_authors:

```
    # print(f'{count}\t{author}')

    df_auth = pd.DataFrame(columns = ['Authors', 'Title', 'Year', 'Cited_by'] )

    for authors in authors_lst:

        if author in authors:
            df_tmp = df_without_countries[df_without_countries.Authors==authors]
            df_auth = df_auth.append(df_tmp,ignore_index=True)

    dct_author_database[author] = df_auth

    count += 1
```

'''

Storing Author:Database dictonary to file so that it can reused again easily

In [19]:
```
with open('Author_database_dictonary.txt','wb') as file:
    file.write(pickle.dumps(dct_author_database))
file.close()
```

Reading Author:Database dictonary from Author_database_dictonary.txt file

In [ ]:
```
dct_author_database = {}
with open('Author_database_dictonary.txt','rb') as file:
    dct_author_database = pickle.load(file)
file.close()
```

In [20]:
```
print(type(dct_author_database))
```

```
<class 'dict'>
```

In [21]:
```
print(set_authors[10])
dct_author_database[set_authors[10]]
```

```
Tsoi, A.C.
```

Out[21]:

	Authors	Title	Year	Cited_by	Funding_Details
0	Scarselli, F., Gori, M., Tsoi, A.C., Hagenbuch...	The graph neural network model	2009	1031.0	0
1	Shilton, A., Palaniswami, M., Ralph, D., Tsoi,...	Incremental training of support vector machines	2005	159.0	0
2	Scarselli, F., Tsoi, A.C., Hagenbuchner, M., N...	Solving graph data issues using a layered arch...	2013	10.0	0
3	Scarselli, F., Tsoi, A.C., Hagenbuchner, M.	The Vapnik–Chervonenkis dimension of graph and...	2018	4.0	0
4	Pucci, A., Gori, M., Hagenbuchner, M., Scarsel...	Investigations into the application of Graph N...	2006	3.0	0

Creating indian authors list and foreign authors list and storing those in file

Creating Authors_collective list from Indian database

In [22]:
```
authors_from_ind_database = list(df[df.Country=='India']['Authors'].unique())
```

separating individual authors from authors_collective list

In [25]:
```
set_of_authors_from_indian_database = ['Jayadeva', 'Khemchandani, R.', 'Chandr
a, S.']

count = 0
for authors in authors_from_ind_database:

    authors_sub_lst = authors.split(',')

    if authors_sub_lst == 'Jayadeva, Khemchandani, R., Chandra, S.'.split(','):
        continue

    authors_sub_lst_mod = []
    for i in range(0,len(authors_sub_lst)-1,2):
            authors_sub_lst_mod.append(authors_sub_lst[i].strip()+','+authors_s
ub_lst[i+1])

    for author in authors_sub_lst_mod:
        if(author not in set_of_authors_from_indian_database):
            set_of_authors_from_indian_database.append(author)
    # print(f'{count}\t{authors}')
    count += 1
```

Storing Indian Authors names to file

In [26]:
```
with open('Indian_authors_list.txt','w') as filehandle:
        filehandle.writelines("%s\n" % author for author in set_of_authors_from
_indian_database)
filehandle.close()
```

After this step we manually removed non-indian authors from the Indian authors list

Reading Indian Authors names to file

In [27]:
```
set_of_indian_authors_from_file = []
with open('Indian_authors_list.txt','r') as filehandle:
    filecontents = filehandle.readlines()

    for line in filecontents:
        # remove linebreak which is the last character of the string
        author = line[:-1]

        # add item to the list
        set_of_indian_authors_from_file.append(author)
```

In [28]:
```
len(set_of_indian_authors_from_file)
```

Out[28]:
```
9267
```

Creating Foreign authors list

In [30]:
```
set_of_foreign_authors = []

count = 0
for author in set_authors:
    if author not in set_of_indian_authors_from_file:
        set_of_foreign_authors.append(author)
        # print(f'{count}\t{author}')
        count += 1
```

Storing Foreign authors in txt file so that it can be reused again easily

In [32]:
```
with open('Foreign_authors_list.txt','w') as filehandle:
        filehandle.writelines("%s\n" % author for author in set_of_foreign_auth
ors)
filehandle.close()
```

Reading Foreign authors from foreign authors list file

In [33]:
```
set_of_foreign_authors_from_file = []
with open('Foreign_authors_list.txt','r') as filehandle:
    filecontents = filehandle.readlines()

    for line in filecontents:
        # remove linebreak which is the last character of the string
        author = line[:-1]

        # add item to the list
        set_of_foreign_authors_from_file.append(author)
```

In [34]:
```
print(len(set_of_foreign_authors_from_file))
```

```
110892
```

Creating a dictonary with foreign author as key and number of paper published by him with india authors as value

