INTEGRATED PROJECT REPORT

On

Citation

Submitted in partial fulfillment of the requirement for the
Course Integrated Project (CS 203) of

COMPUTER SCIENCE AND ENGINEERING
B.E. Batch-2019

in

JUNE-2022

Submitted By

Under the Guidance of:
Dr. Deepak Ahlawat
Name of the Project Guide:
Citation
Designation of the Project Guide:
Assistant Professor

Shivangi Nanda
Roll. No. 1910991956
Nishchay Mahor
Roll. No. 1910991962
Ritagya Jain
Roll. No. 1910990288

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
CHITKARA UNIVERSITY
PUNJAB

INTEGRATED PROJECT REPORT

On

Citation

Submitted in partial fulfillment of the requirement for the
Course Integrated Project (CS 203) of

COMPUTER SCIENCE AND ENGINEERING
Batch-2019

in

JUNE-2022

Submitted By

Shivangi Nanda
Roll. No. 1910991956
Nishchay Mahor
Roll. No. 1910991962
Ritagya Jain
Roll. No. 1910990288

Under the Guidance of:
Dr. Deepak Ahlawat
Name of the Project Guide:
Citation
Designation of the Project Guide:
Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
CHITKARA UNIVERSITY
PUNJAB

CERTIFICATE

This is to be certified that the project entitled "Citation" has been submitted for the Bachelor of Computer Science Engineering at Chitkara University, Punjab during the academic semester January 2022- May-2022 is a bonafide piece of project work carried out by Shivangi Nanda (1910991956), Nishchay Mahor (1910991962), Ritagya Jain (1910990288) towards the partial fulfillment for the award of the course Integrated Project (CS 203) under the guidance of Dr. Deepak Ahlawat and supervision.

Sign. of Project Guide:
Dr Deepak Ahlawat
Assistant professor

CANDIDATE'S DECLARATION

We, **Shivangi Nanda (1910991956), Nishchay Mahor (1910991962), Ritagya Jain (1910990288)**, B.E.-2019 of the Chitkara University, Punjab hereby declare that the Integrated Project Report entitled **"Citation"** is an original work and data provided in the study is authentic to the best of our knowledge. This report has not been submitted to any other Institute for the award of any other course.

Sign. of Student 1	**Sign. of Student 2**	**Sign. of Student 3**
Shivangi Nanda	Nishchay Mahor	Ritagya Jain
1910991956	1910991962	1910990288

Place: Chitkara University
Date: 27th June, 2022

ACKNOWLEDGEMENT

It is our pleasure to be indebted to various people, who directly or indirectly contributed in the development of this work and who influenced my thinking, behavior and acts during the course of study.

We express our sincere gratitude to all for providing me an opportunity to undergo Integrated Project as the part of the curriculum.

We are thankful to Dr. Deepak Ahlawat for his support, cooperation, and motivation provided to us during the training for constant inspiration, presence and blessings.

We also extend our sincere appreciation to Dr. Deepak Ahlawat who provided his valuable suggestions and precious time in accomplishing our Integrated project report.

Lastly, we would like to thank the almighty and our parents for their moral support and friends with whom we shared our day-to day experience and received lots of suggestions that improve our quality of work.

Shivangi Nanda	**Nishchay Mahor**	**Ritagya Jain**
1910991956	**1910991962**	**1910990288**

Content

Abstract

Researchers are expected to find previous literature that is related to their research and potentially has a scientific impact from among a large number of publications. Some of the hallmarks of good research include attention to detail and the ability to discern patterns and make connections. Good citation practices can help with both. Predicting the impact of academic papers can help scholars quickly identify the high-quality papers in the field. Becoming detail-oriented in one aspect automatically instils good habits across the board in your research. As for the ability to spot trends and patterns, preparing a good bibliography train you for this task because of the vast amount of information it condenses into a short space. Providing accurate citations puts your work and ideas into an academic context. They tell your reader that you've done your research and know what others have said about your topic. The bibliometric criterion for judging the impact of biomedical papers and their authors' work is the number of citations received which is commonly referred to as "citation count". we develop efficient predictive model for evaluating potential papers has attracted increasing attention in academia. Many studies have shown that early citations contribute to improving the performance of predicting the long-term impact of a paper. We use deep learning techniques to encode the text, and to predict the impact of the citation used for individual articles using Title of the article, Publication year and Name of senior author. Natural Language Processing pipeline is used to process the title and author list of the articles. Logistic regression and linear regression are used to predict citation metrics based on article title and author list.

Introduction

A citation is a reference to a source. More precisely, a citation is an abbreviated alphanumeric expression embedded in the body of an intellectual work that denotes an entry in the bibliographic references section of the work for the purpose of acknowledging the relevance of the works of others to the topic of discussion at the spot where the citation appears.

Citation makes you a better researcher. The concept of Citation represents the author's effort as well as the detailed research work. An author puts his best in developing his writing and citation gives a space to him to show their attempt clearly. To avoid plagiarism by quoting words and ideas used by other authors. Citation impact quantifies the citation usage of scholarly works.

Scientific citation is providing detailed reference in a scientific publication, typically a paper or book, to previous published (or occasionally private) communications which have a bearing on the subject of the new publication. The purpose of citations in original work is to allow readers of the paper to refer to cited work to assist them in judging the new work, source background information vital for future development, and acknowledge the contributions of earlier workers. Citations in, say, a review paper bring together many sources, often recent, in one place.

To a considerable extent the quality of work, in the absence of other criteria, is judged on the number of citations received, adjusting for the volume of work in the relevant topic. While this is not necessarily a reliable measure, counting citations is trivially easy; judging the merit of complex work can be very difficult.

Reference section in scientific paper.

Citation impact quantifies the citation usage of scholarly works

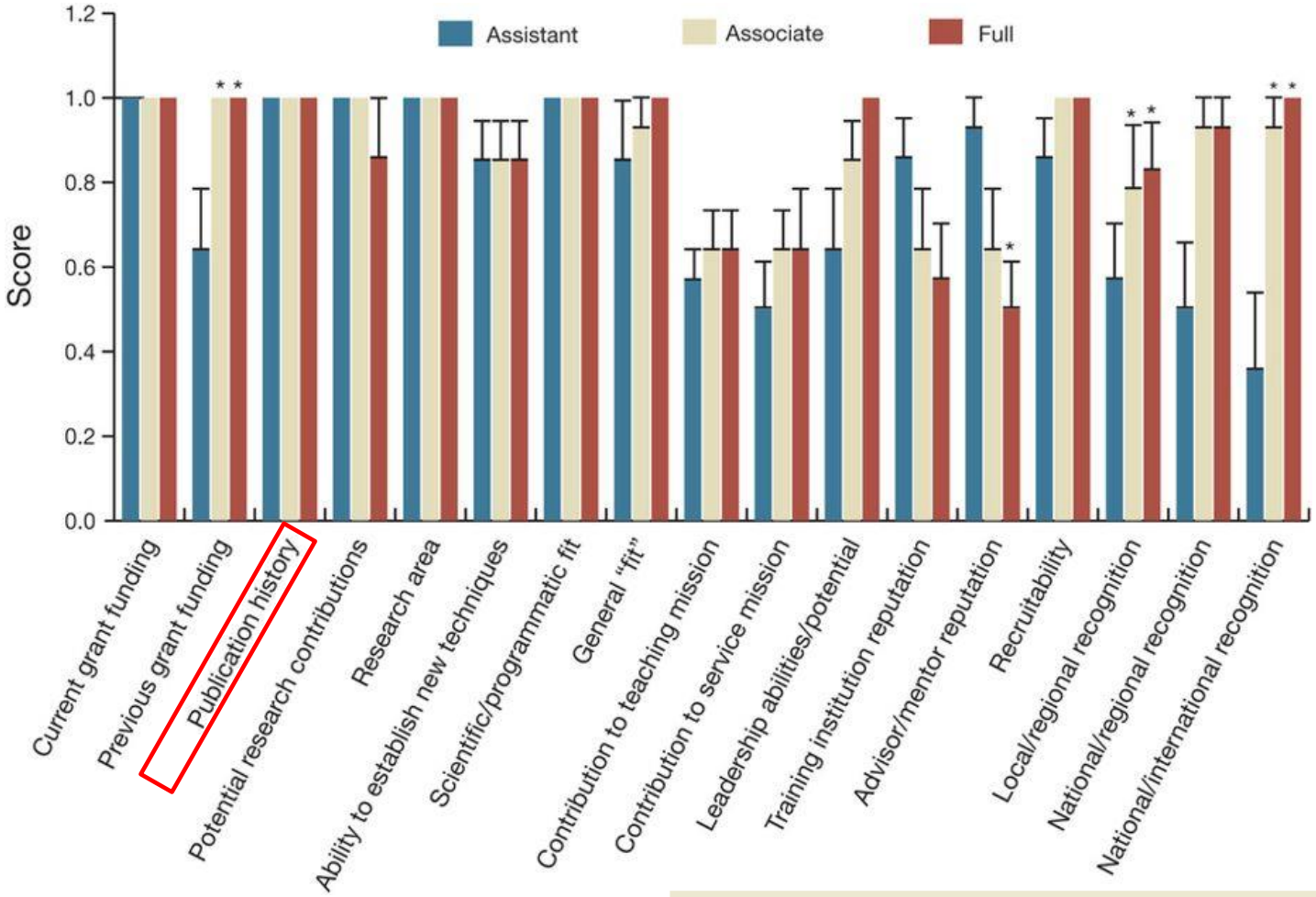
Citation counts for an author, an individual article, or a publication

Background

What faculty hiring committees want

Charles B Wright & Nathan L Vanderford

PhD trainees aspiring to become faculty need to know the credentials search committees value most in an applicant.



Research students are required to write journals, research papers and showcase their work and Researchers are expected to find previous literature that is related to their research and potentially has a scientific impact from among a large number of publications. The bibliometric criterion for judging the impact of biomedical papers and their authors' work is the number of citations received which is commonly referred to as "Citation count". Many studies have shown that early citations contribute to improving the performance of predicting the long-term impact of a paper.

