Predicting the Popularity of Instagram Posts

Patrick, M, Pickard
University of Calgary - M. Eng Software Engineering Student - patrick.pickard@ucalgary.ca
Ziad,T, Chemali
University of Calgary - M. Eng Software Engineering Student
Joshua, D, Posyluzny
University of Calgary - M. Eng Software Engineering Student - joshua.posyluzny@ucalgary.ca

Summary of Contributions:

We would like to point out here that we divided our contributions in such a way that we all collaborated on each section, but one person was driving Databricks or the Crawler for each section.

Patrick Pickard: Drove the Instagram Crawler while we all collaborated on it, and wrote how the new data labeled/collected, and how does the newly added data compare with the original data sections.

Ziad Chemali: Drove the Feature Engineering while we all collaborated on it, and wrote how the data was preprocessed, and how the models performed on the original data vs the new + original data sections**.**

Joshua Posyluzny: Drove the Machine Learning while we all collaborated on it, and wrote how the performance of the models changed based on the choice of hyperparameters, and how the misclassifications of the best performing model were distributed sections.

Feature Engineering Notebook Link:
https://databricks-prod-cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/506476747668026/3695575757594814/8670875371630392/latest.html

Original Models Notebook Link:
https://databricks-prod-cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/506476747668026/3752610390424296/8670875371630392/latest.html

New Model Notebook Link:
https://databricks-prod-cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/506476747668026/479440352926630/8670875371630392/latest.html

Abstract

Context

Social media sites offer a great outlet for training and testing machine learning classification models. They have excess amounts of public data, and are reflective or real world sentiment as it is real time accessible.

Objective

Our group's goal was to mimic the referenced paper "Popularity Prediction of Instagram Posts" [1]. We would use their dataset to replicate their results, extend their dataset with 1000 new data points, and re-train our models on this extended dataset to determine if we could improve on their results.

Method

Our group scraped 1000 public Instagram posts from a total of 10 public ordinary users, processed the obtained Metadata, generated pertinent features, trained machine learning models on this data, and then measured the performance of these models using quantifiable metrics.

Results

Not only were we able to replicate closely the original papers results [1], our newly trained models on the extended dataset did in fact outperform the papers metrics. Additionally, we ended up training an additional model not included in the original papers scope for comparison purposes, which compared favorably as well.

Conclusion

This work depicts that the classification models used, as well as the metadata features scraped from public Instagram posts can be used to adequately train a machine learning classification model.

Introduction

Using machine learning to perform classification tasks can be extremely powerful. Essentially infinite problems can use this technique to remove potential biases that might come from pure human classification analysis, cut straight through possible data manipulation issues, and focus on the statistical analysis of a subject to generate far more objectively sound predictions for these problems than any human alone ever could. The scope of our project was to generate a model capable of predicting the popularity of an Instagram users post based on metadata features that could be gathered on any public profiles posts. While this may not seem readily useful, it demonstrates the vast variety of applications a classification model like this can be applied to; everything from social media sites, to financial analysis, to medical field imaging are all possible candidates for a similar classification project. While machine learning for classification tasks is not a new concept as it has been around for numerous years, and the underlying statistics for classification is far older, this technique can still offer huge benefits when applied appropriately and handled correctly.

In the following section, our procedures/methods, and associated results will be discussed.

Results

How was the new data labeled/collected?

Approach:

As per the scope of this project, our group extended the original dataset (consisting of 127,466 posts) by an additional 1000 posts. The posts were selected using the same criterion used in the reference paper [1], which is to say, only ordinary profiles were selected (profiles which had less than 25,000 followers), and only profiles that had at least 100 public posts made. The profiles were selected and listed out to be used by the Instagram Scraper API [2] used to handle the physical scraping of the posts metadata. This API returned a JSON file for each user that was scraped, containing a list of JS objects holding all of the pertinent metadata required for our model creation.

Once this data was scraped and loaded, the data was labelled. As the original paper labelled their data, which I will now refer to as the target vector, based on the number of likes a post had relative to the average number of likes of the users previous 10, 30, or 50 posts, each new datapoint was labelled based on that. Example, if the 11th post had 50 likes, and the previous 10 posts have an average likes of 46, this 11th post would be labelled as popular (denoted by 1 in our dataframe). Conversely, if the 33rd post had 78 likes, and the previous 30 posts had an average likes of 87, this 33rd post would be labelled as not-popular (denoted by 0 in our dataframe).

As this method by the referenced paper used these 3 different criteria for labelling the data (avg 10, 30, or 50 previous posts), this led to 3 separate target vectors created in order to mimic their process. Additionally, the referenced paper also used a "popularity threshold" (called delta) of 4 different cases as well; 0 (only needs 1 more like than the previous avg being compared against), 5% (needs 5% more likes than the previous avgs being compared against), 10%, and 15%. This led to us creating 12 target vectors that required labelling. Fortunately, as this was arithmetic based labelling, we were able to create a function to simply handle this, and thus, lead to no mislabelling or labelling disagreements between our group members.

Results:

As mentioned above, the labelling was determined based on an average value, and a function was created to determine if the current post being labelled was higher than that value (label as popular, 1), or lower (label as not popular, 0). This allowed for NO disagreements in labelling between group members.

As our data was scraped and labelled functionally, we can attest to the quality of the data captured. There was no possibility for mislabelling criterion (different between our group and the papers group) to affect the results as no human biases were present in our data labelling. This leads to a 100% labelling agreement since the labelling was decided on a value, not a group members opinion. Additionally, our data was manually verified, and no null/empty rows were scraped, 100% of the columns contained properly formatted fields, and all of the data was manipulated soundly.

How does the newly added data compare with the original data?

Approach:

Comparing the original dataset to our newly extended data that was scraped was done through some basic descriptive statistics. The means, minimum values, maximum values, standard deviation, etc, were computed for both datasets key features as can be seen in the below tables.

Results:

Our new dataset features were exactly formatted to match the original dataset formatting. These consisted of the following features:

- Author of the post
- Caption of the post
- If the post was a video or not
- Number of likes the post got
- Account number of followers
- Hashtag counts
- User tag counts
- Number of caption words
- The timestamp of the post

All of these features were obtained from the metadata scraping of the posts using the aforementioned instagram-scraper api [2]. A snip of the original dataset can be seen below:

	num	author	caption	is_video	likes	num_follower	sentiment_score	hashtags_count	users_tagged	num_words	mean_5	mean_10	mean_15	mean_20	mean_30	mean_50	PrevPost_1	PrevPost_2	PrevPost_3	PrevPost_4	timestamp	Type
0	49	mari_nostru	Buongiorno a tutti \nSaremo operativi da vener...	False	31	528.0	0.0	0.0	0.0	170.0	33.2	39.1	42.933333	47.25	47.833333	46.88	30	17	22	37.0	2020-03-11 18:01:00	Old
1	48	mari_nostru	Condividete questo contatto.\nPamela si occupe...	False	23	528.0	0.0	0.0	0.0	11.0	27.4	38.3	41.400000	44.85	47.266667	45.98	31	30	17	22.0	2020-03-12 09:32:00	Old
2	47	mari_nostru	Il colore del cielo mi riempie gli occhi di lu...	False	27	528.0	0.0	0.0	0.0	267.0	24.6	35.0	40.400000	42.40	46.600000	45.66	23	31	30	17.0	2020-03-12 16:59:00	Old
3	46	mari_nostru	Un pochino di ironia ci aiuterà !\nLa speranza...	False	19	528.0	0.0	0.0	0.0	19.0	25.6	32.9	37.133333	41.80	45.766667	44.96	27	23	31	30.0	2020-03-13 14:16:00	Old
4	45	mari_nostru	Servizio rivolto agli anziani, ai malati, agli...	False	16	528.0	0.0	0.0	0.0	28.0	26.0	30.6	35.266667	39.80	44.433333	44.00	19	27	23	31.0	2020-03-14 15:26:00	Old

Figure 1: Pandas dataframe representation of the original dataset obtained from the paper's authors. Some of the generated features have been seen here as this dataset included some attributes that were not present when the profiles were scraped.

Summary statistics for both the original paper's key values can be seen in the following figures 2 and 3 shown below. The non "raw" features obtained from the original post scraping process have been removed for clearer comparison:

	likes	num_follower	hashtags_count	users_tagged	num_words
count	117824.000000	117824.000000	117824.000000	117824.000000	117824.000000
mean	126.284153	2149.081664	7.166723	0.287734	14.578414
std	262.715095	3210.284379	9.902745	1.381853	30.974031
min	0.000000	41.000000	0.000000	0.000000	0.000000
25%	30.000000	667.000000	0.000000	0.000000	1.000000
50%	61.000000	1142.000000	1.000000	0.000000	5.000000
75%	125.000000	1992.000000	12.000000	0.000000	13.000000
max	14897.000000	24700.000000	70.000000	39.000000	390.000000

Figure 2: Descriptive statistics for the original papers dataset.

	likes	num_follower	hashtags_count	users_tagged	num_words
count	970.000000	970.000000	970.000000	970.000000	970.000000
mean	149.734021	3020.565979	8.375258	0.635052	28.385567
std	235.879159	3927.273674	10.937358	1.336480	40.821035
min	3.000000	173.000000	0.000000	0.000000	0.000000
25%	41.000000	720.000000	0.000000	0.000000	7.000000
50%	66.000000	1001.000000	2.000000	0.000000	16.500000
75%	120.750000	2807.000000	16.000000	1.000000	29.000000
max	2278.000000	14210.000000	32.000000	10.000000	274.000000

Figure 3: Descriptive statistics for our newly scraped dataset consisting of 970 new posts.