In [35]:
```
dct_foreign_author_coauth_count = {}
for foreign_author in set_of_foreign_authors_from_file:
    dct_foreign_author_coauth_count[foreign_author] = 0
```

In [37]:
```
count = 0
for foreign_author in set_of_foreign_authors_from_file:
    row_foreign_auth_list = dct_author_database[foreign_author].iloc[:,0].value
s
    for indian_author in set_of_indian_authors_from_file:
        for authors in row_foreign_auth_list:
            if indian_author in authors:
                dct_foreign_author_coauth_count[foreign_author] += 1
    # print(f'{count}     {foreign_author}        {dct_foreign_author_coauth_co
unt[foreign_author]}')
    count += 1
```

Storing above dictonary in file so that it can be easily reused again

In [38]:
```
with open('Foreign_auth_and_their_publication_count_with_india_authors_dct.txt
','wb') as file:
    file.write(pickle.dumps(dct_foreign_author_coauth_count))
file.close()
```

Reading above stored dictonary from file and storing it in a python dictonary variable

In [ ]:
```
dct_foreign_author_coauth_count_from_file = {}
with open('Foreign_auth_and_their_publication_count_with_india_authors_dct.txt
','rb') as file:
    dct_foreign_author_coauth_count_from_file = pickle.load(file)
file.close()
```

Finding Answers

Importing Necessary Libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import scipy.interpolate as interpolate
import _pickle as pickle
```

Function to Sort Dictonary

In [2]:
```
def sort_dict(dct,parameter,order='Ascending'):
    if parameter=='key':
        if(order=='reverse'):
            sorted_tuples = sorted(dct.items(),key=lambda item:item[0],reverse=True)
            return {k:v for k,v in sorted_tuples}
        else:
            sorted_tuples = sorted(dct.items(),key=lambda item:item[0])
            return {k:v for k,v in sorted_tuples}
    else:
        if(order=='reverse'):
            sorted_tuples = sorted(dct.items(),key=lambda item:item[1],reverse=True)
            return {k:v for k,v in sorted_tuples}
        else:
            sorted_tuples = sorted(dct.items(),key=lambda item:item[1])
            return {k:v for k,v in sorted_tuples}
```

Importing Database

In [3]:
```
df = pd.read_csv('Complete_database.csv')
```

In [4]:
```
df.head()
```

Out[4]:

	Unnamed: 0	Authors	Title	Year	Cited by	Country	Funding_Details
0	0	Soares, J.V.B., Leandro, J.J.G., Cesar Jr., R....	Retinal vessel segmentation using the 2-D Gabo...	2006	1083.0	Australia	0
1	1	Scarselli, F., Gori, M., Tsoi, A.C., Hagenbuch...	The graph neural network model	2009	1031.0	Australia	0
2	2	Karantonis, D.M., Narayanan, M.R., Mathie, M.,...	Implementation of a real-time human movement c...	2006	908.0	Australia	0
3	3	Mirjalili, S.	Dragonfly algorithm: a new meta-heuristic opti...	2016	865.0	Australia	0
4	4	Naseem, I., Togneri, R., Bennamoun, M.	Linear regression for face recognition	2010	768.0	Australia	0

In [5]:
```
df.shape
```

Out[5]:
```
(67694, 7)
```

Removing Unnamed column and renaming Cited by column for ease of use

In [6]:
```
df.drop('Unnamed: 0',axis='columns',inplace=True)
```

In [7]:
```
df.rename(columns={'Cited by':'Cited_by','Funding Details':'Funding_Details'},inplace
=True)
```

Reading Authors names from previously created file and storing those in python list

In [8]:
```
set_authors = []
with open('Authors_list.txt','r') as filehandle:
    filecontents = filehandle.readlines()

    for line in filecontents:
        # remove linebreak which is the last character of the string
        author = line[:-1]

        # add item to the list
        set_authors.append(author)
filehandle.close()
```

Reading Previously Created Author:Database Dictonary

In [9]:
```
dct_author_database = {}
with open('Author_database_dictonary.txt','rb') as file:
    dct_author_database = pickle.load(file)
file.close()
```

In [10]:
```
index = 6
print(set_authors[index])
dct_author_database[set_authors[index]]
```