Table 1 Faculty credentials fall into four categories

	Assistant	Associate	Full
Core competencies			
Current grant funding	1.00	1.00	1.00
Publication history	1.00	1.00	1.00
Research area	1.00	1.00	1.00
Scientific/programmatic fit	1.00	1.00	1.00
Potential research contributions	1.00	1.00	0.86
Ability to establish new techniques	0.86	0.86	0.86
General fit	0.86	0.93	1.00
Recruitability	0.86	1.00	1.00
Initial necessities			
Advisor/mentor reputation	0.93	0.64	0.50
Training institution reputation	0.86	0.64	0.57
Necessities for advancement			
National/international recognition	0.36	0.93	1.00
Regional/national recognition	0.50	0.93	0.93
Previous grant funding	0.64	1.00	1.00
Leadership abilities/potential	0.64	0.86	1.00
Unnecessary credentials			
Contribution to service mission	0.50	0.64	0.64
Contribution to teaching mission	0.57	0.64	0.64
Local/regional recognition	0.57	0.79	0.83

Credential scores followed one of four distinct trends: core competencies, initial necessities, necessities for advancement, and unnecessary credentials.

Problem Statement

A commonly accepted metric for evaluating the impact and quality of an article is the *citation count* which is the number of citations received by this article within a pre-specified time horizon. A "citation" indicates your readers that certain material in your work came from certain another source, and refers to how many times a certain publication has been referred in some other publication.
It is an important factor to estimate the relevance and significance of academic publications. In its essence the quality of an article is assessed by the number of times other authors mention it in their own works. So naturally more the citation count, the more a certain publication contains legitimate knowledge and that number of people trust that information to an extent where they utilized that information in their own works hence better the article.

Thousands of academics are published at least once every five days. While some of these publications attract attention and receive a high number of citations, others are scarcely cited, if at all. To the best of the author's knowledge, no analysis has been conducted to determine the effectiveness of different publication features with the advances in machine learning (ML), we can apply existing methodologies to uncover hidden relationships between all publication features and investigate the importance of features and relevant intercorrelations.

Many studies have shown that early citations contribute to improving the performance of prediction the long-term impact of a paper. So how impactful their work will be?

Software and Hardware requirement specification

Material and Methods

To ensure a generic and unbiased conclusion, articles are included in the analysis based only on citations, without consideration of the article type; the journal's impact factor, chronology, or discipline; or any other categorization.

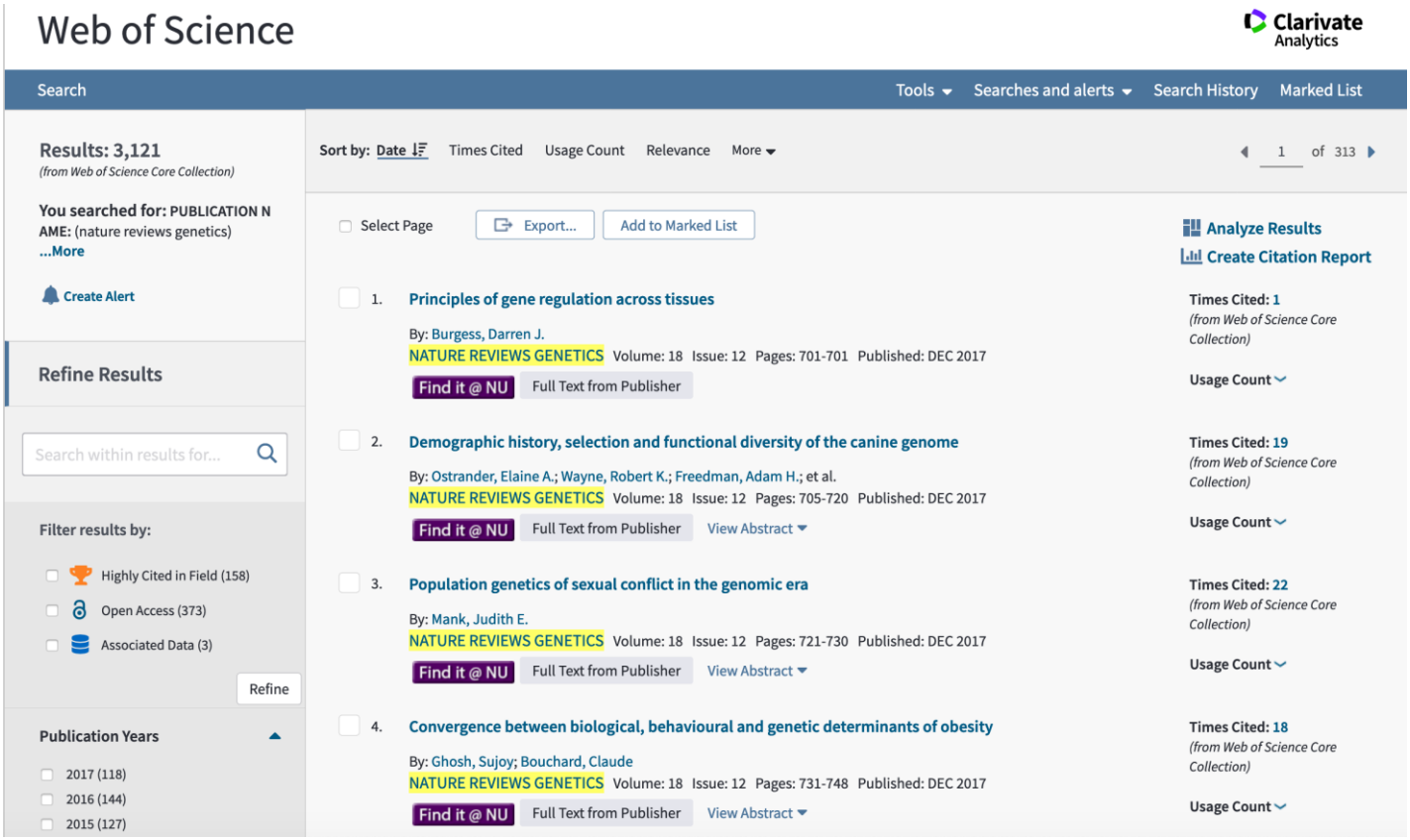
Dataset for Our Project

In order to develop a solid conclusion with quantifiable values regarding title length, the top 100 citations from three indexes of citation databases were analyzed. The databases are described below:

- Google Scholar: This database is publicly available. Approximately two-thirds of the database are books from all disciplines, and the database contains various cited papers. Only the 100 research publications with the most citations were used for this analysis. The number of citations for the 100 articles in this database ranges from 30,948 to 223,131.
- Web of Science: This database is publicly available and contains various cited papers. Only the 100 research publications with the most citations were used for this analysis. The database covers all of Thomson Reuter's Web of Science and includes works on the social sciences and arts and humanities, conference proceedings, and some books published since 1900. The number of citations for the 100 articles in this database ranges from 12,119 to 305,148.
- Altmetric: This database is publicly available and contains the 2018 Altimetric Top 100, which is an annual list of the research that most captured the public's attention last year. Only the 100 research publications with the most citations were used for this analysis. This list includes research published from 2013 to 2018. Individuals from various industries, publishers, and academic institutions are recently looking at the Altmetric score more with interest, which is why it is included in this study.
- For ease of access, we have accessed a third-party website named as Nature Reviews Genetics website which combines data from other websites. We have included 3121 articles published between 2000 to 2017.

Websites that generate citation metrics

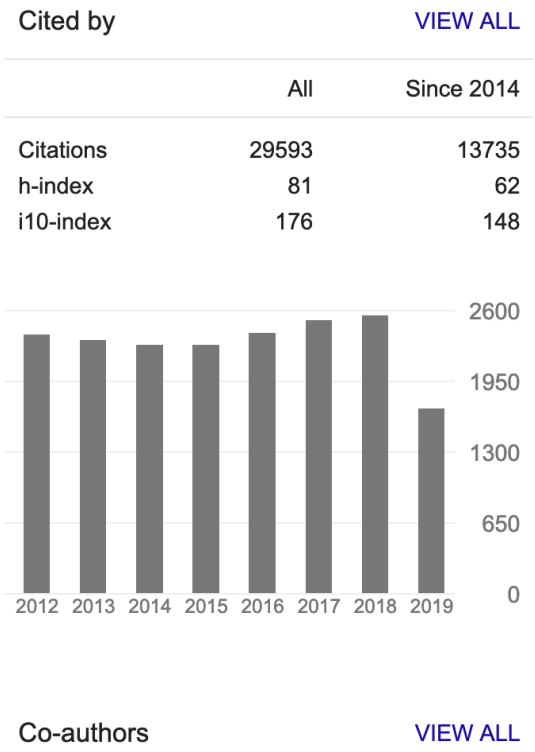
Web of Science is a website that provides access based on database that comprises comprehensive citation data of different academic fields. It is one of the best citation websites known in the field of science and technology.



