Predicting the Appearance of Cited Article on Social Media Site- YouTube

Priyanjani Chandra
Department of Computer Science
Northern Illinois University
Z1864520@students.niu.edu

Sneha Ravi Chandran
Department of Computer Science
Northern Illinois University
Z1856678@students.niu.edu

PROJECT PROPOSAL

Problem Significance:
To potentially identify whether any particular research paper is posted on the social media platform based on related feature selection from the chosen dataset. Firstly, we should know how the articles are perceived on any given social media platform. In order to gain a fair knowledge about the comprehensive details such as how the cited papers are being perceived amongst a group, how the viewers enjoy the content or how they are being used as a reference for other related research proposal, we are considering YouTube as our choice of preference from the altmetric dataset and analyzing the features to classify them based on the appearance it had received on that platform. Thereby, having a tangible effect on benefiting the author to know how many users are engaged and exposed to any scholarly article on YouTube.

Research Hypotheses:
In comparison, YouTube being one of the visually empowering web sources for any citation of scholarly articles, it indulges in predicting whether a research paper will be cited in it or not.

Related Work:
The purpose behind altmetrics is to unite all the article-level metrics [1] that refers to the same publication. Altmetrics allowed us to know the academic use of various online environments. The use of altmetrics[2] properly can help in broader reach of new scientific innovations to the public. Altmetrics has been the source for computing alternative metrics on Twitter, blogs, Wikipedia, etc.
In the era of digitizing the search for the cited articles [3], we are looking at how any relevant research articles are posted on YouTube platform from that of the other social web services. This is chosen because, in real-time, the citation of any article takes a relative amount of time to be cited in any other research published or blogs as such but posting the same on social media would receive a quicker response as well as the count of citations tends to increase rapidly every time it is referred, receiving immediate attention.
Based on the state-of-the-art for the scholarly use of social media[4], we narrow down the identification of the engaging groups in scholarly communication. Observe the generalization of findings regarding altmetrics based on feature selection, the use of social media in academia and various ways that are used to cite the article in the social media. YouTube being our consideration, we would also be able to categorize more on other features such as number of views, likes and dislikes based on the features like category or subject from the YouTube dataset in our future predictions after performing the data collection for the same.
Measuring the rate at which the contents are posted on the social media based on the previously collected information helps us in estimating the cost and accurately predicting its popularity [5]. The evaluation of

the model was checked for the 10-fold cross validation by randomly dividing each dataset into training and test set. Analysis on the performance of the various model likes Szabo-Huberman model, Multivariate Linear model, etc. is detailed to compare the best performing one for YouTube Category. By assigning different weights to different popularity samples, the model would be able to differentiate videos on popularity evolution pattern and helps in reducing the errors on the predicted set.

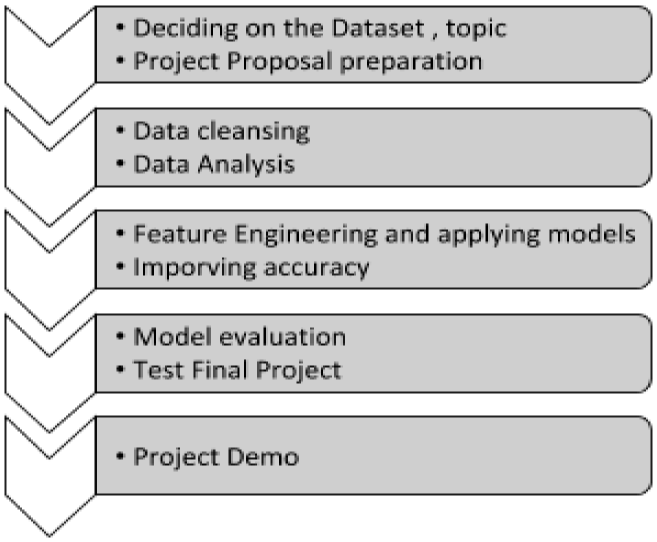
Data:
This big data project uses altmetric dataset. Altmetrics dataset contains normalized qualitative information about any given research article from which we are taking into account a subset of data which is used for categorizing based on the URL defined in one of the features of the dataset.

Methods:
We will perform the data cleansing step to the altmetric data, and we need to analyze the dataset. Features will be selected from the dataset based on the requirement of the classification. Necessary sampling will be applied, and we split the dataset into training and testing sets respectively. We will apply various classification algorithms such as Naive Bayes, Logistic Regression, Support Vector Machine, Decision Tree, etc. to the dataset. The model with the best output and better metrics will be selected as our final model for the prediction. We will be using packages such as Pandas, scikit-learn, seaborn and any other relevant packages in the course of completing the project.

Innovation:
We are using several classification models to make a reliable comparison in predicting the best model as well as it could be extended as a future proposal in predicting the number of views, likes, dislikes based on any category, subject or description.

Evaluation:
After the steps of data cleansing, feature selection and necessary sampling techniques we will be applying different classification algorithms on the dataset. We need to evaluate the models based on the performance on our dataset and also, we will be applying precision and recall metrics to evaluate the models. We may also use other metrics if needed in the course of completing the project.