```
Jelinek, H.F.
```

Out[10]:

	Authors	Title	Year	Cited_by	Funding Details
0	Soares, J.V.B., Leandro, J.J.G., Cesar Jr., R....	Retinal vessel segmentation using the 2-D Gabo...	2006	1083.0	0
1	Rocha, A., Carvalho, T., Jelinek, H.F., Golden...	Points of interest and visual dictionaries for...	2012	96.0	0
2	Jelinek, H.F., Cree, M.J., Leandro, J.J.G., So...	Automated segmentation of retinal blood vessel...	2007	47.0	0
3	Hassan, M.M., Huda, S., Yearwood, J., Jelinek,...	Multistage fusion approaches based on a genera...	2018	18.0	0
4	Abawajy, J., Kelarev, A., Chowdhury, M., Stran...	Predicting cardiac autonomic neuropathy catego...	2013	15.0	0

a) Highest cited author and his h-index (from the world)

In [11]:
```
author_with_highest_citations = ""
max_citations = 0
for author in set_authors:
    cites = dct_author_database[author]['Cited_by'].sum()
    if max_citations<cites:
        author_with_highest_citations = author
        max_citations = cites
```

In [12]:
```
df_of_highest_cited_author = dct_author_database[author_with_highest_citations]
rows,cols = df_of_highest_cited_author.shape
avg_citations_of_author_with_highest_citations = max_citations/rows
h_index = min(rows,avg_citations_of_author_with_highest_citations)
```

In [13]:
```
print(f'Max cited author    = {author_with_highest_citations}')
print(f'Total cited by      = {max_citations}')
print(f'His h-index         = {h_index}')
```

```
Max cited author    = Hassabis, D.
Total cited by      = 17466.0
His h-index         = 13
```

b) Highest publication author

In [14]:
```
author_with_highest_publication = ""
max_publication_count = 0
for author in set_authors:
    rows, columns = dct_author_database[author].shape
    if rows>max_publication_count:
        max_publication_count=rows
        author_with_highest_publication = author
        # print(f'{author}                    \t{rows}')
```

In [15]:
```
print(f'Author\t\t= {author_with_highest_publication}\nPublications\t= {max_publication_count}')
```

```
Author          = Wang, Y.
Publications    = 439
```

c) Highest cited authors avg. citations, and the country name.

In [16]:
```
dct_author_database[author_with_highest_citations]
```

Out[16]:

	Authors	Title	Year	Cited_by	Funding Details
0	Vinyals, O., Babuschkin, I., Czarnecki, W.M., ...	Grandmaster level in StarCraft II using multi-...	2019	224.0	0
1	Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A...	Human-level control through deep reinforcement...	2015	7346.0	0
2	Silver, D., Huang, A., Maddison, C.J., Guez, A...	Mastering the game of Go with deep neural netw...	2016	5282.0	0
3	Silver, D., Schrittwieser, J., Simonyan, K., A...	Mastering the game of Go without human knowledge	2017	2391.0	0
4	Kirkpatrick, J., Pascanu, R., Rabinowitz, N., ...	Overcoming catastrophic forgetting in neural n...	2017	647.0	0
5	De Fauw, J., Ledsam, J.R., Romera-Paredes, B.,...	Clinically applicable deep learning for diagno...	2018	602.0	0
6	Silver, D., Hubert, T., Schrittwieser, J., Ant...	A general reinforcement learning algorithm tha...	2018	457.0	0
7	McKinney, S.M., Sieniek, M., Godbole, V., Godw...	International evaluation of an AI system for b...	2020	266.0	NIHR Imperial Biomedical Research Centre\n\nOf...
8	Wang, J.X., Kurth-Nelson, Z., Kumaran, D., Tir...	Prefrontal cortex as a meta-reinforcement lear...	2018	112.0	0
9	Jaderberg, M., Czarnecki, W.M., Dunning, I., M...	Human-level performance in 3D multiplayer game...	2019	71.0	0
10	Dabney, W., Kurth-Nelson, Z., Uchida, N., Star...	A distributional code for value in dopamine-ba...	2020	43.0	0
11	Yim, J., Chopra, R., Spitz, T., Winkens, J., O...	Predicting conversion to wet age-related macul...	2020	23.0	0
12	Schrittwieser, J., Antonoglou, I., Hubert, T.,...	Mastering Atari, Go, chess and shogi by planni...	2020	2.0	0

In [17]:
```
rows, cols = dct_author_database[author_with_highest_citations].shape

avg_citations_of_author_with_highest_citations = max_citations/rows
print(f'Highest Cited Author \t= {author_with_highest_citations}')
print(f'His Average Citations \t= {avg_citations_of_author_with_highest_citations}')
print(f'His Total Publications \t= {rows}')
```

```
Highest Cited Author    = Hassabis, D.
His Average Citations   = 1343.5384615384614
His Total Publications  = 13
```

A google search with above author's name tells us that he's from 'United Kingdom' and doing reasearch on 'Artificial Intelligence'

d) Total number of publications of the highest cited author

In [18]:
```
print(f'Highest Cited Author \t= {author_with_highest_citations}')
print(f'Total Publications \t= {rows}')
dct_author_database[author_with_highest_citations]
```