As we can see, the original paper's dataset consisted of 117,824 scraped posts used as data points. These are posts that conform to the “ordinary user” constraints mentioned previously in this paper, as well as any points containing NaN values in pertinent columns. This is compared to our 970 newly scraped posts on profiles also conforming to the “ordinary user” constraints. Visualization of the remaining statistics can be seen in the following figures 4, 5, 6, and 7 shown below:

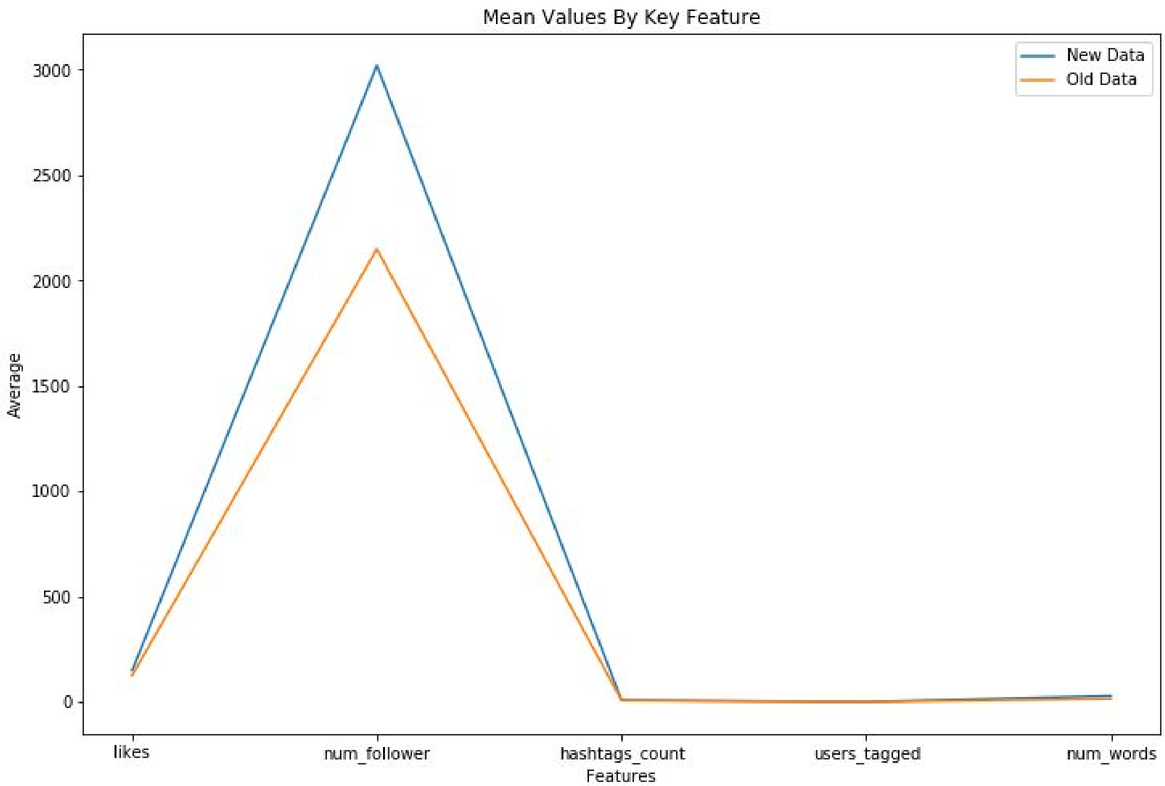


Figure 4: Averages per feature comparing the original dataset values to our new extended dataset.

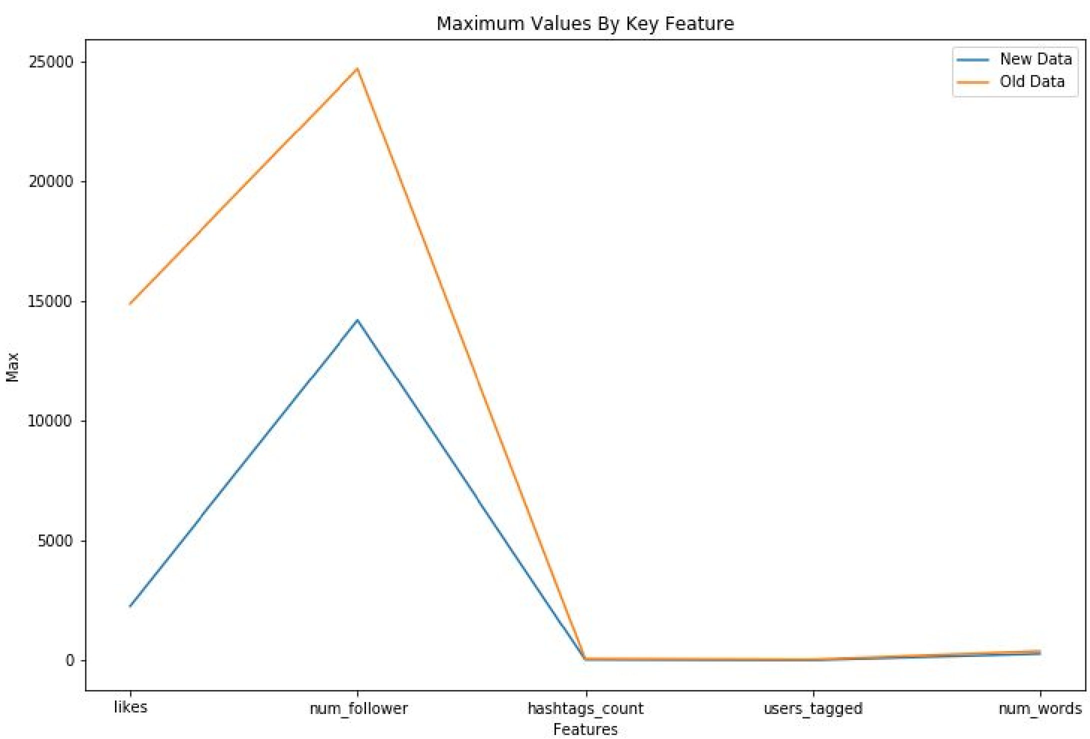


Figure 5: Maximum values per feature comparing the original dataset values to our new extended dataset.

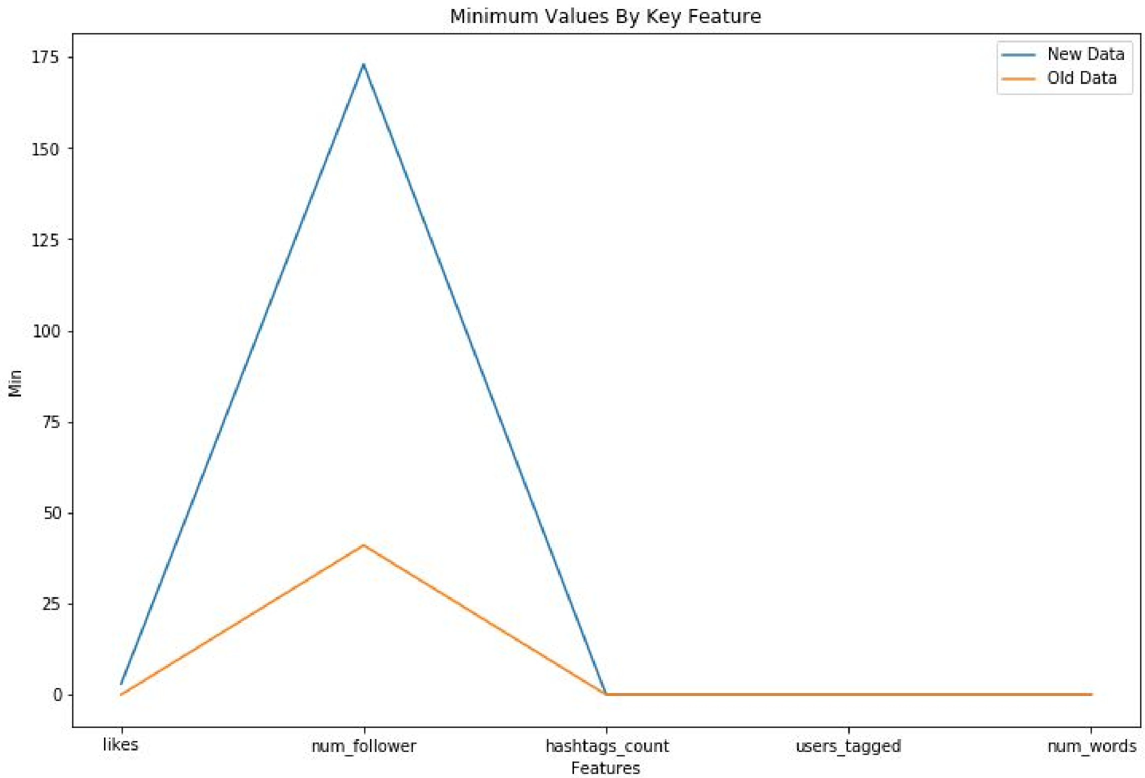


Figure 6: Minimum values per feature comparing the original dataset values to our new extended dataset.

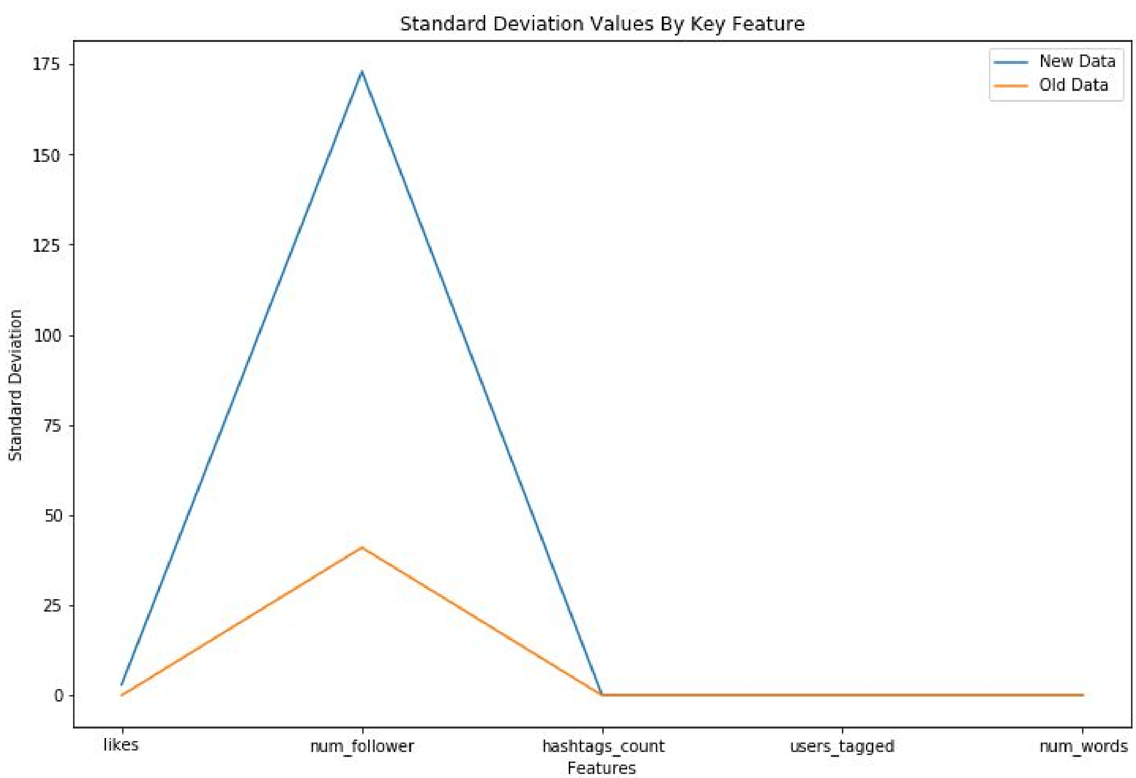


Figure 7: Standard Deviation per feature comparing the original dataset values to our new extended dataset.

As can be seen in the above figures, the extended dataset had a higher average number of followers (as well as the minimum and standard deviation values) per post than our extended dataset, but had a slightly lower maximum value. While the max and min values in these datasets are not particularly useful, they do give a general idea on the range of the data for both sets. The larger standard deviation in our extended dataset can most likely be attributed to the far smaller sample size when compared to the original dataset.

The important points to be taken from this data analysis, is that the averages for the important features shown above are very similar; ~29% difference between the old and extended (new) average number of followers, and all of the other features are exceedingly comparable to each other, indicating that there should be good compatibility between comparing any results using these datasets.

How was the data preprocessed ?

Approach:

Data preprocessing was done by following the steps performed by the paper's authors. The extracted new features can be divided into the following categories :

<u>i) Average likes type:</u>

This type of features, consider the average likes achieved by the K most recent posts published by the account for each $K \in \{5, 10, 15, 20, 30, 50\}$. The average likes was calculated by using a rolling average where K is the size of the rolling average window.