Time Plan:


• Deciding on the Dataset , topic
• Project Proposal preparation
• Data cleansing
• Data Analysis
• Feature Engineering and applying models
• Imporving accuracy
• Model evaluation
• Test Final Project
• Project Demo

We are dividing each week work equally amongst the two of us and expecting to submit the project deliverable on or before Dec 10, 2019.

Expected Results:
The overall result shows the test set being correctly classified as to whether the given altmetrics data is present in YouTube by the model which is trained by training data. We are expecting to compare several model accuracies and visually represent them using the Machine learning libraries for data mining and data analysis.

References:
[1] Bornmann, Lutz. (2014). *Do altmetrics point to the broader impact of research? An overview of benefits and disadvantages of altmetrics. Journal of Informetrics.* 8. 10.1016/j.joi.2014.09.005.
[2] Weller, Katrin. (2015). *Social Media and Altmetrics: An Overview of Current Alternative Approaches to Measuring Scholarly Impact.* 10.1007/978-3-319-09785-5_16.
[3] Thelwall, Mike & Haustein, Stefanie & Larivière, Vincent & Sugimoto, Cassidy. (2013). *Do Altmetrics Work? Twitter and Ten Other Social Web Services*. PloS one. 8. e64841. 10.1371/journal.pone.0064841.
[4] Sugimoto, Cassidy & Work, Sam & Larivière, Vincent & Haustein, Stefanie. (2016). *Scholarly use of social media and altmetrics: A review of the literature. Journal of the Association for Information Science and Technology*. 10.1002/asi.23833.
[5] Pinto, Henrique, Jussara M. Almeida, and Marcos A. Gonçalves. "*Using early view patterns to predict the popularity of youtube videos*." In *Proceedings of the sixth ACM international conference on Web search and data mining*, pp. 365-374. ACM, 2013.

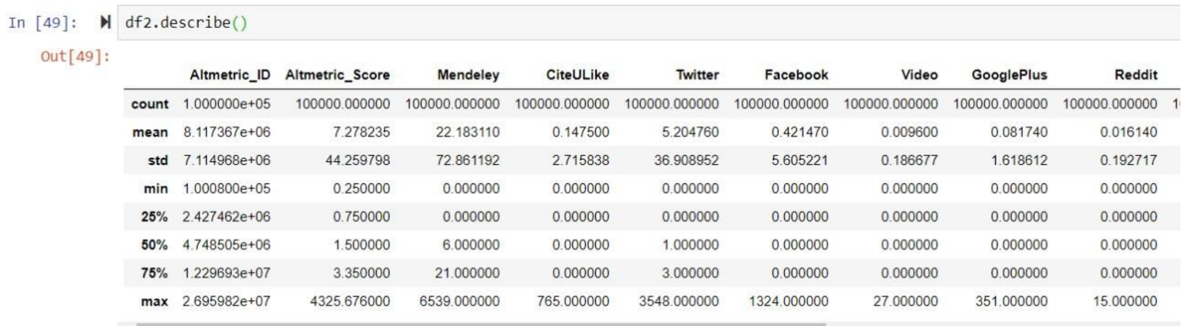
CHECKPOINT 1:

Dataset Used:
Altmetric_sample_dataset: (100,000 samples)
https://www.dropbox.com/sh/ezptabym0szfex3/AAAY30gQ5H3JE6qz3gH3_wqxa?dl=0&preview=altmetric_clean_sample.tar.gz

Altmetrics_dataset: (380,000 samples roughly)
https://www.dropbox.com/sh/zcloke9egpomdza/AAD0IOQrtcw-yUZMVK56R8-za?dl=0

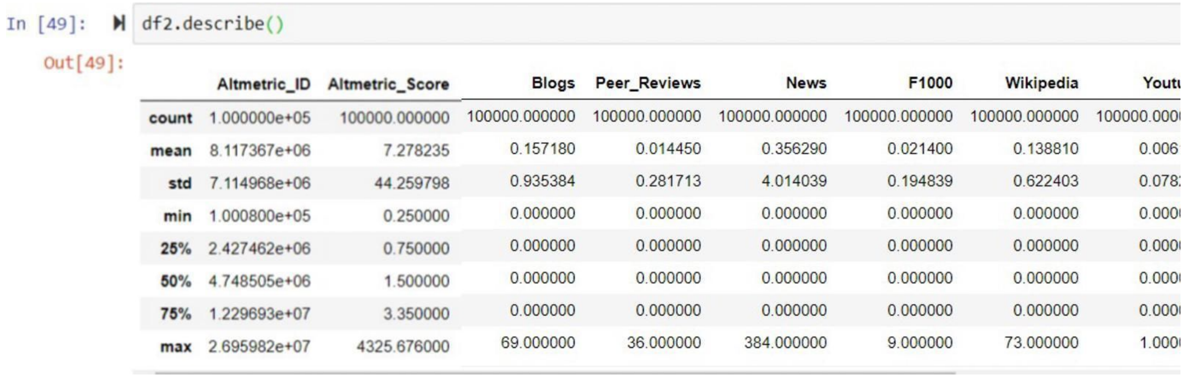
Summary and Descriptive Statistics of Your Data:wordworword
Altmetrics is a qualitative data that gives insights into the citation-based metrics. Our goal was to predict the influence the citation had on social media, YouTube in specific. The sample altmetric dataset had about 100,000 tuples. To get a fair insight of the data, the entire dataset wasn't taken into consideration, only a few samples are taken from the dataset initially. The features were validated against the *Youtube* label which holds the value as a 0 or 1 depending on whether the paper will be posted on YouTube or not. The features chosen were 'Altmetric_ID', 'Altmetric_Score', 'Mendeley', 'CiteULike', 'Twitter', 'Facebook', 'Video', 'GooglePlus', 'Reddit', 'Blogs', 'Peer_Reviews', 'News', 'F1000', 'Wikipedia', 'Youtube'.