```
Highest Cited Author    = Hassabis, D.
Total Publications      = 13
```

Out[18]:

	Authors	Title	Year	Cited_by	Funding Details
0	Vinyals, O., Babuschkin, I., Czarnecki, W.M., ...	Grandmaster level in StarCraft II using multi-...	2019	224.0	0
1	Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A...	Human-level control through deep reinforcement...	2015	7346.0	0
2	Silver, D., Huang, A., Maddison, C.J., Guez, A...	Mastering the game of Go with deep neural netw...	2016	5282.0	0
3	Silver, D., Schrittwieser, J., Simonyan, K., A...	Mastering the game of Go without human knowledge	2017	2391.0	0
4	Kirkpatrick, J., Pascanu, R., Rabinowitz, N., ...	Overcoming catastrophic forgetting in neural n...	2017	647.0	0
5	De Fauw, J., Ledsam, J.R., Romera-Paredes, B.,...	Clinically applicable deep learning for diagno...	2018	602.0	0
6	Silver, D., Hubert, T., Schrittwieser, J., Ant...	A general reinforcement learning algorithm tha...	2018	457.0	0
7	McKinney, S.M., Sieniek, M., Godbole, V., Godw...	International evaluation of an AI system for b...	2020	266.0	NIHR Imperial Biomedical Research Centre\n\nOf...
8	Wang, J.X., Kurth-Nelson, Z., Kumaran, D., Tir...	Prefrontal cortex as a meta-reinforcement lear...	2018	112.0	0
9	Jaderberg, M., Czarnecki, W.M., Dunning, I., M...	Human-level performance in 3D multiplayer game...	2019	71.0	0
10	Dabney, W., Kurth-Nelson, Z., Uchida, N., Star...	A distributional code for value in dopamine-ba...	2020	43.0	0
11	Yim, J., Chopra, R., Spitz, T., Winkens, J., O...	Predicting conversion to wet age-related macul...	2020	23.0	0
12	Schrittwieser, J., Antonoglou, I., Hubert, T.,...	Mastering Atari, Go, chess and shogi by planni...	2020	2.0	0

e) Total publication in year

In [19]:
```
year_lst = sorted(list(df['Year'].unique()))
country_lst = list(df['Country'].unique())
```

In [20]:
```
df_without_duplicates = df.drop_duplicates(subset=['Authors','Title'],keep='first')
```

In [21]:
```
dct_df_per_year_publications = {}
for year in year_lst:
    dct_df_per_year_publications[year], cols = df_without_duplicates[df_without_dupli
cates.Year==year].shape
```

In [22]:
```
dct_df_per_year_publications = sort_dict(dct_df_per_year_publications,'Values','rever
se')
dct_df_per_year_publications
```

Out[22]:
```
{2020: 6222,
 2019: 4688,
 2018: 4122,
 2016: 3341,
 2014: 3159,
 2015: 3117,
 2017: 3049,
 2021: 2688,
 2013: 2277,
 2004: 2191,
 2012: 2010,
 2008: 1933,
 2009: 1565,
 2011: 1559,
 2010: 1542,
 2006: 1464,
 2007: 1401,
 2003: 1357,
 2005: 1346,
 2000: 793,
 1989: 734,
 1999: 730,
 2001: 726,
 1997: 718,
 1996: 713,
 1994: 703,
 2002: 702,
 1998: 697,
 1995: 601,
 1988: 531,
 1993: 485,
 1990: 457,
 1991: 394,
 1992: 357,
 1987: 351,
 1986: 165,
 1985: 117,
 1984: 70,
 1983: 25,
 1982: 23,
 1977: 17,
 1980: 13,
 1973: 12,
 1978: 12,
 1979: 12,
 1981: 11,
 1974: 8,
 1975: 8,
 1976: 8,
 1971: 7,
 1972: 6,
 1970: 3,
 1962: 1,
 1963: 1,
 1964: 1,
 1965: 1,
 1968: 1,
 1969: 1}
```

f) Total citation per year

In [23]:
```
dct_citations_per_year = {}
for year in year_lst:
    dct_citations_per_year[year] = df_without_duplicates[df_without_duplicates.Year==
year]['Cited_by'].sum()