Results can be downloaded as an .xls file

For our analyses, this was reformatted as a csv and cleaned up with pandas

Visualizations were generated with matplotlib



Data Source:

- 3121 articles published between 2000 to 2017
- Source: Nature Reviews Genetics website

nature REVIEWS GENETICS

InCites Journal Citation Reports — Clarivate Analytics

Home > Journal Profile

NATURE REVIEWS GENETICS

ISSN: 1471-0056
eISSN: 1471-0064
NATURE PUBLISHING GROUP
MACMILLAN BUILDING, 4 CRINAN ST, LONDON N1 9XW, ENGLAND
ENGLAND

TITLES
ISO: Nat. Rev. Genet.
JCR Abbrev: NAT REV GENET

CATEGORIES
GENETICS & HEREDITY - SCIE

LANGUAGES
English

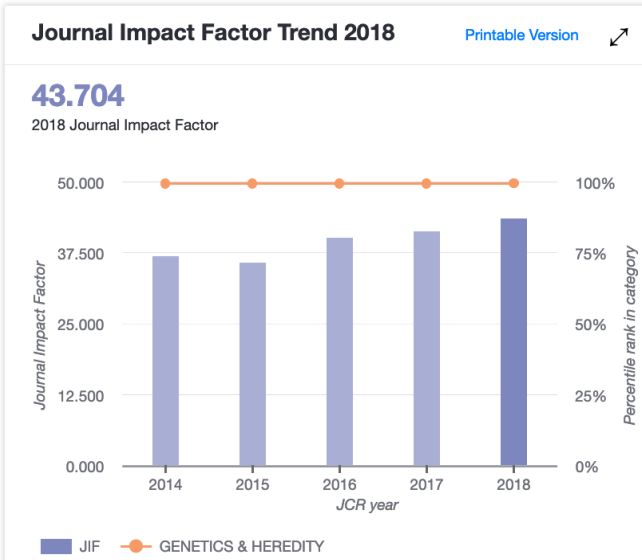
PUBLICATION FREQUENCY
12 issues/year

Go to Journal Table of Contents Printable Version

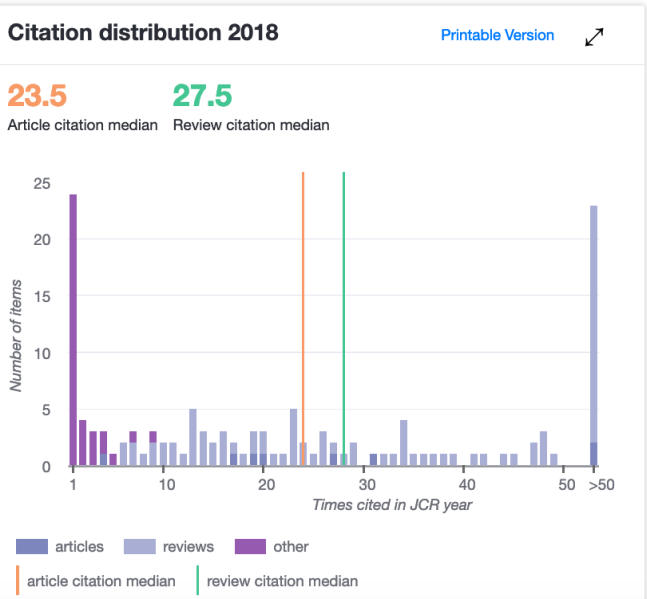
Current Year 2017 All Years

The data in the two graphs below and in the Journal Impact Factor calculation panels represent citation activity in 2018 to items published in the journal in the prior two years. They detail the components of the Journal Impact Factor. Use the "All Years" tab to access key metrics and additional data for the current year and all prior years for this journal.

Journal Impact Factor Trend 2018 Printable Version



Citation distribution 2018 Printable Version



How many times will my article be cited?

…if it is published in <insert favorite journal here>

nature REVIEWS GENETICS

Methods

1 Collection of data: Extracting Labelled data.

2 Data Pre-processing: The majority of the functions are in the pre-processing stage since the raw text cannot be directly inputted into a ML algorithm. The input data needs to go through a number of processes. The processes, which include tokenization, removal of stop words, and clean data for feature extraction, and eventually, model training.

3 Removing Stopwords: These are the words that are used in any language used to connect words or used to declare the tense of sentences. This means that if we use these words in any sentence, they do not add much meaning to the context of the sentence so even after removing the stopwords we can understand the context.

3 Tf-Idf Vectorizer is a common algorithm to transform text into meaningful representation of numbers. It is used to extract features from text strings based on occurrence. We assume that higher number of repetitions of a word would mean greater importance in the given text.

4 Feature extraction: Now, the raw text data has undergone certain functions for it to be cleaned of any misleading features or useless factors. However, the ML algorithm will still not understand text since it's a string. Thus, the text needs to be converted to vectors of numbers. This is called feature extraction or feature encoding.

5 Model training: Many different classification ML algorithms could be used to classify a piece of text.

6 Logistic Regression: This algorithm is fit for problems that have a binary output of 0 or 1, true or false.

Programming/Working Environment:

Building a cohesive team can help make projects more unified and improve productivity.
Working with a team is a different experience, we get to learn new concepts, ideologies and analyze different aspects and combining that to make a solution to solve real world problems. We worked together as a team and came up with an innovative idea and explored new topics as to get a positive outcome.
We worked on gaining new skills and inculcated those in our projects such as Natural Language Process, Count vectorizer. All in all it was a great experience to work together with like-minded, hard work driven people and succeeded in making this project.

Requirements to Run the application:

Hardware Requirement: Ram 8GB, Storage 500 GB, CPU 2 GHz or faster, Architecture 64-bit, Processor i5 10th Generation.
Software Requirement: Python 3.7 in Jupyter Notebook, used for importing text file, data pre- processing, model training and prediction.
Operating System: Windows 7 or above Or Linux based OS Or Mac OS.
GitHub - it is a code hosting platform for version control and collaboration. It lets you and others work together on projects from anywhere.
GitHub Link

Libraries Imported

```
import pandas as pd
```

Python Pandas is defined as an open-source library that provides high-performance data manipulation in Python. Data analysis requires lots of processing, such as restructuring, cleaning or merging, etc. There are different tools are available for fast data processing, such as Numpy, Scipy, Cython, and Panda. But we prefer Pandas because working with Pandas is fast, simple and more expressive than other tools. Pandas is built on top of the Numpy package, means Numpy is required for operating the Pandas. It has a fast and efficient DataFrame object with the default and customized indexing. Used for reshaping and pivoting of the data sets. Group by data for aggregations and transformations. It is used for data alignment and integration of the missing data. Provide the functionality of Time Series. Process a variety of data sets in different formats like matrix data, tabular heterogeneous, time series. Handle multiple operations of the data sets such as subsetting, slicing, filtering, groupBy, re-ordering, and re-shaping. It integrates with the other libraries such as SciPy, and scikit-learn. Provides fast performance, and If you want to speed it, even more, you can use the Cython.

```
import numpy as np
```

NumPy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, fourier transform, and matrices. In Python we have lists that serve the purpose of arrays, but they are slow to process. NumPy aims to provide an array object that is up to 50x faster than traditional Python lists. The array object in NumPy is called ndarray, it provides a lot of supporting functions that make working with ndarray very easy. Arrays are very frequently used in data science, where speed and resources are very important. NumPy arrays are stored at one continuous place in memory unlike lists, so processes can access and manipulate them very efficiently. This behavior is called locality of reference in computer science. This is the main reason why NumPy is faster than lists. Also, it is optimized to work with latest CPU architectures.

```
import json
```

The json module provides an API similar to convert in-memory Python objects to a serialized representation known as JavaScript Object Notation (JSON) and vice-a-versa.

```
import nltk
```

The Natural Language Toolkit (NLTK) is a platform used for building Python programs that work with human language data for applying in statistical natural language processing (NLP).
It contains text processing libraries for tokenization, parsing, classification, stemming, tagging and semantic reasoning. It also includes graphical demonstrations and sample data sets as well as accompanied by a cook book and a book which explains the principles behind the underlying language processing tasks that NLTK supports.

```
import re
```

Regular expression or RegEx in Python is denoted as RE (REs, regexes or regex pattern) are imported through re module. Python supports regular expression through libraries. RegEx in Python supports various things like Modifiers, Identifiers, and White space characters. Identifiers

```
import csv
```

The csv module implements classes to read and write tabular data in CSV format. It allows programmers to say, “write this data in the format preferred by Excel,” or “read data from this file which was generated by Excel,” without knowing the precise details of the CSV format used by Excel.

```
import matplotlib.pyplot as plt
```

Matplotlib is a cross-platform, data visualization and graphical plotting library for Python and its numerical extension NumPy. Pyplot is a plotting library used for 2D graphics in python programming language. It can be used in python scripts, shell, web application servers and other graphical user interface toolkits. Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib makes easy things easy and hard things possible.

When analysts and data scientists use matplotlib, they're usually using it in tandem with other Python libraries. Matplotlib is designed to work with NumPy, a numerical mathematics library and is a core part of the SciPy stack—a group of scientific computing tools for Python.

Some libraries like pandas and Seaborn are “wrappers” over matplotlib—they allow you to access a number of matplotlib's methods with less code. For instance, pandas. plot () combines multiple matplotlib methods into a single method, so you can plot a chart in a few lines.
The large amount of code required in matplotlib to generate a nice-looking plot is often its biggest criticism. As a result, other visualization libraries like Seaborn, Bokeh, and plotly have emerged.

```
import seaborn as sns
```

Seaborn: Seaborn is a Python visualization library for statistical plotting. It comes equipped with preset styles and color palettes so you can create complex, aesthetically pleasing charts with a few lines of code. It's designed to work with NumPy and pandas.) data structures and to support statistical tasks completed in SciPy and statsmodels.

Seaborn is built on top of Python's core visualization library matplotlib, but it's meant to serve as a complement, not a replacement. In most cases, you'll still use matplotlib for simple plotting, and you'll need a knowledge of matplotlib to tweak Seaborn's default plots.

```
from tqdm import tqdm
```

Tqdm: tqdm is a library in Python which is used for creating Progress Meters or Progress Bars.