<u>ii) Recent likes type :</u>

The exact number of likes of the latest published posts upto the last 5 previous posts per user.

<u>iii) Time features :</u>

Transformed the post date to separate features such as time, day of the week, month and season of the post was uploaded.

<u>iv) Text related features:</u>

For each caption we extracted the number of words, the number of users tagged, the number (and importance) of chosen hashtags, and a sentiment score.

Also extracted the emojis corresponding to 10 different features/columns: happiness, love, sadness, travel, food, pet, anger, music, party and sport. Where each of these emoji features is a binary feature ,its value depends on the presence of at least one emoji of respective type.

Finally, did a frequency count of the hashtags used for all users and chose the top 5 and least 5 hashtags used then used them as features. Thes 10 hashtag features are binary features where their value depends on the presence of at least one hashtag in the caption.

v) Popularity features:

Added a separated label for each different pair of parameters K (number of previous posts) and ∆ (tolerance).In the paper they used K values equal to 10, 30 and 50, while ∆ values equal to 0, 0.05, 0.1 and 0.15.

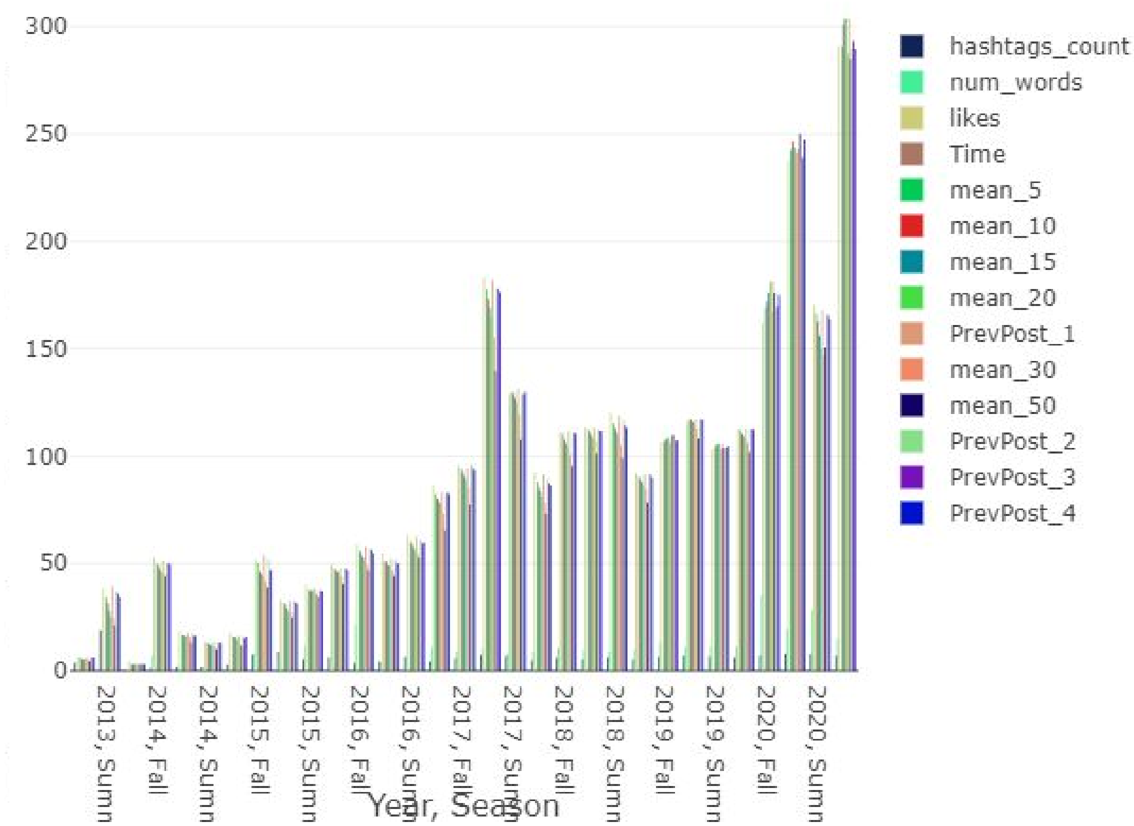
A post is considered popular if the likes achieved by that post is greater than (1+∆)* rolling average of likes over K

Result:

Below is the table that summarizes the averages of some extracted features (**Note:** Not all features are in this table as some features make more sense if they are summarize by count instead)

Year, Season	hashtags_c	num_word	likes	mean_5	mean_10	mean_15	mean_20	PrevPost_1	mean_30	mean_50	PrevPost_2	PrevPost_3	PrevPost_4
2013, Fall	2.63	0.63	11.05	11.95	12.24	11.95	11.75	11.42	11.57	11.29	11.74	11.26	11.42
2013, Spring	3.71	0.44	6.22	5.83	5.31	5.19	5.21	5.86	5.01	4.52	5.95	6.00	5.81
2013, Summer	18.60	0.20	38.20	34.32	31.16	27.79	25.09	39.20	24.13	20.85	37.00	36.20	34.40
2013, Winter	0.31	0.03	4.46	2.91	2.83	2.86	2.87	2.94	2.88	2.86	2.91	2.94	2.83
2014, Fall	1.07	6.72	52.78	49.91	48.81	47.13	46.10	51.08	45.01	44.13	50.19	49.93	49.37
2014, Spring	1.53	2.06	17.92	16.79	16.42	16.05	15.74	17.44	15.57	12.73	17.47	16.19	16.36
2014, Summer	1.61	1.76	13.49	12.89	12.43	12.03	11.71	13.22	10.88	9.92	12.91	13.04	12.71
2014, Winter	2.70	1.24	17.41	15.81	15.55	15.10	14.13	16.34	12.52	11.02	16.10	15.10	15.74
2015, Fall	7.50	7.99	51.51	50.00	46.05	45.06	43.96	53.36	41.50	38.74	51.96	46.16	46.79
2015, Spring	8.54	6.12	32.89	31.51	30.25	28.81	27.87	32.62	27.13	24.78	32.24	31.77	31.13
2015, Summer	5.37	11.50	39.81	37.41	37.20	37.23	36.91	38.13	35.98	34.35	37.43	37.06	36.94
2015, Winter	5.89	3.81	49.50	47.06	46.22	45.96	45.58	47.49	43.54	40.30	47.31	47.45	46.66
2016, Fall	3.65	21.30	58.71	55.88	53.87	52.57	51.11	57.72	49.14	46.49	56.78	56.35	54.53
2016, Spring	4.41	3.55	54.39	50.66	50.58	49.27	48.30	52.03	46.53	44.32	51.56	50.10	49.64
2016, Summer	6.20	6.77	63.09	60.16	58.41	56.95	55.81	62.16	54.30	52.72	60.68	59.27	59.56
2016, Winter	4.29	10.85	86.42	82.09	80.09	78.44	77.03	83.10	73.23	65.36	82.09	82.94	82.09
2017, Fall	5.46	8.85	95.18	93.73	92.51	90.48	89.18	93.85	84.92	77.40	96.16	93.90	93.43
2017, Spring	7.33	10.01	183.15	177.61	173.05	168.60	164.50	181.95	155.24	139.34	178.03	177.91	176.39
2017, Summer	6.46	7.95	129.18	129.71	128.87	126.98	124.58	130.77	119.14	107.65	129.28	128.50	129.70
2017, Winter	4.73	8.84	92.04	87.94	85.43	83.12	81.22	91.46	78.11	72.84	89.36	87.13	86.09
2018, Fall	5.79	10.71	111.26	110.22	107.75	105.86	104.20	111.33	100.73	95.41	110.46	110.84	110.31
2018, Spring	5.08	9.56	113.53	112.10	111.14	109.97	108.65	112.90	106.81	100.85	112.37	111.69	111.30
2018, Summer	6.09	8.74	119.92	115.01	112.85	110.65	108.62	118.42	105.13	98.87	116.53	114.34	113.18
2018, Winter	5.25	9.45	92.27	90.25	88.96	87.81	86.62	91.36	84.35	78.16	91.18	91.03	89.34
2019, Fall	6.37	13.10	106.09	106.90	107.69	108.28	108.62	106.17	109.16	109.49	106.57	107.09	107.25
2019, Spring	6.83	10.97	115.68	116.84	116.96	115.90	114.76	116.96	112.74	108.23	117.11	116.70	116.71
2019, Summer	6.37	11.48	102.92	104.13	105.00	105.59	105.81	103.15	105.56	103.50	103.79	103.92	104.51
2019, Winter	5.97	11.11	112.72	112.19	110.39	109.25	108.11	112.39	106.03	101.66	111.95	112.59	112.33
2020, Fall	6.97	34.94	162.16	168.96	171.94	176.07	181.19	166.66	181.38	176.04	168.37	169.81	174.61
2020, Spring	7.88	19.41	237.22	242.45	246.48	243.77	242.79	241.13	242.99	249.94	237.21	238.98	247.30
2020, Summer	7.50	28.24	170.32	166.38	162.54	156.10	149.90	167.52	146.65	150.51	166.15	165.43	163.51
2020, Winter	6.88	14.97	290.77	290.69	300.57	308.49	311.18	287.53	311.48	284.62	289.70	293.34	289.47

Visualizing this table:



Comment:

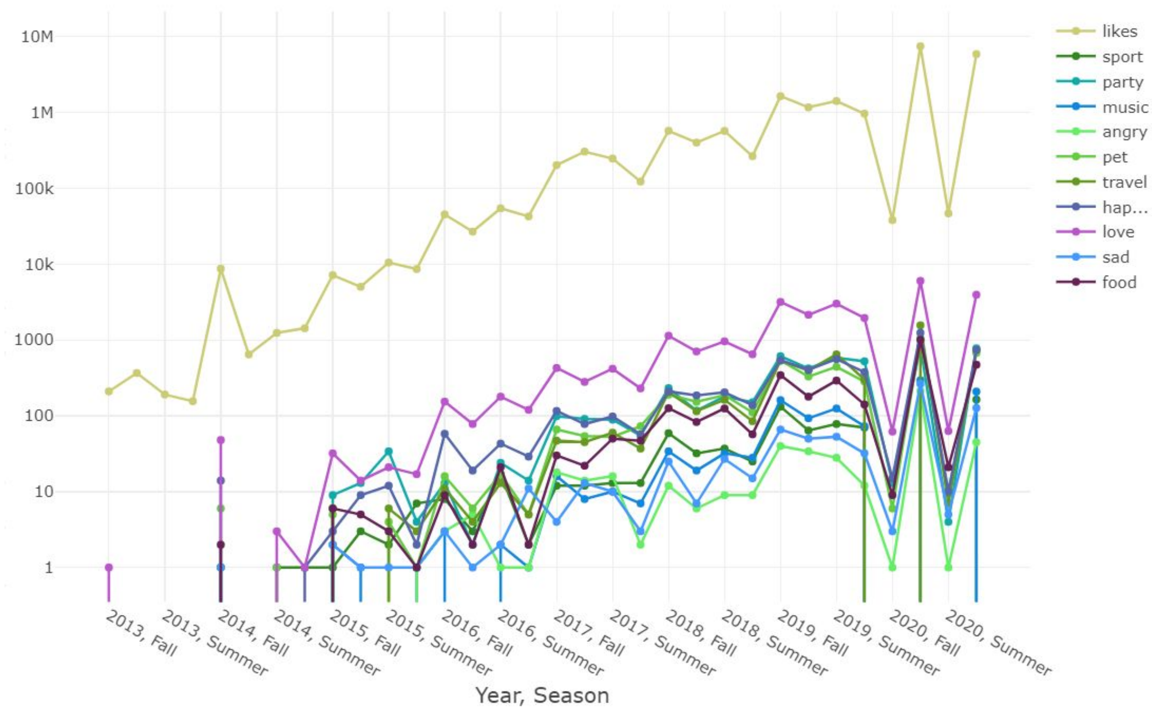
It can be noted that as the overall average of the features presented in the previous table have increased over the season per year.