dct_citations_per_year = sort_dict(dct_citations_per_year,'Values','reverse')
```

In [24]:
```
dct_citations_per_year
```

Out[24]:
```
{2008: 86548.0,
 2016: 85426.0,
 2015: 81343.0,
 2005: 78325.0,
 2004: 75544.0,
 2007: 73896.0,
 2006: 73652.0,
 2014: 69065.0,
 2009: 68433.0,
 2018: 66871.0,
 2013: 63245.0,
 2017: 61390.0,
 2010: 61118.0,
 2012: 56435.0,
 2011: 55280.0,
 2019: 42281.0,
 2000: 36085.0,
 2003: 32783.0,
 1999: 28128.0,
 1998: 26193.0,
 2002: 25418.0,
 2001: 25176.0,
 1997: 21638.0,
 2020: 21386.0,
 1995: 18774.0,
 1994: 18694.0,
 1996: 16943.0,
 1989: 12072.0,
 1992: 10540.0,
 1990: 9800.0,
 1993: 8807.0,
 1991: 8453.0,
 1988: 8385.0,
 1987: 5439.0,
 1986: 3652.0,
 1980: 2616.0,
 1985: 1930.0,
 1977: 1644.0,
 2021: 1568.0,
 1979: 997.0,
 1984: 812.0,
 1971: 446.0,
 1973: 256.0,
 1978: 252.0,
 1976: 215.0,
 1983: 208.0,
 1982: 202.0,
 1975: 186.0,
 1970: 83.0,
 1981: 76.0,
 1972: 49.0,
 1974: 33.0,
 1963: 17.0,
 1962: 16.0,
 1969: 4.0,
 1964: 2.0,
 1965: 2.0,
 1968: 0.0}
```

g) Author(country) having highest co-authorship with indian authors.

Reading India Authors name from previously created file

In [25]:
```
set_of_indian_authors = []
with open('Indian_authors_list.txt','r') as filehandle:
    filecontents = filehandle.readlines()

    for line in filecontents:
        # remove linebreak which is the last character of the string
        author = line[:-1]

        # add item to the list
        set_of_indian_authors.append(author)
filehandle.close()
```

Reading Foreign Authors name from previously created file

In [26]:
```
set_of_foreign_authors = []
with open('Foreign_authors_list.txt','r') as filehandle:
    filecontents = filehandle.readlines()

    for line in filecontents:
        # remove linebreak which is the last character of the string
        author = line[:-1]

        # add item to the list
        set_of_foreign_authors.append(author)
```

Loading Previously created dictonary from file

In [27]:
```
dct_foreign_author_coauth_count = {}
with open('Foreign_auth_and_their_publication_count_with_india_authors_dct.txt','rb')
as file:
    dct_foreign_author_coauth_count = pickle.load(file)
file.close()
```

Finding Foreign author with highest co-authorship with India authors

In [28]:
```
mx_pubs_with_indian_authors = 0
foreign_auth_corr_to_mx_pubs_with_indian_authors = ""
for author in set_of_foreign_authors:
    pubs = dct_foreign_author_coauth_count[author]
    if mx_pubs_with_indian_authors < pubs:
        mx_pubs_with_indian_authors = pubs
        foreign_auth_corr_to_mx_pubs_with_indian_authors = author
```

In [29]:
```
print(f'Foreign author with Highest Co-authorship with Indian Authors =  {foreign_aut
h_corr_to_mx_pubs_with_indian_authors}')
print(mx_pubs_with_indian_authors)
```

```
Foreign author with Highest Co-authorship with Indian Authors =  Nicolaides, A.
127
```

A google search with above authors name tells us that he is from "Vascular Screening and Diagnostic Centre, University of Nicosia, Nicosia, Cyprus"

h) Highest cited author from India and the university

In [31]:
```
highest_cited_indian_author
```

Out[31]:
```
'Raghava, G.P.'
```

In [30]:
```
max_cites_of_indian_author = 0
highest_cited_indian_author = ""

for author in set_of_indian_authors:
    cites = dct_author_database[author]['Cited_by'].sum()
    if max_cites_of_indian_author < cites:
        max_cites_of_indian_author = cites
        highest_cited_indian_author = author
```

In [32]:
```
print(f'Highest Cited Author from India = {highest_cited_indian_author}')
print(f'His Total Citations = {max_cites_of_indian_author}')
```