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
```

Sklearn: Scikit-learn is probably the most useful library for machine learning in Python. The Sklearn library contains a lot of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction. Scikit-learn comes loaded with a lot of features including: Supervised learning algorithms, Cross-validation, Unsupervised learning algorithms, Various toy datasets, Feature extraction.

Dataset

	Title	Authors	Publication Year	Total Citations	Average per Year	1997	1998	1999	2000	2001	...	2008	2009	2010	2011	2012
0	RNA-Seq: a revolutionary tool for transcriptomics	Wang, Zhong; Gerstein, Mark; Snyder, Michael	2009	5424	493.09	0	0	0	0	0	...	0	69	223	341	469
1	Network biology: Understanding the cell's func...	Barabasi, AL; Oltvai, ZN	2004	4202	262.63	0	0	0	0	0	...	248	326	285	316	359
2	The fundamental role of epigenetic events in c...	Jones, PA; Baylin, SB	2002	3855	214.17	0	0	0	0	0	...	262	263	284	292	270
3	APPLICATIONS OF NEXT-GENERATION SEQUENCING Seq...	Metzker, Michael L.	2010	3395	339.50	0	0	0	0	0	...	0	0	148	343	444
4	Microrna: Small RNAs with a big role in gene ...	He, L; Hannon, GJ	2004	3343	208.94	0	0	0	0	0	...	119	144	177	206	265

```
count    3121.000000
mean      115.244473
std       284.517875
min         0.000000
25%         0.000000
50%         2.000000
75%       130.000000
max      5424.000000
Name: Total Citations, dtype: float64
```

Our dataset comprises of Title of the journal, Author's Name of the respective Journal, year in which the Journal was Publicized, Total Citations, Average Citations per Year, and Count of Citations each year.

Our goal is to predict whether a certain Journal with a certain "Title" will it be impactful if a researcher cites a Journal in their Research paper.

Similarly which Author's Journal are more likely to be impactful and enhance your findings and make the research paper more impactful.

Cleaning the Data

```
def clean_text(text):
    # remove backslash-apostrophe
    text = re.sub("\'", "", text)
    # remove everything except alphabets
    text = re.sub("[^a-zA-Z]"," ",text)
    # remove whitespaces
    text = ' '.join(text.split())
    # convert text to lowercase
    text = text.lower()

    return text
```

Tiding up the Titles by:

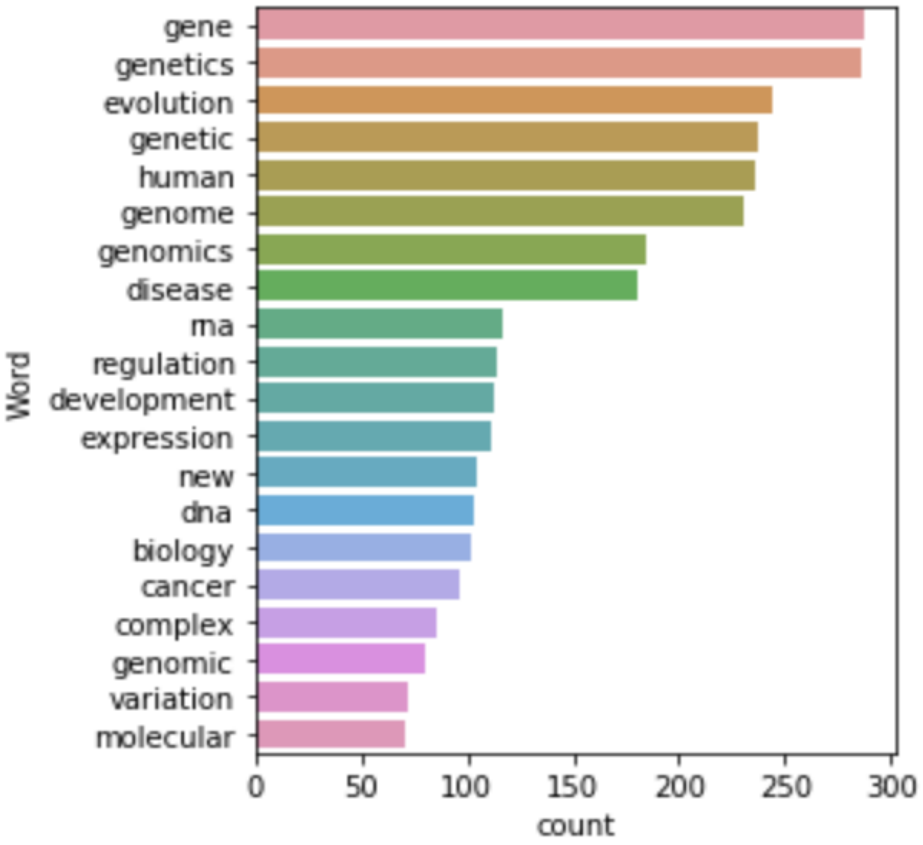
- ✓ Removing Backlash-Apostrophe
- ✓ Removing characters except Alphabets
- ✓ Removing Spaces

Title	cleaned_title
RNA-Seq: a revolutionary tool for transcriptomics	rna seq a revolutionary tool for transcriptomics
Network biology: Understanding the cell's func...	network biology understanding the cells functi...
The fundamental role of epigenetic events in c...	the fundamental role of epigenetic events in c...
APPLICATIONS OF NEXT-GENERATION SEQUENCING Seq...	applications of next generation sequencing seq...
Micrornas: Small RNAs with a big role in gene ...	micrornas small rnas with a big role in gene r...

Determine most-frequent words

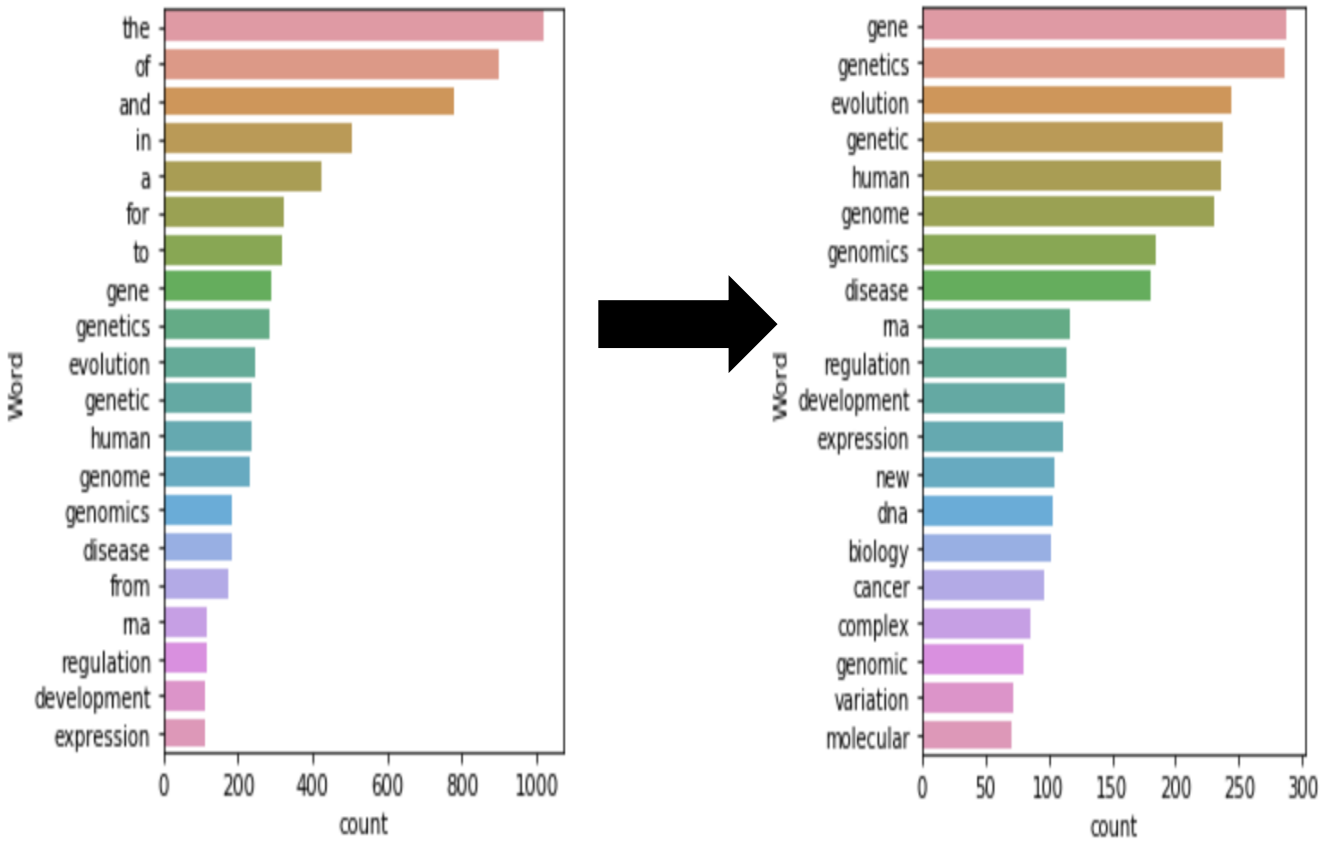
```
def freq_words(x, terms = 30):
  all_words = ' '.join([text for text in x])
  all_words = all_words.split()
  fdist = nltk.FreqDist(all_words)
  words_df = pd.DataFrame({'word':list(fdist.keys()), 'count':list(fdist.values())})

  # selecting top 20 most frequent words
  d = words_df.nlargest(columns="count", n = terms)

  # visualize words and frequencies
  plt.figure(figsize=(4,5))
  ax = sns.barplot(data=d, x= "count", y = "word")
  ax.set(ylabel = 'Word')
  plt.show()