Users are including more words/hashtags in their caption thus increasing the avg of likes acquired per post

Below is another table that summarizes the count of the features that were not included in the average table from before

Year, Season	likes	sport	party	music	angry	pet	travel	happy	love	sad	food
2013, Fall	210.00	-	-	-	-	-	-	-	1.00	-	-
2013, Spring	367.00	-	-	-	-	-	-	-	-	-	-
2013, Summer	191.00	-	-	-	-	-	-	-	-	-	-
2013, Winter	156.00	-	-	-	-	-	-	-	-	-	-
2014, Fall	8,709.00	1.00	1.00	-	-	6.00	-	14.00	48.00	1.00	2.00
2014, Spring	645.00	-	-	-	-	-	-	-	-	-	-
2014, Summer	1,241.00	1.00	-	-	-	1.00	-	-	3.00	-	-
2014, Winter	1,428.00	1.00	-	-	-	-	-	1.00	1.00	-	-
2015, Fall	7,160.00	1.00	9.00	2.00	2.00	5.00	6.00	3.00	32.00	2.00	6.00
2015, Spring	5,032.00	3.00	13.00	1.00	-	-	-	9.00	14.00	1.00	5.00
2015, Summer	10,509.00	2.00	34.00	-	-	4.00	6.00	12.00	21.00	1.00	3.00
2015, Winter	8,613.00	7.00	4.00	1.00	1.00	1.00	3.00	2.00	17.00	1.00	1.00
2016, Fall	45,324.00	8.00	13.00	3.00	3.00	16.00	11.00	58.00	154.00	3.00	9.00
2016, Spring	26,869.00	3.00	2.00	-	5.00	6.00	4.00	19.00	78.00	1.00	2.00
2016, Summer	54,507.00	17.00	24.00	2.00	1.00	17.00	13.00	43.00	179.00	2.00	21.00
2016, Winter	42,604.00	2.00	14.00	1.00	1.00	5.00	5.00	29.00	120.00	11.00	2.00
2017, Fall	202,359.00	12.00	99.00	16.00	18.00	66.00	47.00	116.00	428.00	4.00	30.00
2017, Spring	305,132.00	12.00	91.00	8.00	14.00	54.00	45.00	78.00	280.00	13.00	22.00
2017, Summer	246,866.00	13.00	89.00	10.00	16.00	53.00	60.00	98.00	418.00	10.00	50.00
2017, Winter	122,692.00	13.00	55.00	7.00	2.00	73.00	37.00	57.00	230.00	3.00	47.00
2018, Fall	574,119.00	59.00	232.00	34.00	12.00	190.00	205.00	210.00	1,138.00	25.00	126.00
2018, Spring	400,771.00	32.00	117.00	19.00	6.00	153.00	115.00	186.00	706.00	7.00	83.00
2018, Summer	571,521.00	37.00	180.00	32.00	9.00	187.00	163.00	204.00	960.00	27.00	125.00
2018, Winter	265,264.00	25.00	151.00	28.00	9.00	111.00	85.00	138.00	648.00	15.00	57.00
2019, Fall	1,642,536.00	132.00	612.00	161.00	40.00	535.00	535.00	543.00	3,176.00	66.00	344.00
2019, Spring	1,167,877.00	64.00	422.00	93.00	34.00	330.00	397.00	405.00	2,149.00	50.00	179.00
2019, Summer	1,416,178.00	78.00	582.00	125.00	28.00	444.00	646.00	560.00	3,016.00	53.00	292.00
2019, Winter	964,876.00	70.00	524.00	73.00	12.00	283.00	302.00	377.00	1,952.00	32.00	141.00
2020, Fall	38,108.00	-	13.00	-	1.00	6.00	-	15.00	62.00	3.00	9.00
2020, Spring	7,467,236.00	154.00	824.00	295.00	213.00	1,052.00	1,562.00	1,244.00	6,020.00	264.00	1,004.00
2020, Summer	46,668.00	-	4.00	-	1.00	10.00	7.00	10.00	63.00	5.00	21.00
2020, Winter	5,885,836.00	164.00	777.00	209.00	45.00	675.00	736.00	740.00	3,956.00	127.00	472.00

Visualizing the table



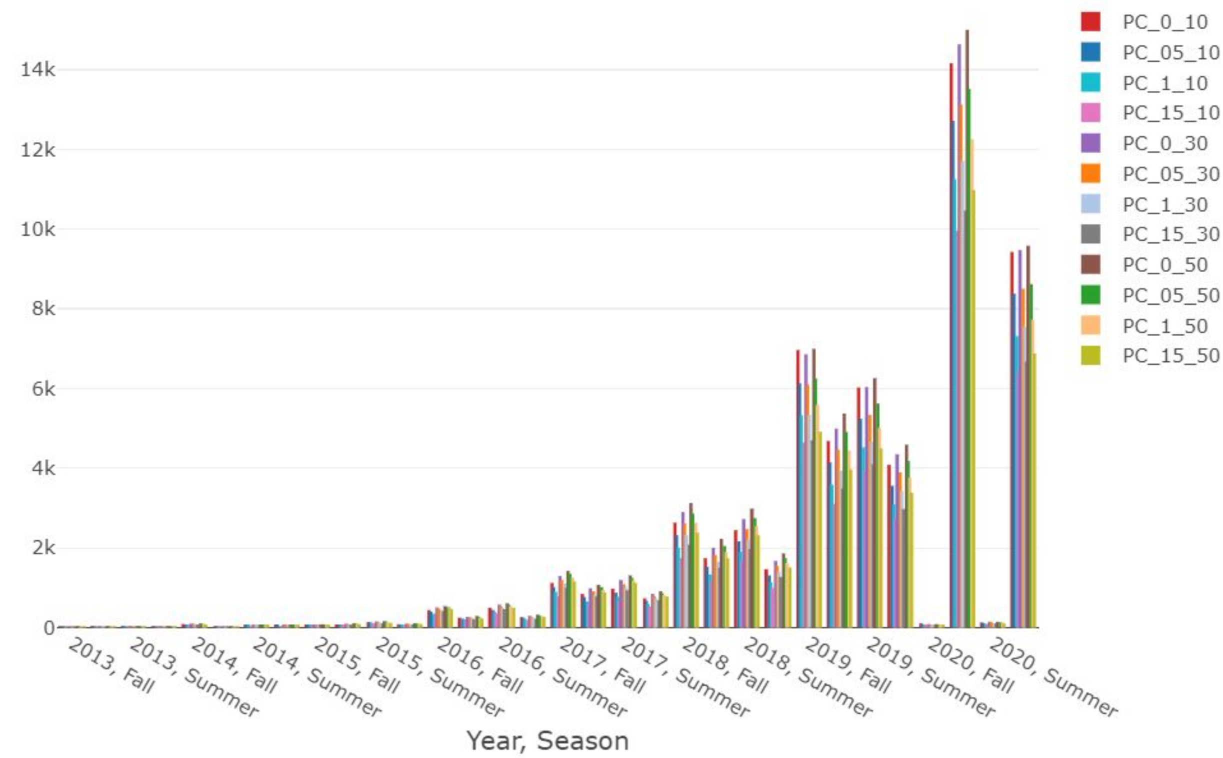
Comment:

The number of likes seem to have direct correlation with the emoji’s used

Below is another table that summarizes the sum of the popularity features:

Year, Seasor	PC_0_1(	PC_05_1(	PC_1_1(	PC_15_1(	PC_0_3(	PC_05_3(	PC_1_3(	PC_15_3(	PC_0_5(	PC_05_5(	PC_1_5(	PC_15_5(
2013, Fall	11	8	8	3	11	8	6	4	11	10	4	4
2013, Spring	33	33	31	30	35	33	29	27	33	33	29	27
2013, Summer	2	1	1	1	5	4	3	3	5	5	5	5
2013, Winter	14	14	12	12	17	14	11	10	16	13	10	10
2014, Fall	91	73	64	58	101	91	82	73	104	93	85	82
2014, Spring	21	18	16	13	22	20	19	16	28	26	26	24
2014, Summer	48	46	43	37	60	58	54	52	70	64	60	56
2014, Winter	44	42	35	33	54	51	49	46	60	58	57	54
2015, Fall	79	75	69	64	78	73	67	61	84	77	74	69
2015, Spring	75	68	64	60	97	87	81	73	112	107	98	88
2015, Summer	141	131	121	106	157	142	123	107	164	151	134	121
2015, Winter	87	76	66	57	99	91	86	73	112	108	99	89
2016, Fall	443	400	351	317	509	475	438	409	542	513	489	466
2016, Spring	251	226	207	192	272	259	237	213	295	275	251	229
2016, Summer	500	442	394	362	580	539	503	466	612	576	540	502
2016, Winter	266	248	219	194	298	276	250	226	326	305	285	264
2017, Fall	1,122	1,019	900	793	1,297	1,197	1,112	994	1,426	1,351	1,256	1,158
2017, Spring	847	767	657	593	988	914	832	774	1,077	1,018	950	882
2017, Summer	977	876	775	671	1,196	1,090	1,020	939	1,312	1,245	1,188	1,124
2017, Winter	735	672	599	534	848	791	740	690	917	857	819	779
2018, Fall	2,636	2,316	2,006	1,748	2,902	2,616	2,318	2,085	3,128	2,863	2,630	2,383
2018, Spring	1,743	1,531	1,334	1,162	2,007	1,821	1,655	1,502	2,231	2,050	1,895	1,742
2018, Summer	2,445	2,170	1,902	1,667	2,725	2,473	2,208	1,974	2,985	2,750	2,545	2,316
2018, Winter	1,468	1,313	1,142	996	1,675	1,549	1,404	1,274	1,863	1,743	1,626	1,512
2019, Fall	6,966	6,128	5,331	4,647	6,858	6,103	5,345	4,701	6,995	6,251	5,585	4,918
2019, Spring	4,684	4,146	3,586	3,108	4,994	4,463	3,945	3,494	5,374	4,910	4,444	3,968
2019, Summer	6,026	5,241	4,523	3,950	6,037	5,335	4,665	4,102	6,259	5,624	5,016	4,495
2019, Winter	4,083	3,561	3,099	2,712	4,352	3,896	3,424	2,979	4,591	4,177	3,762	3,386
2020, Fall	105	88	65	56	91	74	62	55	89	78	73	64
2020, Spring	14,160	12,714	11,252	9,961	14,633	13,125	11,713	10,462	15,000	13,518	12,252	10,978
2020, Summer	131	114	96	85	150	133	113	104	149	134	119	102
2020, Winter	9,426	8,379	7,313	6,383	9,474	8,501	7,545	6,675	9,582	8,621	7,726	6,880