```
Highest Cited Author from India = Raghava, G.P.
His Total Citations = 3132.0
```

Out[32]:

	Authors	Title	Year	Cited_by	Funding Details
0	Saha, S., Raghava, G.P.S.	AlgPred: Prediction of allergenic proteins and...	2006	348.0	0
1	Gupta, S., Kapoor, P., Chaudhary, K., Gautam, ...	In Silico Approach for Predicting Toxicity of ...	2013	298.0	0
2	Bhasin, M., Raghava, G.P.S.	ESLpred: SVM-based method for subcellular loca...	2004	252.0	0
3	Bhasin, M., Raghava, G.P.S.	Prediction of CTL epitopes using QM, SVM and A...	2004	239.0	0
4	Kumar, M., Gromiha, M.M., Raghava, G.P.S.	Prediction of RNA binding sites in a protein u...	2008	202.0	0
5	Bhasin, M., Garg, A., Raghava, G.P.S.	PSLpred: Prediction of subcellular localizatio...	2005	161.0	0
6	Singh, H., Ansari, H.R., Raghava, G.P.S.	Improved Method for Linear B-Cell Epitope Pred...	2013	158.0	0
7	Kumar, M., Gromiha, M.M., Raghava, G.P.S.	Identification of DNA-binding proteins using s...	2007	158.0	0
8	Bhasin, M., Raghava, G.P.S.	GPCRpred: An SVM-based method for prediction o...	2004	151.0	0
9	Dhanda, S.K., Vir, P., Raghava, G.P.S.	Designing of interferon-gamma inducing MHC cla...	2013	140.0	0
10	Bhasin, M., Raghava, G.P.S.	SVM based method for predicting HLA-DRB1*0401 ...	2004	115.0	0
11	Bhasin, M., Raghava, G.P.S.	Analysis and prediction of affinity of TAP bin...	2004	113.0	0
12	Rashid, M., Saha, S., Raghava, G.P.S.	Support Vector Machine-based method for predic...	2007	101.0	0
13	Kaundal, R., Kapoor, A.A., Raghava, G.P.S.	Machine learning techniques in disease forecas...	2006	83.0	0
14	Bhasin, M., Raghava, G.P.S.	Pcleavage: An SVM based method for prediction ...	2005	75.0	0
15	Kumar, M., Bhasin, M., Natt, N.K., Raghava, G....	BhairPred: Prediction of β-hairpins in a prote...	2005	63.0	0
16	Mishra, N.K., Agarwal, S., Raghava, G.P.S.	Prediction of cytochrome P450 isoform responsi...	2010	55.0	0
17	Bhasin, M., Raghava, G.P.S.	GPCRsclass: A web tool for the classification ...	2005	55.0	0
18	Verma, R., Varshney, G.C., Raghava, G.P.S.	Prediction of mitochondrial proteins of malari...	2010	43.0	0
19	Garg, A., Raghava, G.P.S.	ESLpred2: Improved method for predicting subce...	2008	43.0	0
20	Garg, A., Raghava, G.P.S.	A machine learning based method for the predic...	2008	40.0	0
21	Lata, S., Bhasin, M., Raghava, G.P.	Application of machine learning techniques in ...	2007	38.0	0
22	Kaundal, R., Raghava, G.P.S.	RSLpred: An integrative system for predicting ...	2009	34.0	0
23	Rashid, M., Ramasamy, S., Raghava, G.P.S.	A simple approach for predicting protein-prote...	2010	33.0	0
24	Kumar, M., Raghava, G.P.	Prediction of nuclear proteins using SVM and H...	2009	29.0	0
25	Bhasin, M., Lata, S., Raghava, G.P.	TAPPred prediction of TAP-binding peptides in ...	2007	27.0	0
26	Verma, R., Tiwari, A., Kaur, S., Varshney, G.C...	Identification of proteins secreted by malaria...	2008	25.0	0

	Authors	Title	Year	Cited_by	Funding Details
27	Kalita, M.K., Nandal, U.K., Pattnaik, A., Siva...	CyclinPred: A SVM-based method for predicting ...	2008	23.0	0
28	Mishra, N.K., Kumar, M., Raghava, G.P.S.	Support vector machine based prediction of glu...	2007	16.0	0

i) Comparative year wise article publication analysis of India, China and USA

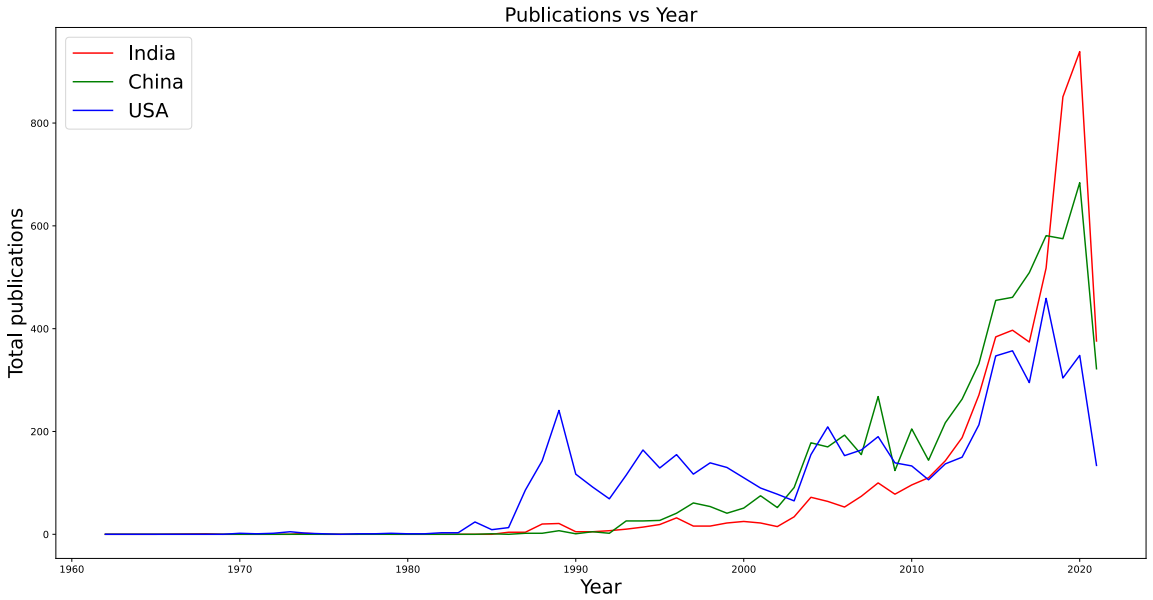
In [33]:
```
df_india = df[df.Country=='India'].copy().reset_index(drop=True)
df_china = df[df.Country=='China'].copy().reset_index(drop=True)
df_usa = df[df.Country=='United States'].copy().reset_index(drop=True)
```

In [34]:
```
dct_india_year_publications = {}
dct_china_year_publications = {}
dct_usa_year_publications = {}
for year in year_lst:
    dct_india_year_publications[year], cols1 = df_india[df_india.Year==year].shape
    dct_china_year_publications[year], cols2 = df_china[df_china.Year==year].shape
    dct_usa_year_publications[year], cols3 = df_usa[df_usa.Year==year].shape
```

Comparative analysis using graph

In [35]:
```
x_data = [list(dct_india_year_publications.keys()),
          list(dct_china_year_publications.keys()),
          list(dct_usa_year_publications.keys())]