# print 100 most frequent words
freq_words(df['cleaned_title'],20)
```

- ✓ Import Counter class from collections module.
- ✓ Split the string into list using split (), it will return the lists of words.
- ✓ Now pass the list to the instance of Counter class.
- ✓ The function 'most-common ()' inside Counter will return the list of most frequent words from list and its count.



Stopwords Filtering

A stop word is a commonly used word (such as “the”, “a”, “an”, “in”) that a search engine has been programmed to ignore, both when indexing entries for searching and when retrieving them as the result of a search query.
We would not want these words to take up space in our database, or taking up valuable processing time. For this, we can remove them easily, by storing a list of words that you consider to stop words. NLTK (Natural Language Toolkit) in python has a list of stopwords stored in 16 different languages.

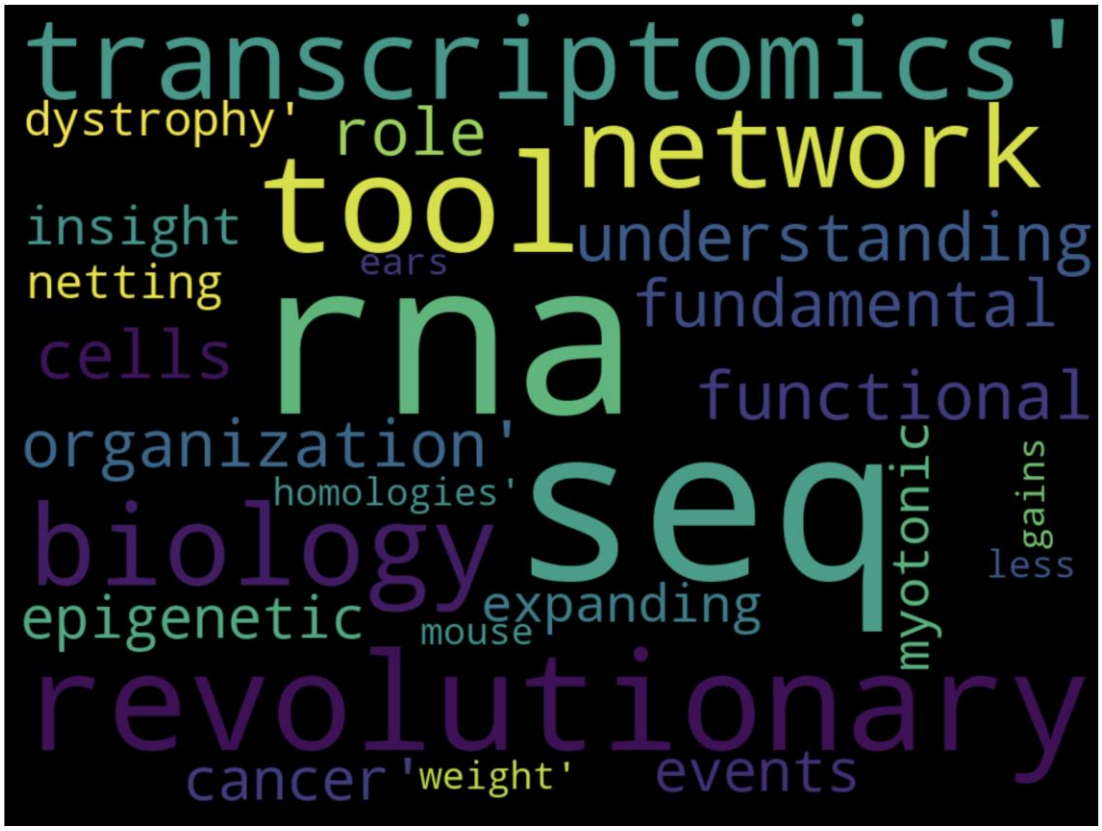


By removing these words, we remove the low-level information from our text in order to give more focus to the important information.

Most frequent words

```
from wordcloud import WordCloud, STOPWORDS
text = df.cleaned_title.values
wordcloud = WordCloud(
    width = 700,
    height = 700,
    max_font_size = 200,
    background_color = 'black',
    stopwords = STOPWORDS).generate(str(text))
fig = plt.figure(
    figsize = (40, 30),
    facecolor = 'k',
    edgecolor = 'k')
plt.imshow(wordcloud, interpolation = 'bilinear')
plt.axis('off')
plt.tight_layout(pad=0)
plt.show()
```

A word cloud (also known as a tag cloud) is a visual representation of words. Cloud creators are used to highlight popular words and phrases based on frequency and relevance. They provide you with quick and simple visual insights that can lead to more in-depth analyses.



Most Frequent words in the "Title" of the Journal

Created New Column

1. Cleaned up the text in “Title”
2. Removed Stop Words
3. Created a new column and named “finalcleaned_title”
4. Created a new Column Category to know if a particular Journal’s Title Total Citation count is above median total citation count, then it is categorized as “Top_Half” and if Title’s Total Citation count is below median total citation count, then it will be categorized as “Bottom_half”.

$$IF\ for\ year\ X\ = \frac{Citations\ in\ X\ to\ articles\ published\ in\ X-1\ and\ X-2}{Articles\ published\ in\ X-1\ and\ X-2}$$

	finalcleaned_title	Average per Year	Total Citations	Category
0	rna seq revolutionary tool transcriptomics	493.09	5424	Top_half
1	network biology understanding cells functional...	262.63	4202	Top_half
2	fundamental role epigenetic events cancer	214.17	3855	Top_half
3	applications next generation sequencing sequen...	339.50	3395	Top_half
4	micrornas small rnas big role gene regulation	208.94	3343	Top_half

TF-IDF vs. Count Vectorizer

TF-IDF Vectorizer weights down common words, gives more importance to rare words
Count Vectorizer converts a collection of text to a matrix of the counts of occurrences of each word in the document.

```
from sklearn.feature_extraction.text import TfidfVectorizer
# initalise the vectoriser
tvec = TfidfVectorizer()
# fit the training data on the model
tvec.fit(X_train)

#transform training data into sparse matrix
X_train_tvec = tvec.transform(X_train)

# cross val score/ predict
tvec_score = cross_val_score(lr, X_train_tvec, y_train, cv=3)
```

```
from sklearn.feature_extraction.text import CountVectorizer
# initalise the vectoriser
cvec = CountVectorizer()
# fit the training data on the model
cvec.fit(X_train)

#transform training data into sparse matrix
X_train_cvec = cvec.transform(X_train)

# cross val score/ predict
cvec_score = cross_val_score(lr, X_train_cvec, y_train, cv=3 )
X_train_cvec
```

```
TF-IDF Vectorizer Score: 0.714198009803197
Count Vectorizer Score: 0.7234898027045001
```

	params	scores
0	Count	0.723490
1	TF-IDF	0.714198

K-Fold Cross Validation:

We used Cross-validation which is primarily used in applied machine learning to estimate the skill of a machine learning model on unseen data.

The basic idea in calculating cross validation error is to divide up training data into k-folds.

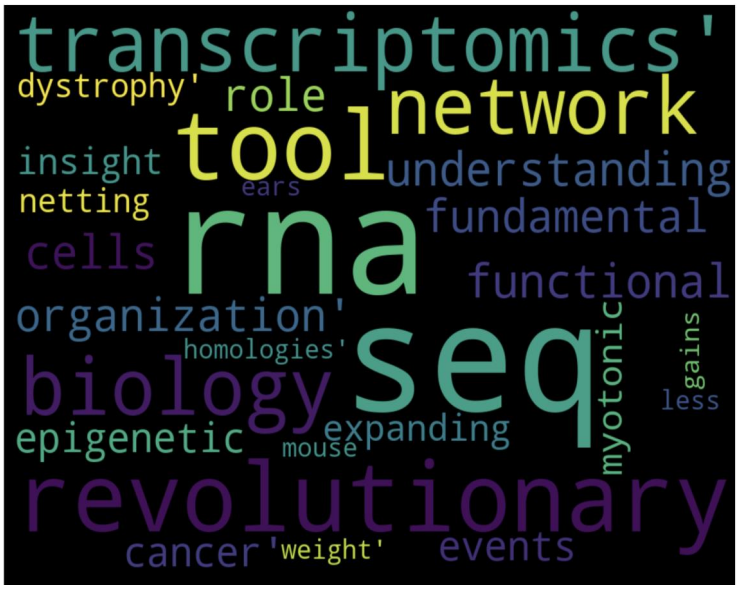
Each fold will then be held out one at a time, the model will be trained on the remaining data, and that model will then be used to predict the target for the holdout observations.

$$R^2 = 1 - \frac{\frac{1}{n}\sum_i |y_i - \hat{y}_i|^2}{\frac{1}{n}\sum_i |y_i - \bar{y}_{train}|^2},$$

Cross Validation ensures that every observation from the original dataset has the chance of appearing in training and test set. This is one among the best approach if we have a limited input data.

For “Titles” we are using TF-IDF vectorizer as it not only focuses on the frequency of words present in the corpus but also provides the importance of the words. We can then remove the words that are less important for analysis, hence making the model building less complex by reducing the input dimensions.

TF-IDF is used due to frequent appearance of words such as “RNA” and “Seq” in a genomics journal



Predict citation based on title using logistic regression

Train Test Split

```
tfidf_vectorizer = TfidfVectorizer(max_df=0.8, max_features=1000)
xtrain, xval, ytrain, yval = train_test_split(traindf['finalcleaned_title'], traindf['Category'], test_size=0.2, random_stat
```

Split the dataset into two pieces: a training set and a testing set. This consists of random sampling without replacement about 80% (you can vary this) of the rows and putting them into your training set and putting the remaining 20% to your test set. Note that the colors in “Features” and “Target” indicate where their data will go (“xtrain”, “xval”, “ytrain”, “yval”) for a particular train test split.
Train the model on the training set. This is “xtrain” and “ytrain”.
Test the model on the testing set (“xtest” and “ytest”) and evaluate the performance.

Logistic Regression with TF-IDF

```
tfidf_vectorizer = TfidfVectorizer(max_df=0.8, max_features=1000)
xtrain, xval, ytrain, yval = train_test_split(traindf['finalcleaned_title'], traindf['Categ

# create TF-IDF features
xtrain_tfidf = tfidf_vectorizer.fit_transform(xtrain)
xval_tfidf = tfidf_vectorizer.transform(xval)

from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression()
classifier.fit(xtrain_tfidf, ytrain)
print(f"Training Data Score: {classifier.score(xtrain_tfidf, ytrain)}")
print(f"Testing Data Score: {classifier.score(xval_tfidf, yval)}")
```