Descriptive Statistics: Each of the social media platform had its own post count that influenced the paper cited and the altmetrics score of the same. Most of the JSON file we analyzed did not have all of the features we wanted to extract, so we had to go through several of them to get a common subset to give us clarity on the dataset we are trying to work on.
It was given to us that post counts are the number of unique items created by those users that mention this article. The same user could blog, tweet or otherwise share the same article more than once, so posts_count can be greater than unique_users_count from the feature list.



```
In [49]:  df2.describe()
Out[49]:
```

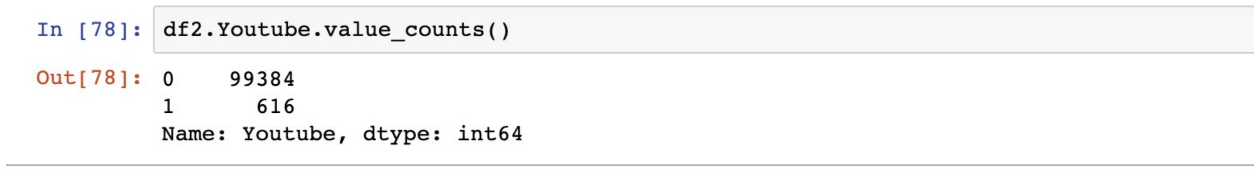
	Altmetric_ID	Altmetric_Score	Mendeley	CiteULike	Twitter	Facebook	Video	GooglePlus	Reddit
count	1.000000e+05	100000.000000	100000.000000	100000.000000	100000.000000	100000.000000	100000.000000	100000.000000	100000.000000
mean	8.117367e+06	7.278235	22.183110	0.147500	5.204760	0.421470	0.009600	0.081740	0.016140
std	7.114968e+06	44.259798	72.861192	2.715838	36.908952	5.605221	0.186677	1.618612	0.192717
min	1.000800e+05	0.250000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	2.427462e+06	0.750000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	4.748505e+06	1.500000	6.000000	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000
75%	1.229693e+07	3.350000	21.000000	0.000000	3.000000	0.000000	0.000000	0.000000	0.000000
max	2.695982e+07	4325.676000	6539.000000	765.000000	3548.000000	1324.000000	27.000000	351.000000	15.000000



```
In [49]:  df2.describe()
Out[49]:
```

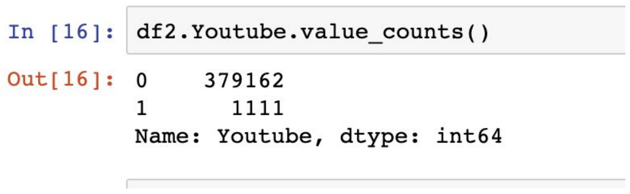
	Altmetric_ID	Altmetric_Score	Blogs	Peer_Reviews	News	F1000	Wikipedia	Youtu
count	1.000000e+05	100000.000000	100000.000000	100000.000000	100000.000000	100000.000000	100000.000000	100000.000
mean	8.117367e+06	7.278235	0.157180	0.014450	0.356290	0.021400	0.138810	0.006
std	7.114968e+06	44.259798	0.935384	0.281713	4.014039	0.194839	0.622403	0.078
min	1.000800e+05	0.250000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000
25%	2.427462e+06	0.750000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000
50%	4.748505e+06	1.500000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000
75%	1.229693e+07	3.350000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000
max	2.695982e+07	4325.676000	69.000000	36.000000	384.000000	9.000000	73.000000	1.000

The *sample altmetrics dataset* used gave us below count for YouTube label:



```
In [78]: df2.Youtube.value_counts()

Out[78]: 0    99384
         1      616
         Name: Youtube, dtype: int64
```

The below gave us count for the original *altmetrics dataset*.



```
In [16]: df2.Youtube.value_counts()

Out[16]: 0    379162
         1      1111
         Name: Youtube, dtype: int64
```

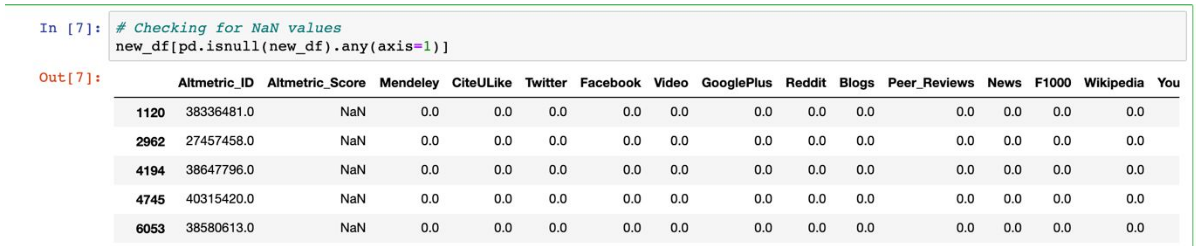
Data Cleaning Steps Taken:

We started with the *sample altmetric dataset* which was relatively smaller to work with in order to get clarity on the data cleansing and storing the data to the *pandas dataframe* for further processing. After being successfully able to execute that step, we loaded the original *altmetrics dataset*.