y_data = [list(dct_india_year_publications.values()),
          list(dct_china_year_publications.values()),
          list(dct_usa_year_publications.values())]
```

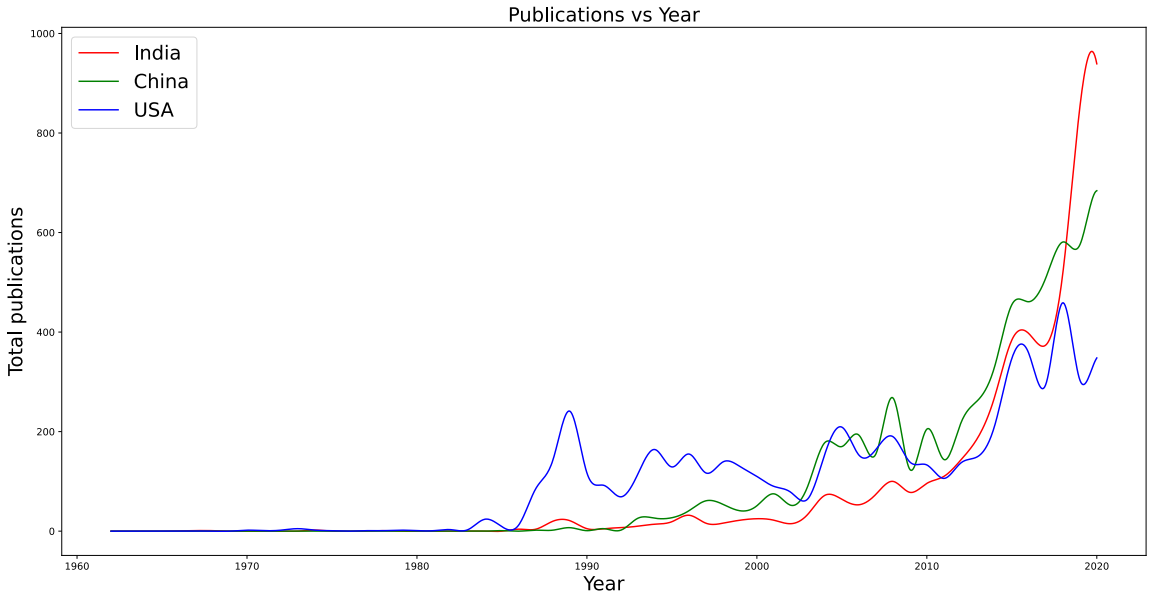
In [36]:
```
fig = plt.figure(figsize=[20,10])

plt.plot(x_data[0], y_data[0],    label='India',  color='r')
plt.plot(x_data[1], y_data[1],    label='China',  color='g')
plt.plot(x_data[2], y_data[2],    label='USA',    color='b')