```
Training Data Score: 0.8441506410256411
Testing Data Score: 0.7472
```

After created a 80/20 train-test split of the dataset, I've applied logistic regression which is a classification algorithm used to solve binary classification problems. The logistic regression classifier uses the weighted combination of the input features and passes them through a sigmoid function. Sigmoid function transforms any real number input, to a number between 0 and 1.
We have applied logistic regression classifier, on Tf-Idf features and got classifier score on Training Data and Testing Data.

Prediction

	Predictions	Actual
0	Top_half	Top_half
1	Top_half	Top_half
2	Bottom_half	Bottom_half
3	Top_half	Bottom_half
4	Bottom_half	Bottom_half
5	Bottom_half	Bottom_half
6	Bottom_half	Bottom_half
7	Top_half	Top_half
8	Top_half	Top_half
9	Top_half	Top_half

Prediction for Validation set. We randomly chose 10 Rows and tested the classifier attaining 90% accuracy.

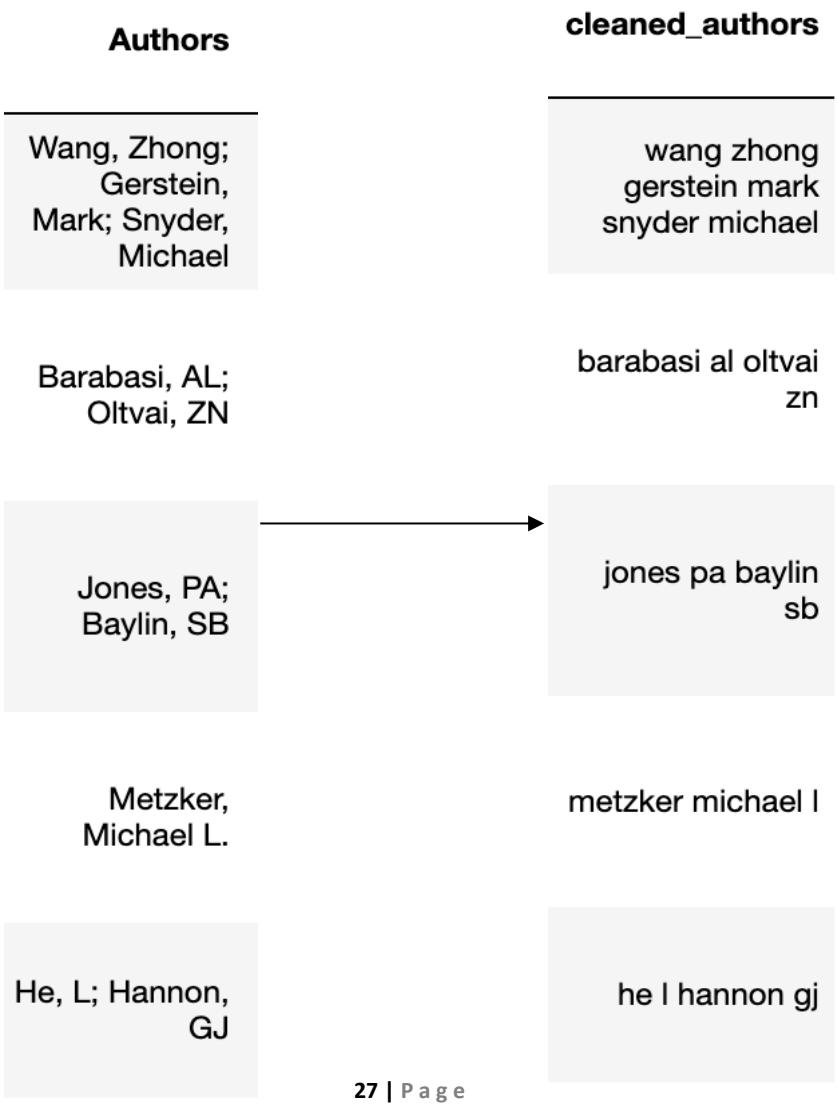
Journal's Author Name

Cleaning the Data

```
def clean_text(text):
    # remove backslash-apostrophe
    text = re.sub("\'", "", text)
    # remove everything except alphabets
    text = re.sub("[^a-zA-Z]"," ",text)
    # remove whitespaces
    text = ' '.join(text.split())
    # convert text to lowercase
    text = text.lower()

    return text
```

Tiding up the Author's Name by:

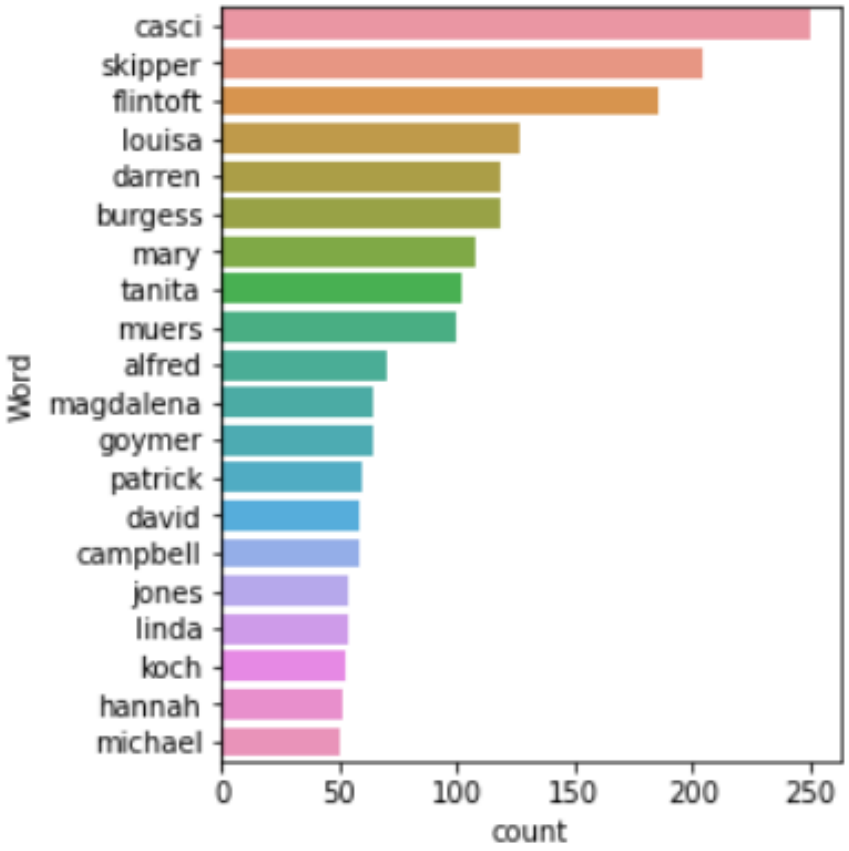
- ✓ Removing Backlash-Apostrophe
- ✓ Removing characters except Alphabets
- ✓ Removing Spaces



Determine most-frequent words

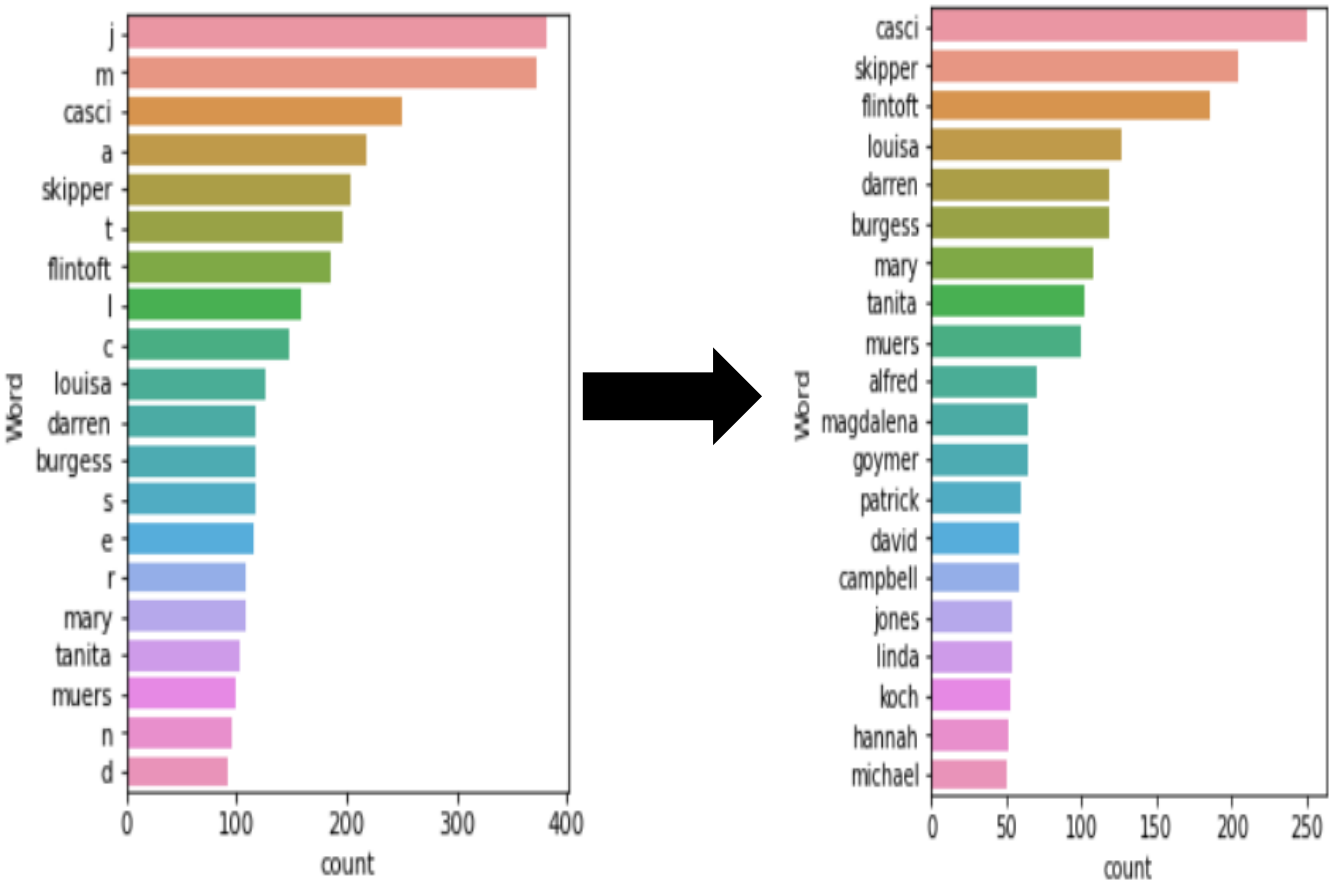
```
def freq_words(x, terms = 30):
  all_words = ' '.join([text for text in x])
  all_words = all_words.split()
  fdist = nltk.FreqDist(all_words)
  words_df = pd.DataFrame({'word':list(fdist.keys()), 'count':list(fdist.values())})