Firstly, in order to identify the features, we extracted the JSON format file and analyzed on the available keys present in various levels. Upon necessary formatting, we were able to narrow down our features and label to run our model for prediction on the test dataset from the training dataset.

Fetching of the data from the local path specified would fetch all the sub folder inside the main folder using *os.listdir(PATH),* we then opened the.
From each JSON file we need to fetch the above-mentioned features and store them as a dictionary *dict_values*. 'Youtube' was taken as the test label whose values are 0 or 1 depending on whether the Altmetric_ID has the keyword *youtube* under posts -> video -> url. Features mendeley and citeulike are under counts-> readers. The remaining features are taken from counts -> respective feature -> posts_count. The dictionary is then converted into a pandas dataframe and saved as a CSV file.

Reading the CSV file, we saved after cleaning the data from json, using pandas dataframe we printed out the head of the (first 5 records) to show a sample dataframe table of data. The *info()* helped us to gain insight on the table information, column data types, entry count and the memory usage. It is to be noted that the data had some NaN values for the Altmetrics_Score.

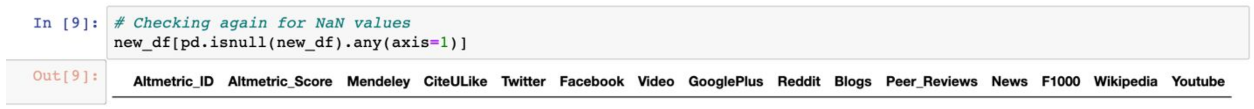


```
In [7]: # Checking for NaN values
        new_df[pd.isnull(new_df).any(axis=1)]
```

Out[7]:

	Altmetric_ID	Altmetric_Score	Mendeley	CiteULike	Twitter	Facebook	Video	GooglePlus	Reddit	Blogs	Peer_Reviews	News	F1000	Wikipedia	You
1120	38336481.0	NaN	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
2962	27457458.0	NaN	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
4194	38647796.0	NaN	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
4745	40315420.0	NaN	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
6053	38580613.0	NaN	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	

Of the entire data, 245 rows × 15 columns were extracted having NaN values for the Altmetrics_Score. So, we dropped the rows with NaN values by *new_df.dropna()*. The result after performing this operation is given below,

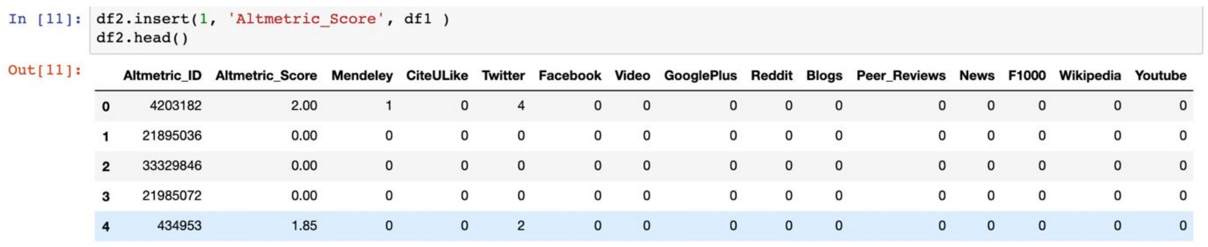


```
In [9]: # Checking again for NaN values
        new_df[pd.isnull(new_df).any(axis=1)]
```

Out[9]:

Altmetric_ID	Altmetric_Score	Mendeley	CiteULike	Twitter	Facebook	Video	GooglePlus	Reddit	Blogs	Peer_Reviews	News	F1000	Wikipedia	Youtube

The dtype of all the features and label data shows as *float64* but we had to convert it to integer format. However, 'Altmetric_Score' need not be changed, so we dropped the particular column from the dataframe and converted the type of the other columns to integer.
The final dataframe looks like the below,



```
In [11]: df2.insert(1, 'Altmetric_Score', df1 )
         df2.head()
```

Out[11]:

	Altmetric_ID	Altmetric_Score	Mendeley	CiteULike	Twitter	Facebook	Video	GooglePlus	Reddit	Blogs	Peer_Reviews	News	F1000	Wikipedia	Youtube
0	4203182	2.00	1	0	4	0	0	0	0	0	0	0	0	0	0
1	21895036	0.00	0	0	0	0	0	0	0	0	0	0	0	0	0
2	33329846	0.00	0	0	0	0	0	0	0	0	0	0	0	0	0
3	21985072	0.00	0	0	0	0	0	0	0	0	0	0	0	0	0
4	434953	1.85	0	0	2	0	0	0	0	0	0	0	0	0	0

Problems faced:

Our data wasn't getting populated correctly, we noticed that all the values were 0 regardless of having an absolute value in the JSON data. So, we had to narrow down on what we were missing, and it came to our notice that we made mistakes in calling the feature while storing in the dictionary. For example, 'Facebook' was present inside the key 'counts' from where we had to take the 'posts_count'. Similar mishaps we corrected resulting in proper values being populated.