plt.xlabel('Year',fontsize=20)
plt.ylabel('Total publications',fontsize=20)
plt.title('Publications vs Year',fontsize=20)
plt.legend(loc='upper left',fontsize=20)
plt.show()
```



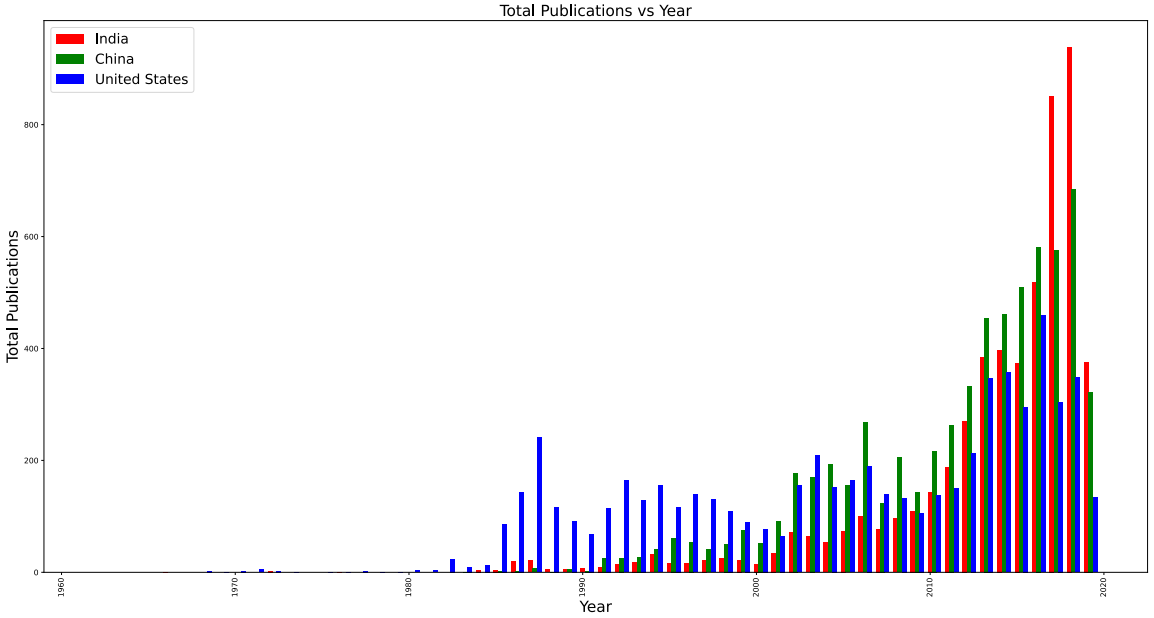
Generating a smoother curve using scipy.interpolate library

In [37]:
```
y_new = []
x_new = []
for i in range(3):
    x_new_tmp = np.linspace(year_lst[0],year_lst[0]+len(x_data[0]),1000)
    x_new.append(x_new_tmp)

    spline = interpolate.make_interp_spline(x_data[i], y_data[i])
    y_new.append(spline(x_new_tmp))
```

In [38]:
```
fig = plt.figure(figsize=[20,10])

plt.plot(x_new[0], y_new[0],    label='India',  color='r')
plt.plot(x_new[1], y_new[1],    label='China',  color='g')
plt.plot(x_new[2], y_new[2],    label='USA',    color='b')

plt.xlabel('Year',fontsize=20)
plt.ylabel('Total publications',fontsize=20)
plt.title('Publications vs Year',fontsize=20)
plt.legend(loc='upper left',fontsize=20)
plt.show()
```



In [39]:
```
# plt.rcParams['figure.figsize'] = [20,10]
fig = plt.figure(figsize=[20,10])

X = np.arange(len(year_lst))
X = X + year_lst[0]

fig = fig.add_axes([0,0,1,1])

fig.bar(X + 0.00, list(dct_india_year_publications.values()),  label='India',
color='r',  width =0.25)
fig.bar(X + 0.25, list(dct_china_year_publications.values()),  label='China',
color='g',  width =0.25)
fig.bar(X + 0.50, list(dct_usa_year_publications.values()),    label='United States',
color='b',  width =0.25)

fig.legend(loc='upper left',fontsize=18)

plt.xticks(rotation = 'vertical')

plt.title('Total Publications vs Year',fontsize=20)
plt.xlabel('Year',fontsize=20)
plt.ylabel('Total Publications',fontsize=20)
plt.show()
```



j) Total number of grants given to the field

In [40]:
```
grants, cols = df_without_duplicates[df_without_duplicates['Funding_Details']!='0'].s
hape
```

In [41]:
```
print(f'Grants given to field = {grants}')
```

```
Grants given to field = 4771
```

k) Country wise total number of publication

In [42]:
```
dct_country_publications = {}
for country in country_lst:
    rows, columns = df[df.Country==country].shape
    dct_country_publications[country] = rows
```

In [43]:
```
dct_country_publications = sort_dict(dct_country_publications,'Value','reverse')
```

In [44]:
```
dct_country_publications
```

Out[44]:
```
{'United Kingdom': 8994,
 'China': 6401,
 'United States': 6104,
 'India': 5383,
 'Germany': 5186,
 'Spain': 4759,
 'Canada': 4486,
 'Japan': 4324,
 'Italy': 4214,
 'France': 4133,
 'Australia': 3361,
 'South Korea': 3026,
 'Iran': 2720,
 'Taiwan': 2430,
 'Netherlands': 2173}
```