  # selecting top 20 most frequent words
  d = words_df.nlargest(columns="count", n = terms)

  # visualize words and frequencies
  plt.figure(figsize=(4,5))
  ax = sns.barplot(data=d, x= "count", y = "word")
  ax.set(ylabel = 'Word')
  plt.show()

# print 100 most frequent words
freq_words(df['cleaned_authors'],20)
```

- ✓ Import Counter class from collections module.
- ✓ Split the string into list using split (), it will return the lists of words.
- ✓ Now pass the list to the instance of Counter class.
- ✓ The function 'most-common ()' inside Counter will return the list of most frequent words from list and its count.



Stopwords Filtering

A stop word is a commonly used word (such as “the”, “a”, “an”, “in”) that a search engine has been programmed to ignore, both when indexing entries for searching and when retrieving them as the result of a search query.
We would not want these words to take up space in our database, or taking up valuable processing time. For this, we can remove them easily, by storing a list of words that you consider to stop words. NLTK (Natural Language Toolkit) in python has a list of stopwords stored in 16 different languages.

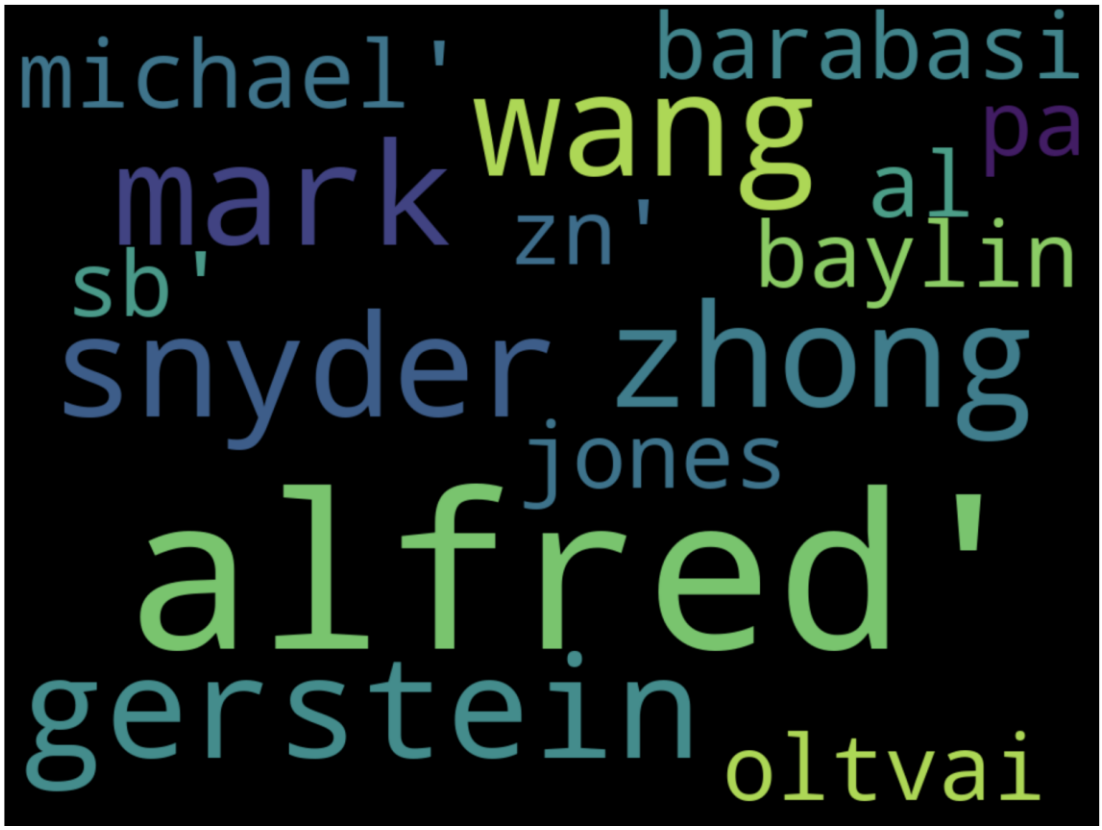


By removing these words, we remove the low-level information from our text in order to give more focus to the important information.

Most frequent words

```
from wordcloud import WordCloud, STOPWORDS
text = df.finalcleaned_authors.values
wordcloud = WordCloud(
    width = 700,
    height = 700,
    max_font_size = 200,
    background_color = 'black',
    stopwords = STOPWORDS).generate(str(text))
fig = plt.figure(
    figsize = (40, 30),
    facecolor = 'k',
    edgecolor = 'k')
plt.imshow(wordcloud, interpolation = 'bilinear')
plt.axis('off')
plt.tight_layout(pad=0)
plt.show()
```

A word cloud (also known as a tag cloud) is a visual representation of words. Cloud creators are used to highlight popular words and phrases based on frequency and relevance. They provide you with quick and simple visual insights that can lead to more in-depth analyses.



Most Frequent words in the “Author” of the Journal

Created New Column

1. Cleaned up the text in "Title"
2. Removed Stop Words
3. Created a new column and named "finalcleaned_title"
4. Created a new Column Category to know if a particular Journal's Title Total Citation count is above median total citation count, then it is categorized as "Top_Half" and if Title's Total Citation count is below median total citation count, then it will be categorized as "Bottom_half".

$$IF\ for\ year\ X\ = \frac{Citations\ in\ X\ to\ articles\ published\ in\ X-1\ and\ X-2}{Articles\ published\ in\ X-1\ and\ X-2}$$

	finalcleaned_authors	Average per Year	Total Citations	Category
0	wang zhong gerstein mark snyder michael	493.09	5424	Top_half
1	barabasi al oltvai zn	262.63	4202	Top_half
2	jones pa baylin sb	214.17	3855	Top_half
3	metzker michael	339.50	3395	Top_half
4	he hannon gj	208.94	3343	Top_half

TF-IDF vs. Count Vectorizer

TF-IDF Vectorizer weights down common words, gives more importance to rare words
Count Vectorizer converts a collection of text to a matrix of the counts of occurrences of each word in the document.

```
from sklearn.feature_extraction.text import TfidfVectorizer
# initalise the vectoriser
tvec = TfidfVectorizer()
# fit the training data on the model
tvec.fit(X_train)

#transform training data into sparse matrix
X_train_tvec = tvec.transform(X_train)

# cross val score/ predict
tvec_score = cross_val_score(lr, X_train_tvec, y_train, cv=3)
```

```python
from sklearn.feature_extraction.text import CountVectorizer
# initalise the vectoriser
cvec = CountVectorizer()
# fit the training data on the model
cvec.fit(X_train)

#transform training data into sparse matrix
X_train_cvec = cvec.transform(X_train)

# cross val score/ predict
cvec_score = cross_val_score(lr, X_train_cvec, y_train, cv=3 )
```

```
TF-IDF Vectorizer Score: 0.7767109411561862
Count Vectorizer Score: 0.8167510283504519
```

	params	scores
0	Count	0.816751
1	TF-IDF	0.776711

K-Fold Cross Validation:

We used Cross-validation which is primarily used in applied machine learning to estimate the skill of a machine learning model on unseen data.

The basic idea in calculating cross validation error is to divide up training data into k-folds.

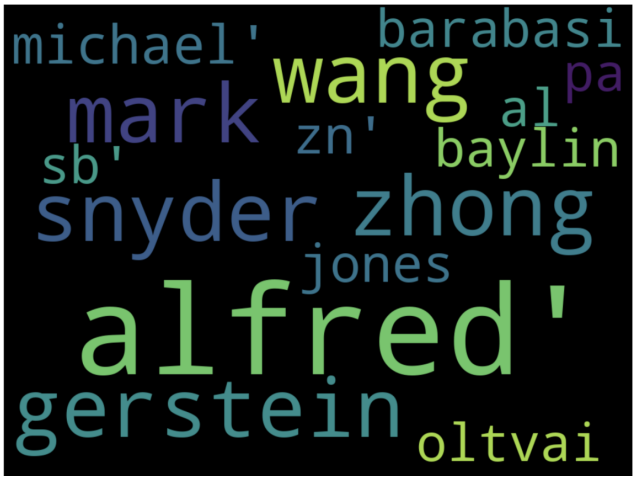
Each fold will then be held out one at a time, the model will be trained on the remaining data, and that model will then be used to predict the target for the holdout observations.

$$R^2 = 1 - \frac{\frac{1}{n}\sum_i |y_i - \hat{y}_i|^2}{\frac{1}{n}\sum_i |y_i - \bar{y}_{train}|^2},$$

Cross Validation ensures that every observation from the original dataset has the chance of appearing in training and test set. This is one among the best approach if we have a limited input data.

For “Authors” we are using Count Vectorizer to convert a collection of text documents to a vector of term/token counts. It also enables the pre-processing of text data prior to generating the vector representation. This functionality makes it a highly flexible feature representation module for text.

Count vectorizer is used because we don’t want to discard anyone’s name…



Predict citation based on title using logistic regression

Train Test Split

```
from sklearn.feature_extraction.text import CountVectorizer

xtrain, xval, ytrain, yval = train_test_split(traindf['finalcleaned_authors'], traindf['Category'], test_size=0.2, random_st

```

Split the dataset into two pieces: a training set and a testing set. This consists of random sampling without replacement about 80% (you can vary this) of the rows and putting them into your training set and putting the remaining 20% to your test set. Note that the colors in “Features” and “Target” indicate where their data will go (“xtrain”, “xval”, “ytrain”, “yval”) for a particular train test split.
Train the model on the training set. This is “xtrain” and “ytrain”.
Test the model on the testing set (“xtest” and “ytest”) and evaluate the performance.

Logistic Regression with Count Vectorizer