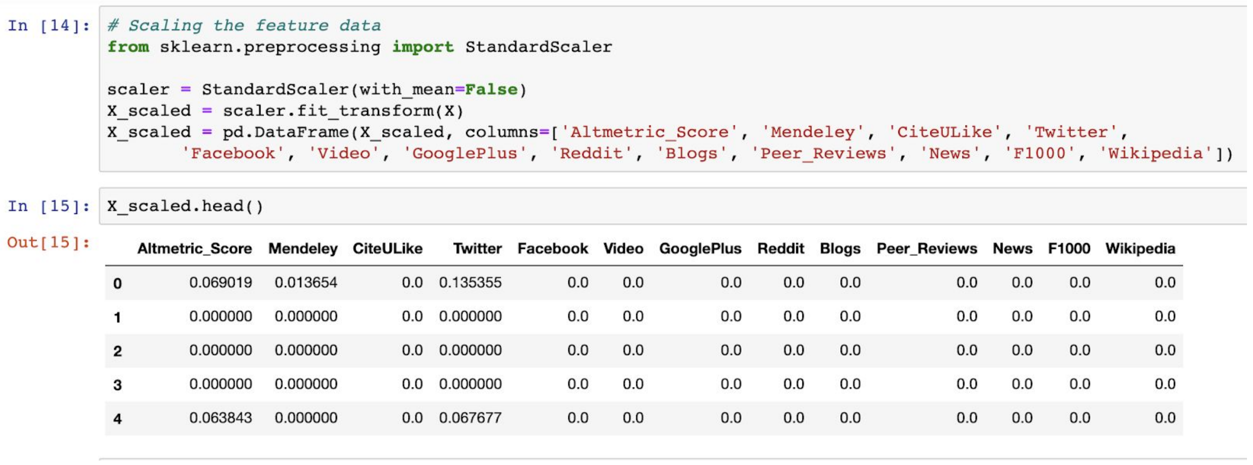
Insights and Next Steps:

We found that we had to scale the values since Mendeley had relatively larger value than that of the other columns. Scaling was performed on all the columns except 'Altmetric_ID', 'Youtube'. We will use preprocessing.*StandardScaler* from *Scikit-learn* library and transform the data.

CHECKPOINT 2:

Feature Engineering:

We obtained the scaled data for all the columns except 'Altmetric_ID', 'Youtube'.



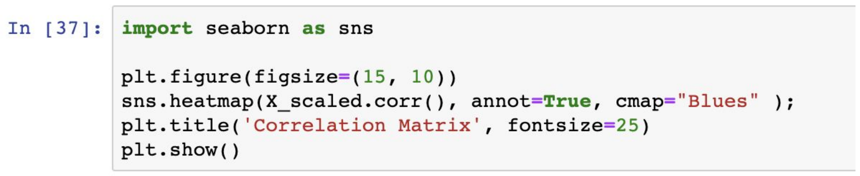
```
In [14]: # Scaling the feature data
         from sklearn.preprocessing import StandardScaler

         scaler = StandardScaler(with_mean=False)
         X_scaled = scaler.fit_transform(X)
         X_scaled = pd.DataFrame(X_scaled, columns=['Altmetric_Score', 'Mendeley', 'CiteULike', 'Twitter',
                'Facebook', 'Video', 'GooglePlus', 'Reddit', 'Blogs', 'Peer_Reviews', 'News', 'F1000', 'Wikipedia'])
```

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

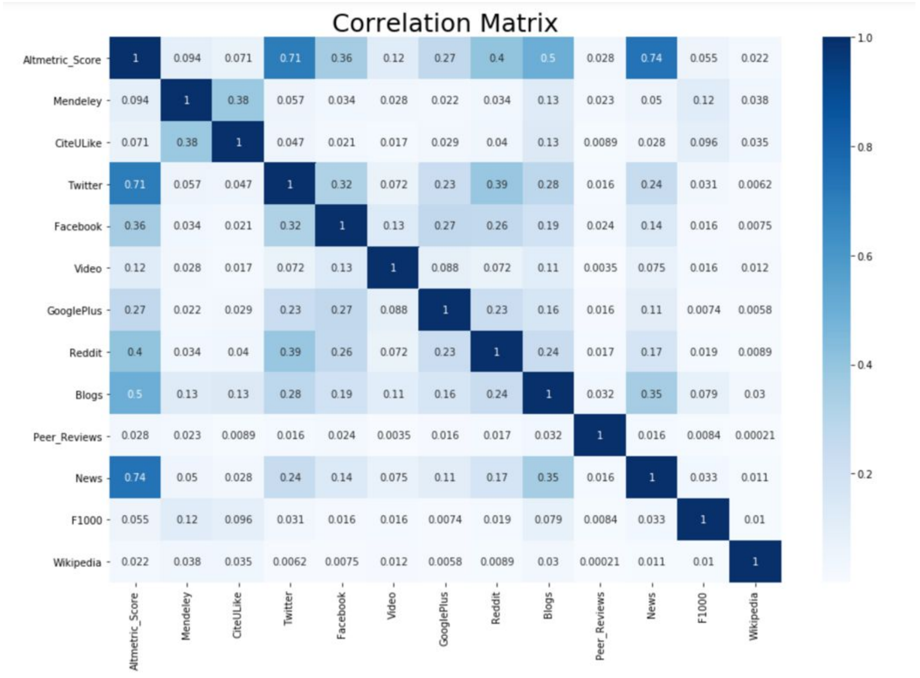
Out[15]:

	Altmetric_Score	Mendeley	CiteULike	Twitter	Facebook	Video	GooglePlus	Reddit	Blogs	Peer_Reviews	News	F1000	Wikipedia
0	0.069019	0.013654	0.0	0.135355	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.000000	0.000000	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.000000	0.000000	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.000000	0.000000	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.063843	0.000000	0.0	0.067677	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

For printing out the correlation matrix, we imported *seaborn* library and used a heatmap to give a decent look and feel. The *annot* value is set as *True* because we wanted to show the data value in each cell, *cmap* helps us map the data values to the color space. Also, we used *matplotlib* library for 2D plotting.



```
In [37]: import seaborn as sns

         plt.figure(figsize=(15, 10))
         sns.heatmap(X_scaled.corr(), annot=True, cmap="Blues" );
         plt.title('Correlation Matrix', fontsize=25)
         plt.show()
```

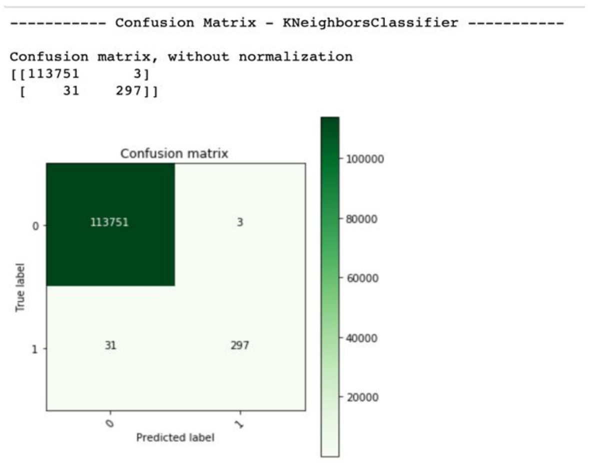


	Altmetric_Score	Mendeley	CiteULike	Twitter	Facebook	Video	GooglePlus	Reddit	Blogs	Peer_Reviews	News	F1000	Wikipedia
Altmetric_Score	1	0.094	0.071	0.71	0.36	0.12	0.27	0.4	0.5	0.028	0.74	0.055	0.022
Mendeley	0.094	1	0.38	0.057	0.034	0.028	0.022	0.034	0.13	0.023	0.05	0.12	0.038
CiteULike	0.071	0.38	1	0.047	0.021	0.017	0.029	0.04	0.13	0.0089	0.028	0.096	0.035
Twitter	0.71	0.057	0.047	1	0.32	0.072	0.23	0.39	0.28	0.016	0.24	0.031	0.0062
Facebook	0.36	0.034	0.021	0.32	1	0.13	0.27	0.26	0.19	0.024	0.14	0.016	0.0075
Video	0.12	0.028	0.017	0.072	0.13	1	0.088	0.072	0.11	0.0035	0.075	0.016	0.012
GooglePlus	0.27	0.022	0.029	0.23	0.27	0.088	1	0.23	0.16	0.016	0.11	0.0074	0.0058
Reddit	0.4	0.034	0.04	0.39	0.26	0.072	0.23	1	0.24	0.017	0.17	0.019	0.0089
Blogs	0.5	0.13	0.13	0.28	0.19	0.11	0.16	0.24	1	0.032	0.35	0.079	0.03
Peer_Reviews	0.028	0.023	0.0089	0.016	0.024	0.0035	0.016	0.017	0.032	1	0.016	0.0084	0.00021
News	0.74	0.05	0.028	0.24	0.14	0.075	0.11	0.17	0.35	0.016	1	0.033	0.011
F1000	0.055	0.12	0.096	0.031	0.016	0.016	0.0074	0.019	0.079	0.0084	0.033	1	0.01
Wikipedia	0.022	0.038	0.035	0.0062	0.0075	0.012	0.0058	0.0089	0.03	0.00021	0.011	0.01	1