```
from sklearn.feature_extraction.text import CountVectorizer

xtrain, xval, ytrain, yval = train_test_split(traindf['finalcleaned_authors'], traindf['Cat

# initalise the vectoriser
cvec = CountVectorizer()

# fit the training data on the model, transform training data into sparse matrix
xtrain_cvec = cvec.fit_transform(xtrain)

from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression()
classifier.fit(xtrain_cvec, ytrain)


cvec_score = cross_val_score(lr, xtrain_cvec, ytrain, cv=10)
#print(cvec_score)
df_cvec = pd.DataFrame(xtrain_cvec.toarray(),columns=cvec.get_feature_names())
print(df_cvec.head())

cvec_score = cross_val_score(lr, X_train_cvec, y_train, cv=10)
print('Score:',tvec_score.mean())

```

```
   aa  aaron  aarron  abbasi  abd  abdallah  abecasis  abel  abiola  abrahams  \
0   0      0       0       0    0         0         0     0       0         0
1   0      0       0       0    0         0         0     0       0         0
2   0      0       0       0    0         0         0     0       0         0
3   0      0       0       0    0         0         0     0       0         0
4   0      0       0       0    0         0         0     0       0         0

   ...  zollner  zon  zondervan  zoran  zou  zoya  zuben  zuber  zuccala  \
0  ...        0    0          0      0    0     0      0      0        0
1  ...        0    0          0      0    0     0      0      0        0
2  ...        0    0          0      0    0     0      0      0        0
3  ...        0    0          0      0    0     0      0      0        0
4  ...        0    0          0      0    0     0      0      0        0

   zuniga
0       0
1       0
2       0
3       0
4       0

[5 rows x 3420 columns]
Score: 0.7767109411561862
```

After created a 80/20 train-test split of the dataset, I've applied logistic regression which is a classification algorithm used to solve binary classification problems. The logistic regression classifier uses the weighted combination of the input features and passes them through a sigmoid function. Sigmoid function transforms any real number input, to a number between 0 and 1.
We have applied logistic regression classifier, on Tf-Idf features and got classifier score on Training Data and Testing Data.

	Predictions	Actual
0	Bottom_half	Bottom_half
1	Top_half	Top_half
2	Top_half	Top_half
3	Bottom_half	Top_half
4	Top_half	Top_half
5	Bottom_half	Top_half
6	Top_half	Top_half
7	Top_half	Top_half
8	Bottom_half	Bottom_half
9	Top_half	Top_half

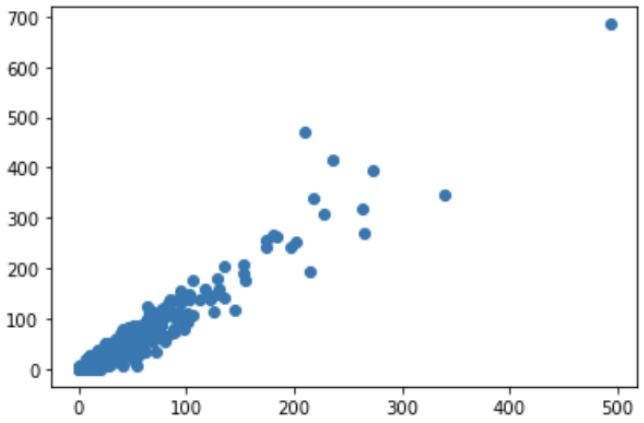
Prediction for Validation set. We randomly chose 10 Rows and tested the classifier attaining 80% accuracy.

Predict total citation based on previous trends using linear regression

Average citation per year and last year's citations

```
1  peryear = naturegenetics["Average per Year"]
2  lastyear = naturegenetics["2017"]
3  plt.scatter(peryear, lastyear)
```

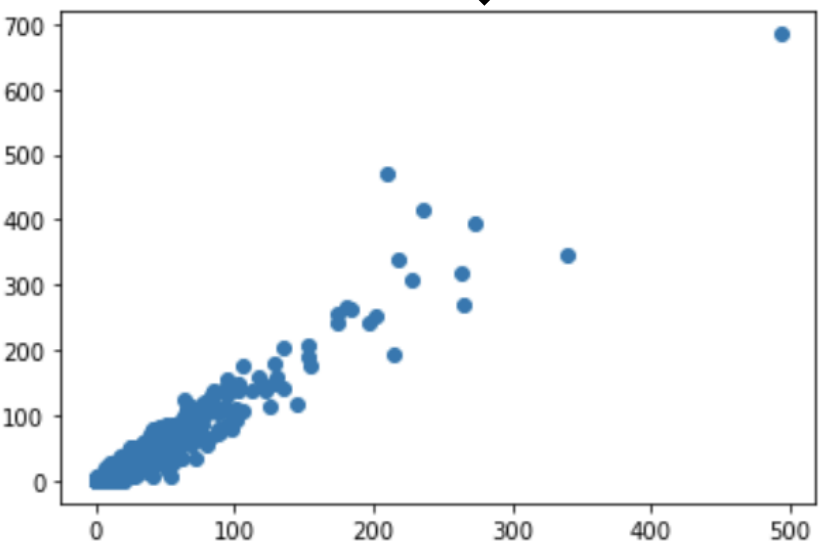
```
<matplotlib.collections.PathCollection at 0x11a77dd68>
```




```
1  # Assign the data to X and y
2  # Note: Sklearn requires a two-dimensional array of values
3  # so we use reshape to create this
4
5  X = peryear.values.reshape(-1, 1)
6  y = lastyear.values.reshape(-1, 1)
7
8  print("Shape: ", X.shape, y.shape)
```

```
Shape:  (3121, 1) (3121, 1)
```

Reshape



```
# Use sklearn's `train_test_split` to split the data into training and testing

from sklearn.model_selection import train_test_split

# YOUR CODE HERE
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42)

```

```
# Create the model

# YOUR CODE HERE
from sklearn.datasets import make_regression
from sklearn.linear_model import LinearRegression
model = LinearRegression()

```

```
# Fit the model to the training data.

# YOUR CODE HERE
model.fit(X_train, y_train)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

```
# Calculate the mean_squared_error and the r-squared value
# for the testing data

from sklearn.metrics import mean_squared_error, r2_score

# YOUR CODE HERE
# Use our model to predict a value
predicted = model.predict(X_test)

# Score the prediction with mse and r2
mse = mean_squared_error(y_test, predicted)
r2 = r2_score(y_test, predicted)

print(f"Mean Squared Error (MSE): {mse}")
print(f"R-squared (R2 ): {r2}")
```

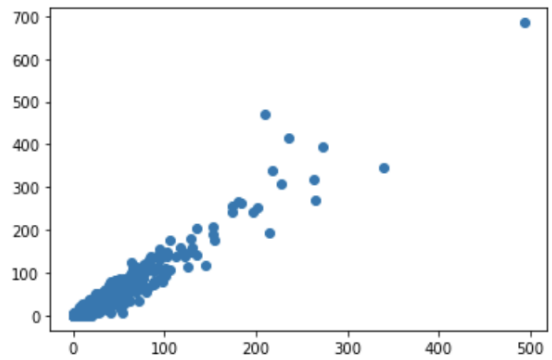
```
Mean Squared Error (MSE): 66.3033064342903
R-squared (R2 ): 0.9544251088797852
```

```
# Call the `score` method on the model to show the r2 score

# YOUR CODE HERE
model.score(X_test, y_test)
```

```
0.954425108879785
```

```
totalyear = naturegenetics["Average per Year"]
lastyear2 = naturegenetics["2017"]
plt.scatter(totalyear, lastyear2)
```

```
<matplotlib.collections.PathCollection at 0x11ee43da0>
```



Summary

- 3121 articles published between 2000 and 2017
- The highest cited paper was cited >5000 times
- On average, a NRG paper is cited 115 times
- Half of the dataset had no citations

Limitations

1. Biased Labels
2. Self-Citation
3. Citation count is not normalized by field/community.
4. Citations don't capture the impact of the work on folks who read, but don't cite, such as industry researchers, clinicians, or patients.
5. Different sources (for example, Scopus, Web of Science, and Crossref) often have different numbers for the same paper.
6. Citations take a long time to accumulate. It takes 6-7 months for a paper to make it through peer review and appear in most journals, which means it takes that much longer for each citation of the paper to appear.

Conclusion

Publishing a highly cited paper is gratifying and confirms that the work has an impact on the scientific community. However, the number of citations the top articles accrue depends on factors other than quality and originality.

Future Scope

If the short comes can be overcome then Citation count can actually help to a considerable extent the quality of work, in the absence of other criteria, is judged on the number of citations received, adjusting for the volume of work in the relevant topic. While this is not necessarily a reliable measure, counting citations is trivially easy; judging the merit of complex work can be very difficult.

References

Learning to Estimate Future Citations for Literature Rui Yan Dept. of Computer Science and Technology Peking University Beijing 100871, P. R. China r.yan@pku.edu.cn Jie Tang Dept. of Computer Science and Technology Tsinghua University Beijing 100871, P. R. China jietang@tsinghua.edu.cn Xiaobing Liu Dept. of Computer Science and Technology Peking University Beijing 100871, P. R. China lxb@net.pku.edu.cn Dongdong Shan Dept. of Computer Science and Technology Peking University Beijing 100871, P. R. China sdd@net.pku.edu.cn Xiaoming Li Dept. of Computer Science and Technology Peking University Beijing 100871, P. R. China lxm@pku.edu.cn

The Source Normalised Impact per Paper (SNIP) was created by Henk Moed (Moed, 2010)

A deep-learning based citation count prediction model with paper metadata semantic features, Anqi Ma, Yu Liu, Xiujuan Xu & Tao Dong.

Models for Predicting and Explaining Citation Count of Biomedical Articles
Lawrence D. Fu, M.S.1 and Constantin Aliferis, M.D., Ph.D.