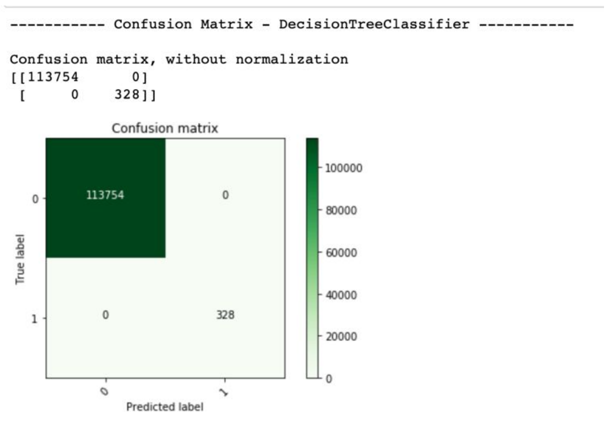
From the correlation matrix, we were able to understand that Altmetrics score had correlation with few of the other features since the altmetrics score calculation was based on the post count in other social media which we have considered as our features.
Using *train_test_split from sklearn.model_selection* training and testing were split randomly in the ratio of 70:30. We defined the function for plotting the confusion matrix without normalization.

The Models used to predict are:

- K- Nearest Neighbor
- Decision tree classifier
- Logistic Regression



```
----------- Confusion Matrix - KNeighborsClassifier -----------

Confusion matrix, without normalization
[[113751      3]
 [    31    297]]
```

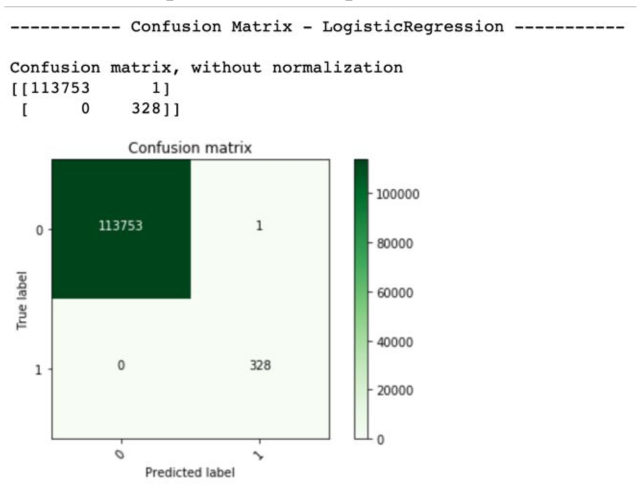


```
----------- Confusion Matrix - DecisionTreeClassifier -----------

Confusion matrix, without normalization
[[113754      0]
 [     0    328]]
```

K- Nearest Neighbor: For classification, we created the instance of the *KNeighborsClassifier* by importing the same from *Scikit-learn* library. In order to predict the test results, we fit the training feature and label and check on the unseen set of results. Internally what this algorithm does is, it picks a value for k (we have given the *n_neighbors* size as 4) and searches for the k observations in the training set that is near to the unknown data point. The most appearing class value is taken the predicted response.
From the confusion matrix we can conclude that 31 samples are classified as false positive and 3 are false negative. The false negative condition is the critical consideration for checking if there is a high risk of improper classification.

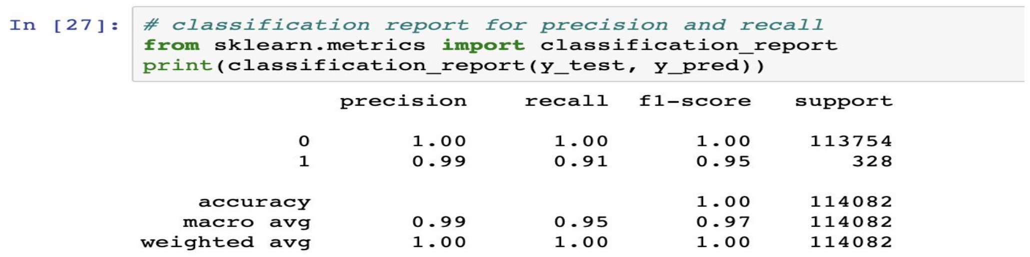
Decision Tree classifier: The *criterion* used as entropy which is the measure of uncertainty in the *DecisionTreeClassifier* which is imported from *sklearn.tree*. The entropy decides where the split happens in the data. The *random_state* is the seed value used by the random number generator. Each node in the decision tree specifies a test of some attributes while the branch corresponds to the attribute value. The leaf node denotes the class labels. Decision tree is generally prone to overfitting if we have many features and one other concern is to stop the growth of the tree in case of more feature values.

Logistic Regression: It is to be noted that for smaller training set, logistic regression yields better accuracy than decision tree. We imported *LogisticRegression* from the s*klearn.linear_model* and used fit function for the training features and label. Then prediction was performed on the unseen test dataset.



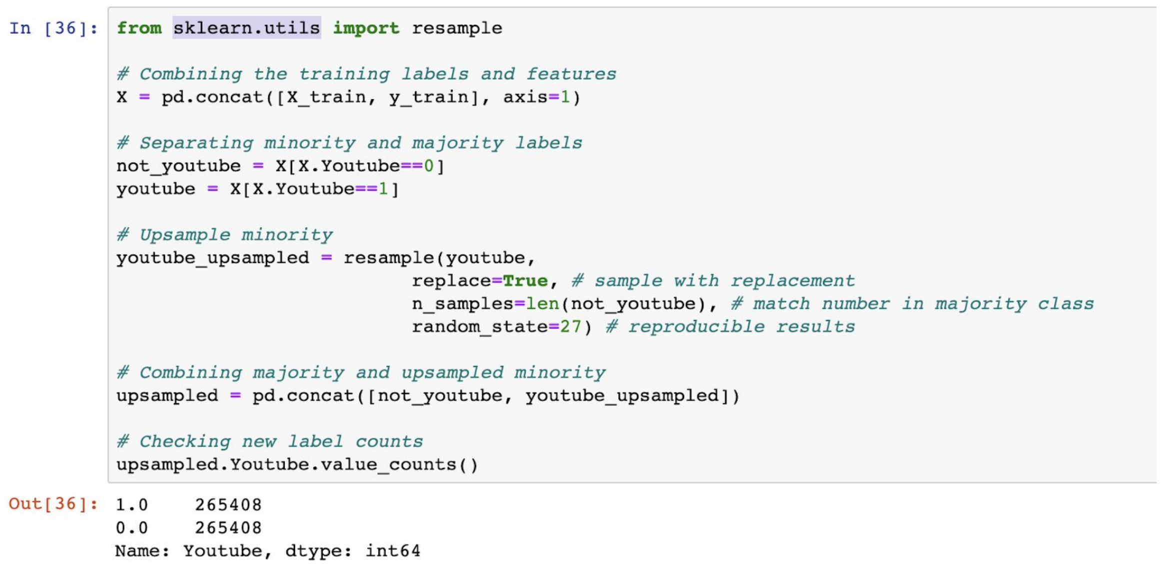
```
----------- Confusion Matrix - LogisticRegression -----------

Confusion matrix, without normalization
[[113753      1]
 [     0    328]]
```

Ways to improve your accuracy:
For the data we collected we have pretty good accuracy, which makes us wonder if there was any overfitting of the data. We are looking at what necessary steps to be taken before our final submission to get a difference in the marginal accuracy so that when we apply any oversampling techniques to handle class imbalance problem we could potentially end up getting a good accuracy rate making it a reliable for the comparison of the model as well.
Precision and Recall report:
KNN before Sampling:



```
In [27]: # classification report for precision and recall
         from sklearn.metrics import classification_report
         print(classification_report(y_test, y_pred))

                       precision    recall  f1-score   support

                    0       1.00      1.00      1.00    113754
                    1       0.99      0.91      0.95       328

             accuracy                           1.00    114082
            macro avg       0.99      0.95      0.97    114082
         weighted avg       1.00      1.00      1.00    114082
```

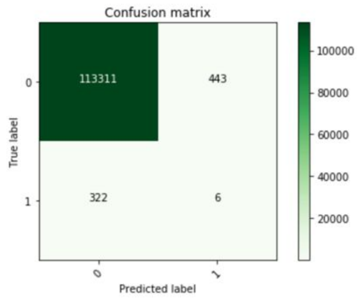
Oversampling:
As the labels are highly imbalanced we need to perform oversampling to the data. We imported *resample* from *sklearn.utils* and combined the training labels and features using *concat.* Separation of the minority and majority labels was performed in order to oversample minority labels matching the count with that of the majority class. The total dimension the result of this operation is 265408*2 (predicted cited paper in YouTube or not) is shown below. There are 156475 rows with Altmetric_Score value as NaN, so we have dropped these rows from the dataset before proceeding any further.



```
In [36]: from sklearn.utils import resample

         # Combining the training labels and features
         X = pd.concat([X_train, y_train], axis=1)

         # Separating minority and majority labels
         not_youtube = X[X.Youtube==0]
         youtube = X[X.Youtube==1]

         # Upsample minority
         youtube_upsampled = resample(youtube,
                                   replace=True, # sample with replacement
                                   n_samples=len(not_youtube), # match number in majority class
                                   random_state=27) # reproducible results

         # Combining majority and upsampled minority
         upsampled = pd.concat([not_youtube, youtube_upsampled])

         # Checking new label counts
         upsampled.Youtube.value_counts()

Out[36]: 1.0    265408
         0.0    265408
         Name: Youtube, dtype: int64
```

Again, on applying KNN to the sampled dataset after splitting the data into training and testing the below result is obtained.

```
----------- Confusion Matrix - KNeighborsClassifier ----------

Confusion matrix, without normalization
[[113311    443]
 [   322      6]]
```



```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    113754
           1       0.01      0.02      0.02       328

    accuracy                           0.99    114082
   macro avg       0.51      0.51      0.51    114082
weighted avg       0.99      0.99      0.99    114082
```

Future Work

We have to work on the perfect accuracy problem. This might be occurring either due to overfitting(but the training accuracy is also high) or because video is a feature and YouTube is the class. Whenever there is a certain value in the video, the model may learn to give the output class as 1(Video has 616 non-zero values same as that of the YouTube count). We have to try by removing this feature. For the case of class imbalance, we need to apply few sampling techniques and observe which gives more accuracy. We are planning to plot ROC, precision- recall curve for the same to visually understand better. If the time permits, we are interested in doing a Regression task on the YouTube dataset.