Visualizing the table:

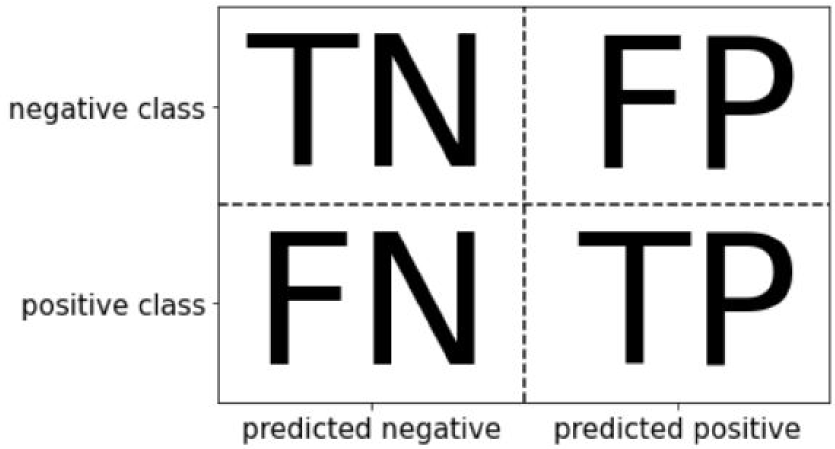


Overall there is a upward trend of the overall popularity of instagram posts

How do the models perform on the original data vs the new + original data??

Approach:

In binary classification we have two classes. The class we are after (positive class) and the other negative class. In our model the positive class is popular and the negative class is non-popular. It is easier to visualize it using this confusion matrix below



Note: On the main diagonal, are the numbers of correctly predicted samples. The off-diagonal has the mistakes.

For assessing the model performance the following metrics are used:

i) Precision: this metic explains how many samples predicted positive are actually positive,can be defined by the following equation

$$precision = \frac{TP}{TP+FP}$$

ii) Recall :this metric explains how many actual positive samples do we catch, can be defined by the following equation

$$recall = \frac{TP}{TP+FN}$$

iii) Recall: Combine precision and recall into one score, can be explained by the following equation

$$f_1 = 2 \cdot \frac{precision \cdot recall}{precision+recall}$$

The paper uses two machine learning models RandomForest and XGBoost, we decided to add another machine learning model which is LightGBM[3].

RandomForest and XGBoost were used for original data and original+new data LightGBM was used on the original+new data only.

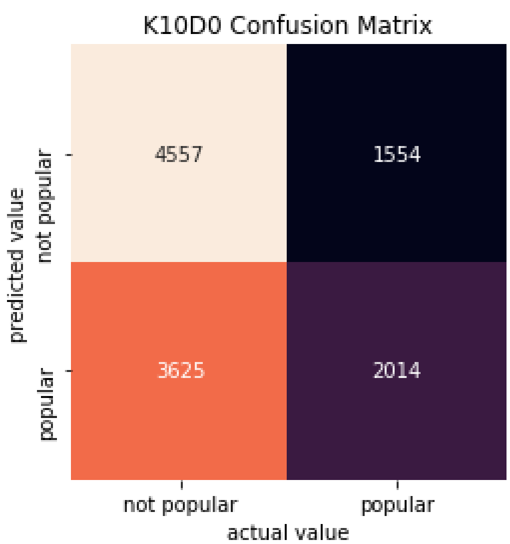
From the pre-processing step we ended up having 12 vector columns where each column is the popularity feature classification based on K and delta.
Each of the target vectors was used to train/validate the three machine learning models mentioned above.

Result:

a)Original data

1) K=10, delta=0:

   <u>*RandomForestClassifier:*</u>

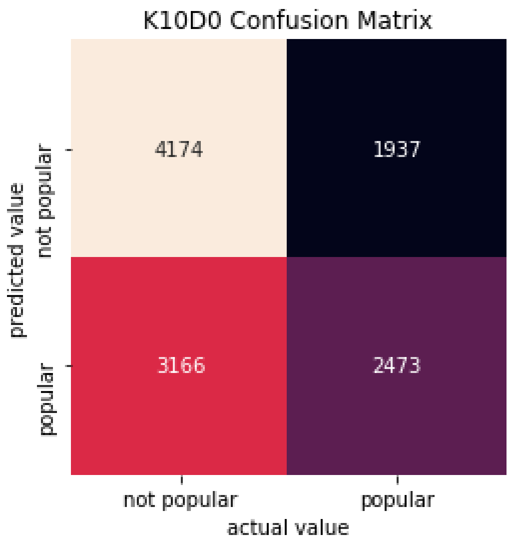


Recall=0.36

Precision=0.56

f1-score=0.54

<u>*XGBClassifier:*</u>



Recall=0.43

Precision=0.56

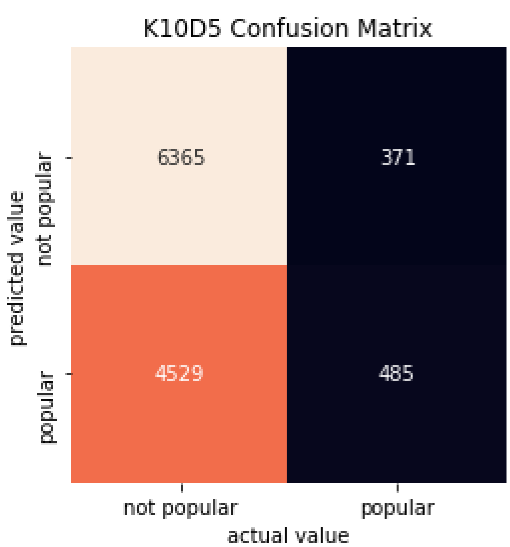
f1-score=0.56

Comment:

XGBClassifier performed better than the random forest for this particular target vector

2) K=10, delta=0.5

<u>RandomForestClassifier:</u>

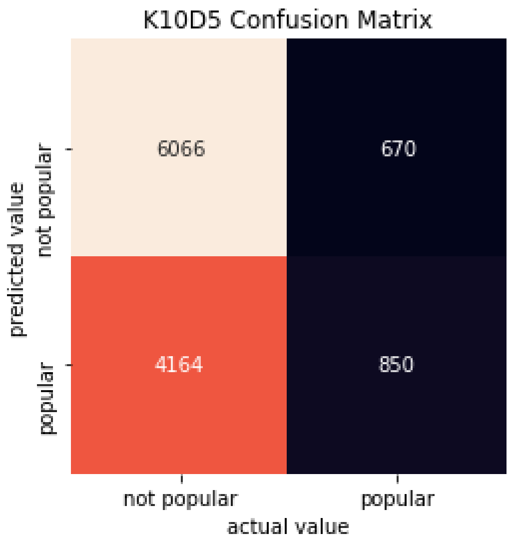


Recall=0.1

Precision=0.57

f1-score=0.48

XGBClassifier:



Recall=0.17

Precision=0.56

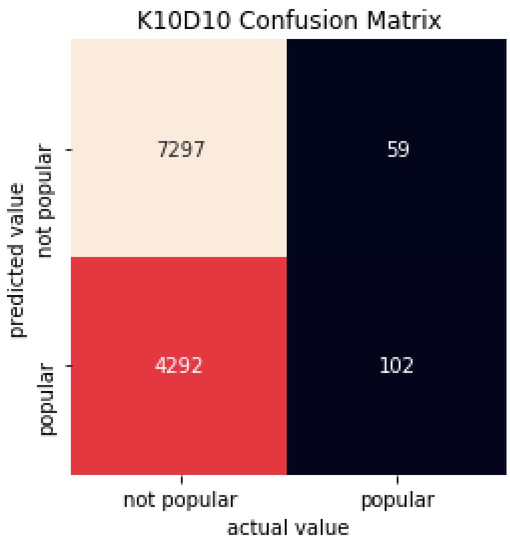
f1-score=0.52

Comment:

XGBClassifier performed better than the random forest for this particular target vector

3) K=10, delta=0.1

RandomForestClassifier:

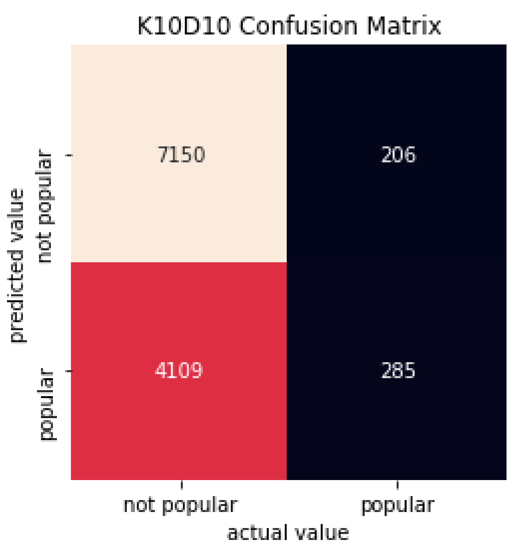


Recall=0.02

Precision=0.63

f1-score=0.5

XGBClassifier:



Recall=0.06

Precision=0.58

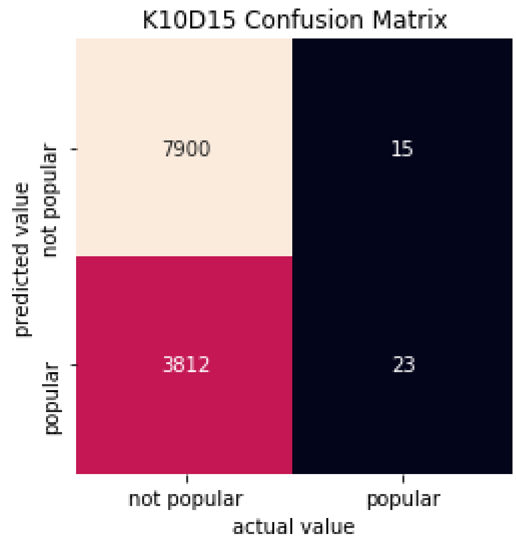
f1-score=0.52

Comment:

XGBClassifier performed better than the random forest for this particular target vector

4) K=10, delta=0.15

RandomForestClassifier:

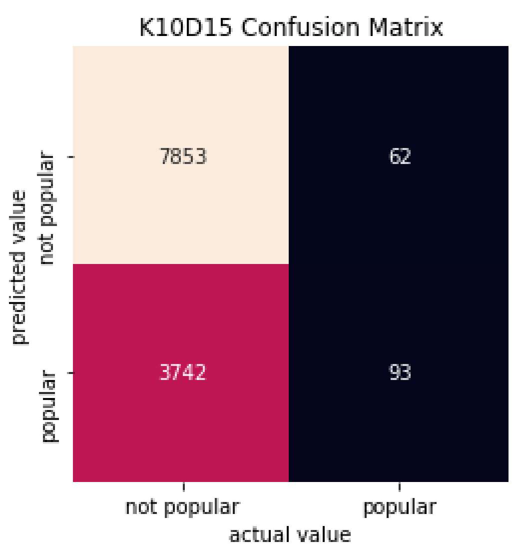


Recall=0.01

Precision=0.6

f1-score=0.55

XGBClassifier:



Recall=0.02

Precision=0.6

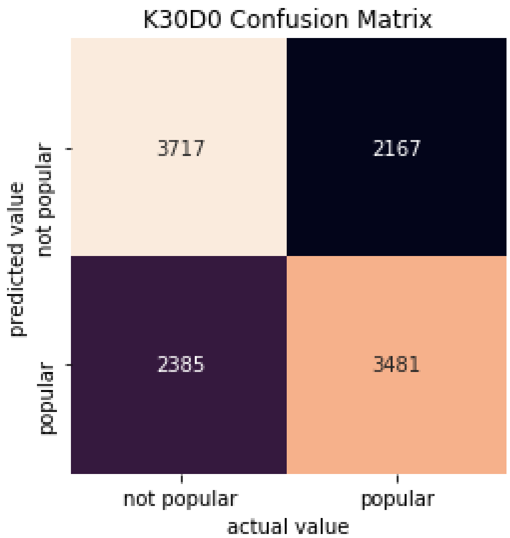
f1-score=0.56

Comment:

XGBClassifier performed better than the random forest for this particular target vector

5) K=30, delta=0

RandomForestClassifier:

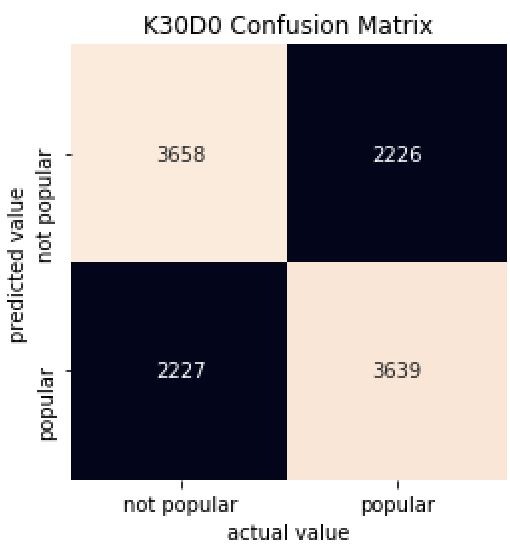


Recall=0.6

Precision=0.62

f1-score=0.61

XGBClassifier:



Recall=0.62

Precision=0.62

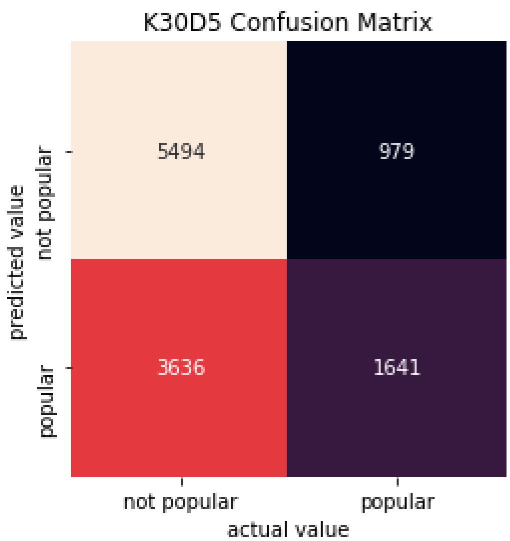
f1-score=0.62

Comment:

XGBClassifier performed better than the random forest for this particular target vector

6) K=30, delta=0.5

RandomForestClassifier:

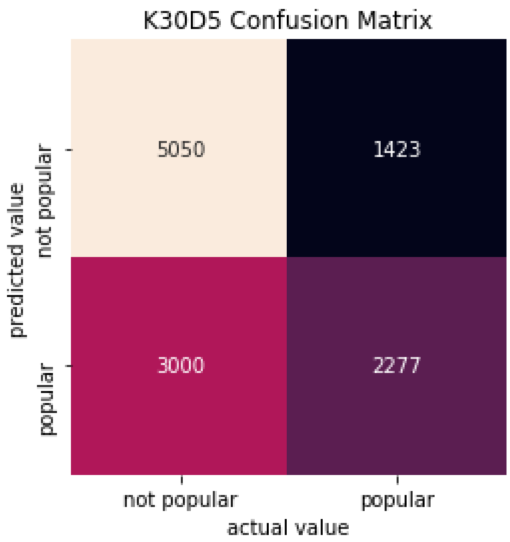


Recall=0.31

Precision=0.62

f1-score=0.57

XGBClassifier:



Recall=0.43

Precision=0.62

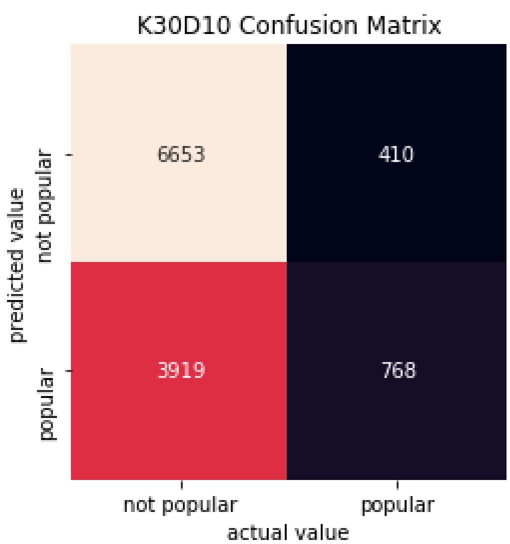
f1-score=0.61

Comment:

XGBClassifier performed better than the random forest for this particular target vector

7) K=30, delta=0.1

<u>*RandomForestClassifier:*</u>

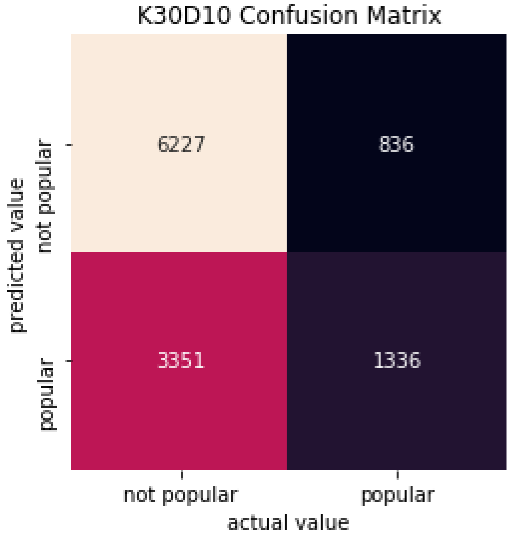


Recall=0.16

Precision=0.65

f1-score=0.56

XGBClassifier:



Recall=0.29

Precision=0.62

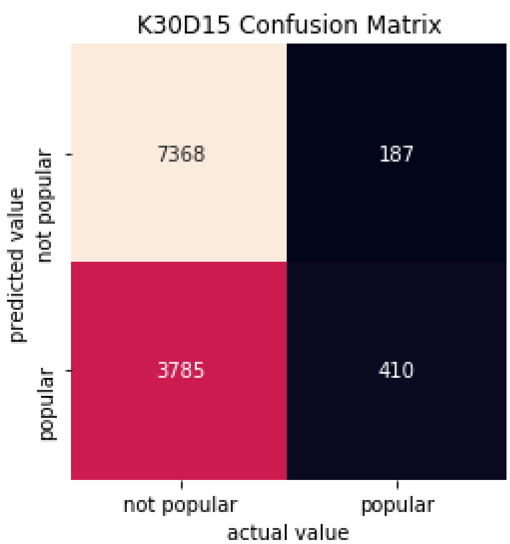
f1-score=0.61

Comment:

XGBClassifier performed better than the random forest for this particular target vector

8) K=30, delta=0.15

RandomForestClassifier:

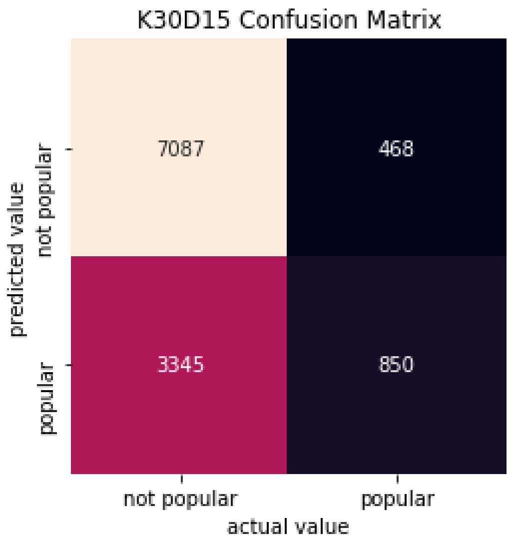


Recall=0.1

Precision=0.69

f1-score=0.57

XGBClassifier:



Recall=0.2

Precision=0.64

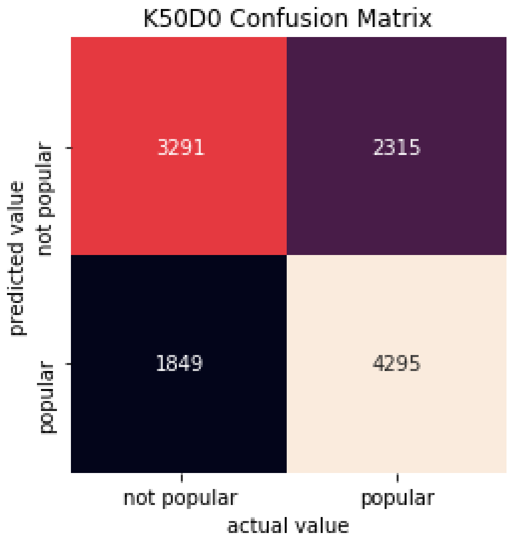
f1-score=0.62

Comment:

XGBClassifier performed better than the random forest for this particular target vector

9) K=50, delta=0

RandomForestClassifier:

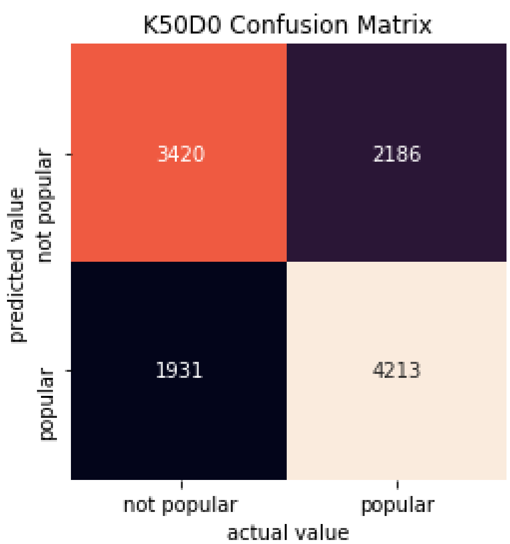


Recall=0.7

Precision=0.65

f1-score=0.64

XGBClassifier:



Recall=0.69

Precision=0.66

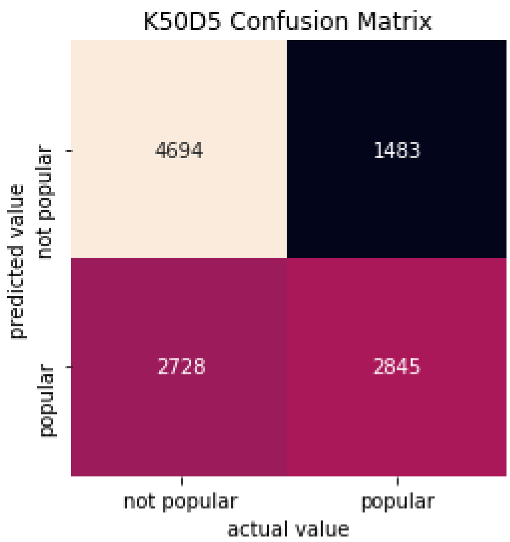
f1-score=0.65

Comment:

Both models have similar metrics

10)K=50, delta=0.5

RandomForestClassifier:

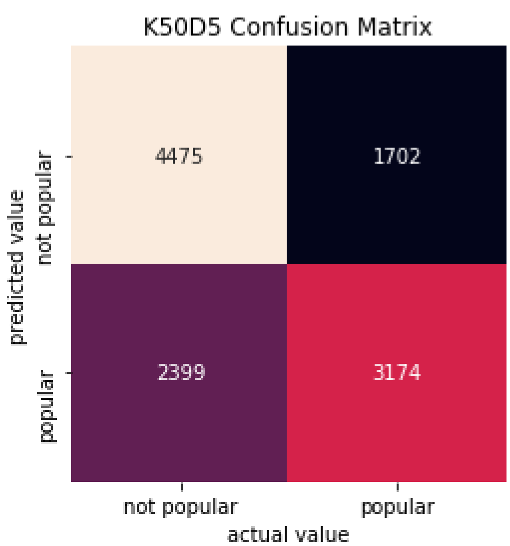


Recall=0.51

Precision=0.66

f1-score=0.64

XGBClassifier:



Recall=0.57

Precision=0.65

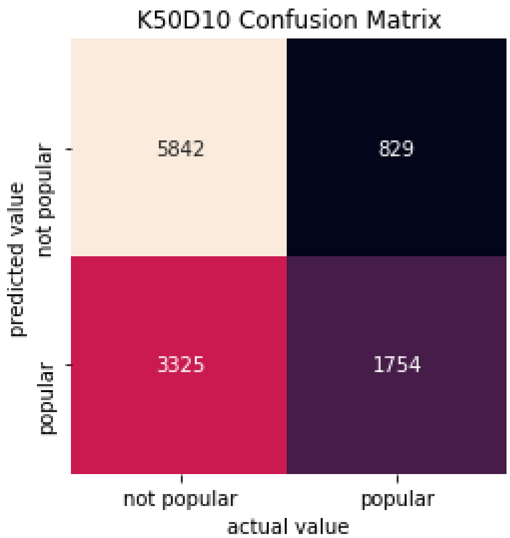
f1-score=0.65

Comment:

XGBClassifier performed better than the random forest for this particular target vector

11)K=50, delta=0.1

RandomForestClassifier:

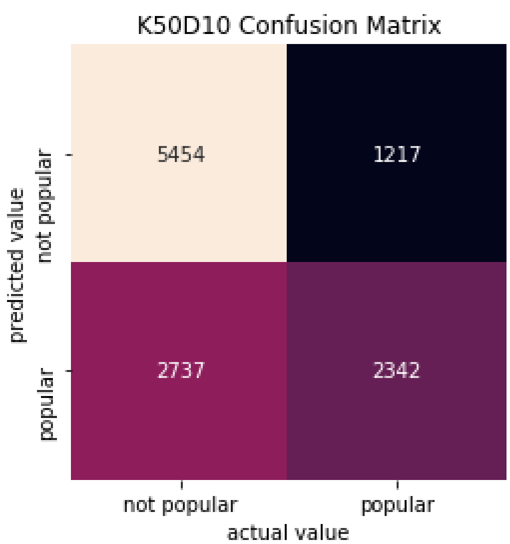


Recall=0.34

Precision=0.68

f1-score=0.62

XGBClassifier:



Recall=0.46

Precision=0.66

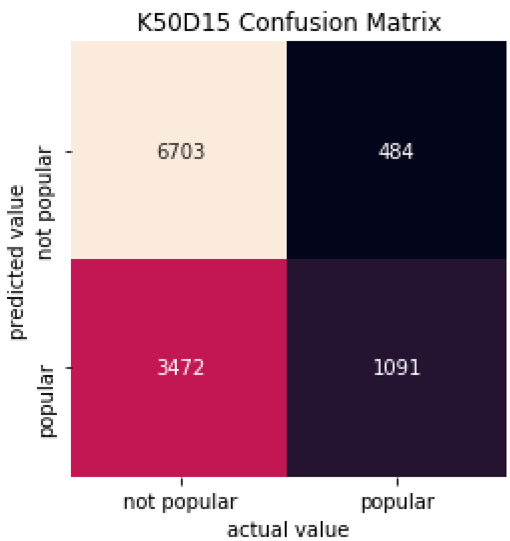
f1-score=0.65

Comment:

XGBClassifier performed better than the random forest for this particular target vector

12)K=50, delta=0.15

RandomForestClassifier:

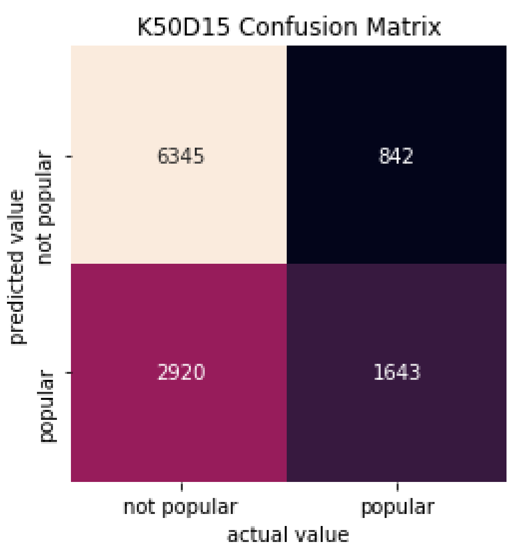


Recall=0.23

Precision=0.69

f1-score=0.61

XGBClassifier:



Recall=0.36

Precision=0.66

f1-score=0.65

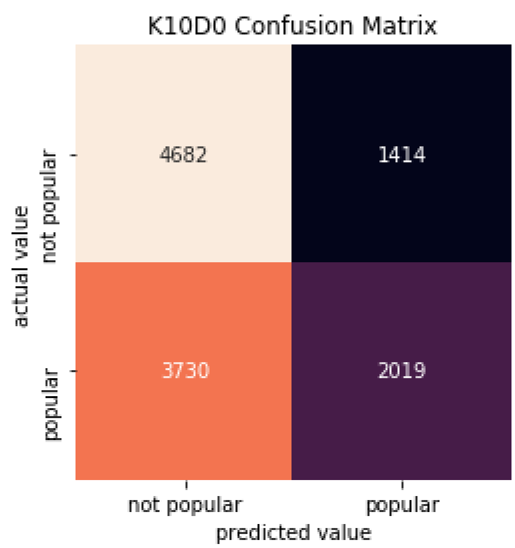
Comment:

XGBClassifier performed better than the random forest for this particular target vector

b)Original data + new data:

1) K=10, delta=0:

RandomForestClassifier:

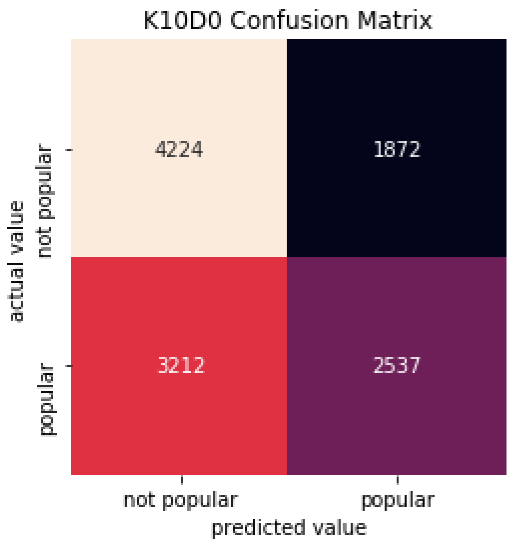


Recall=0.35

Precision=0.59

f1-score=0.5

XGBClassifier:

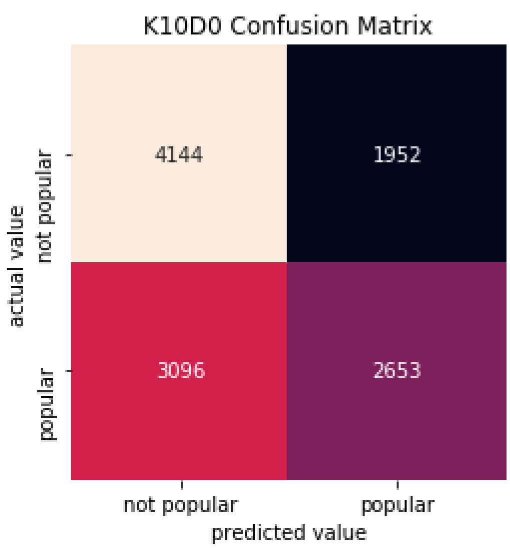


Recall=0.44

Precision=0.57

f1-score=0.56

LightGBM:



Recall=0.46

Precision=0.58

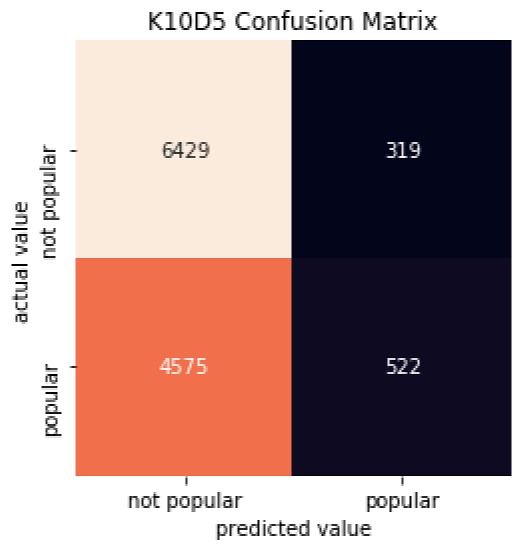
f1-score=0.57

Comment:

LGBM Classifier performedr better that the RandomForest, and XGB Classifiers

2) K=10, delta=0.5

RandomForestClassifier:

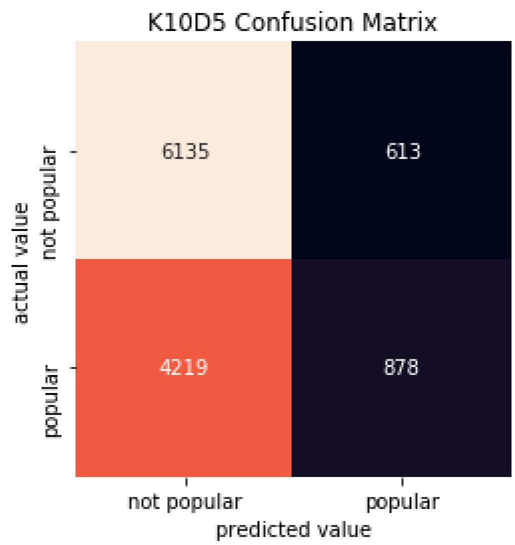


Recall=0.1

Precision=0.62

f1-score=0.48

XGBClassifier:

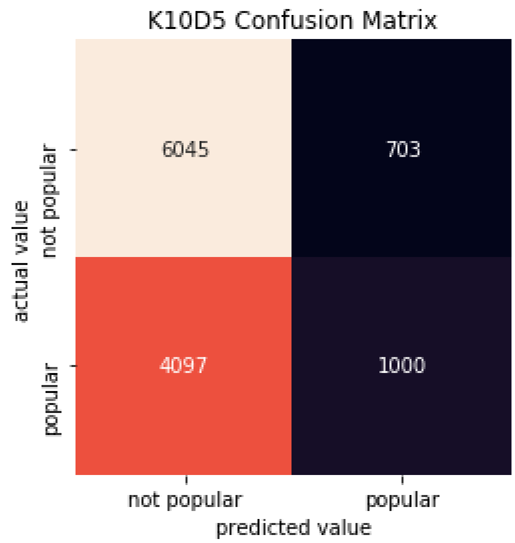


Recall=0.17

Precision=0.59

f1-score=0.52

LightGBM:



Recall=0.2

Precision=0.59

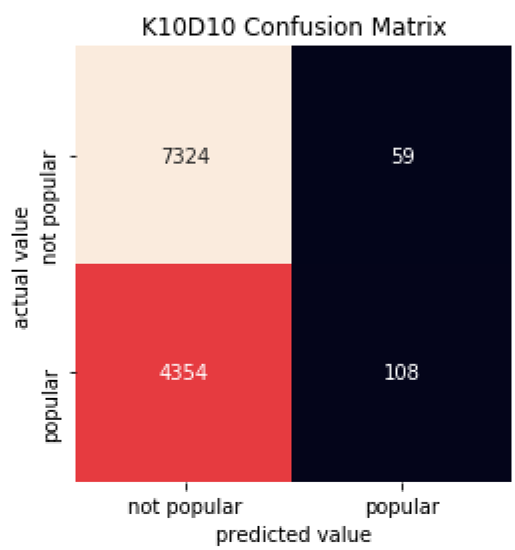
f1-score=0.53

Comment:

LGBM Classifier performedr better that the RandomForest, and XGB Classifiers

3) K=10, delta=0.1

RandomForestClassifier:

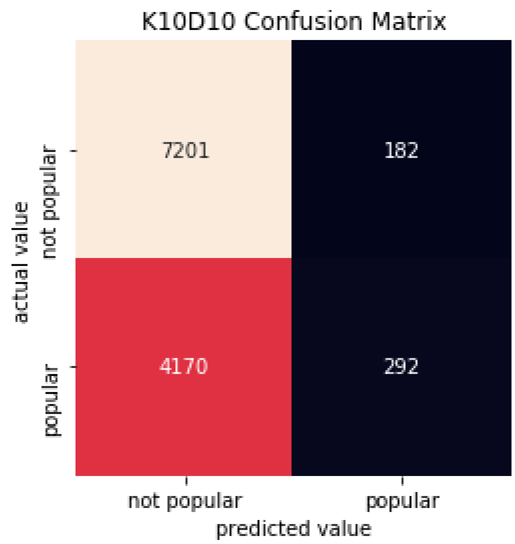


Recall=0.02

Precision=0.65

f1-score=0.5

XGBClassifier:

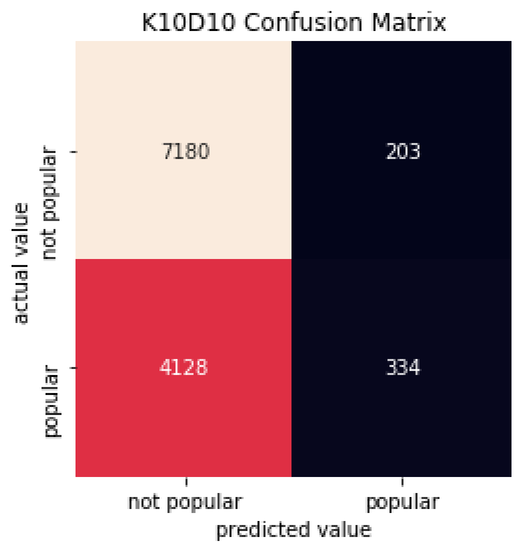


Recall=0.07

Precision=0.62

f1-score=0.52

LightGBM:



Recall=0.07

Precision=0.62

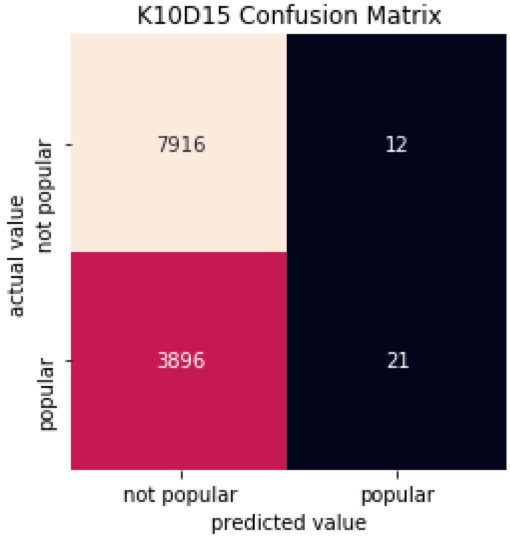
f1-score=0.53

Comment:

All three models performed the same for this particular target vector

4) K=10, delta=0.15

RandomForestClassifier:

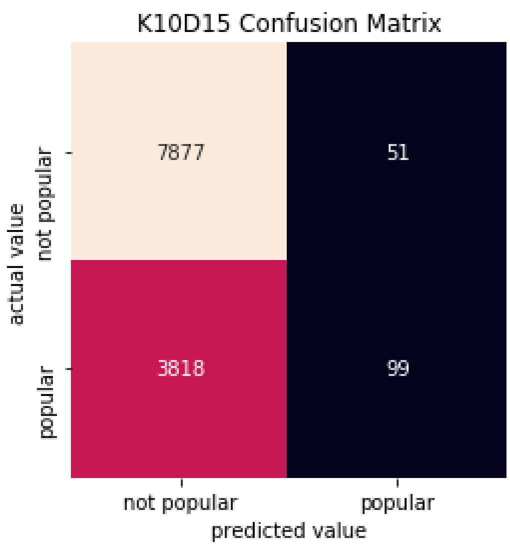


Recall=0.01

Precision=0.64

f1-score=0.54

XGBClassifier:

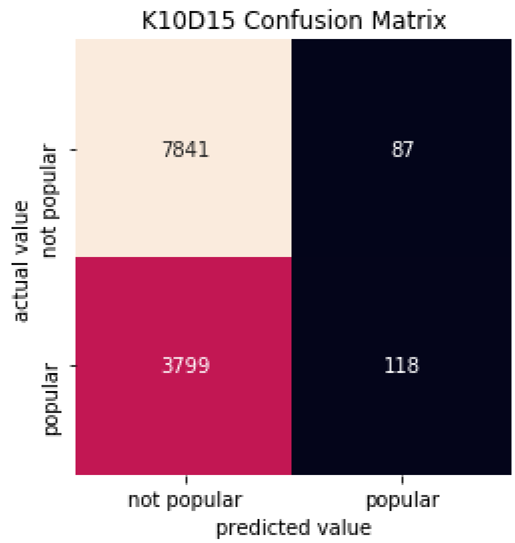


Recall=0.03

Precision=0.66

f1-score=0.55

LightGBM:



Recall=0.03

Precision=0.58

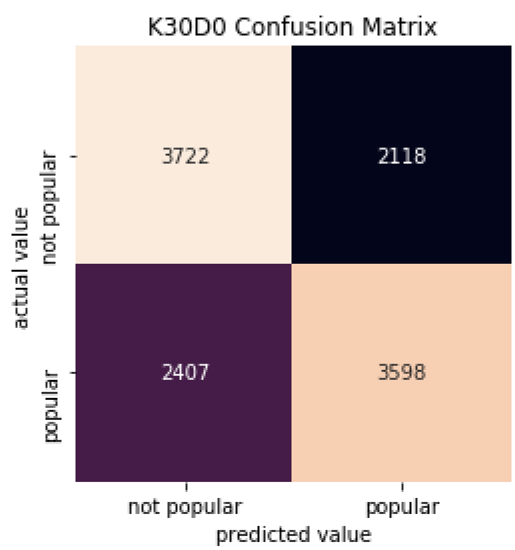
f1-score=0.56

Comment:

LGBM classifier is slightly better than RanodForest and XGB Classifiers

5) K=30, delta=0

RandomForestClassifier:

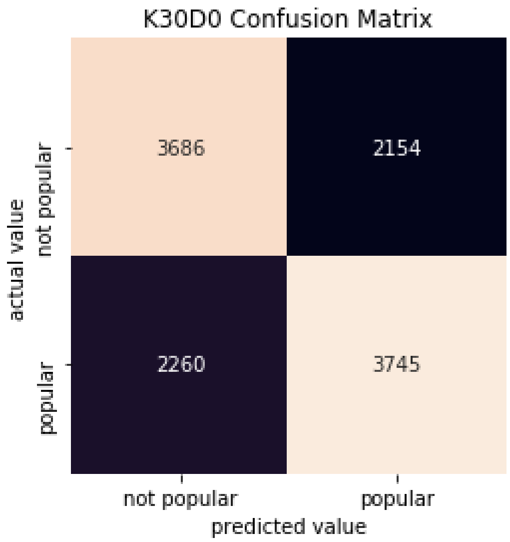


Recall=0.6

Precision=0.63

f1-score=0.62

XGBClassifier:

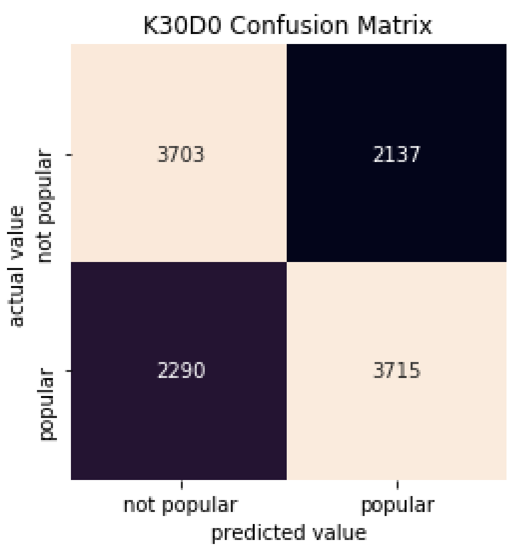


Recall=0.62

Precision=0.63

f1-score=0.63

LightGBM:



Recall=0.62

Precision=0.63

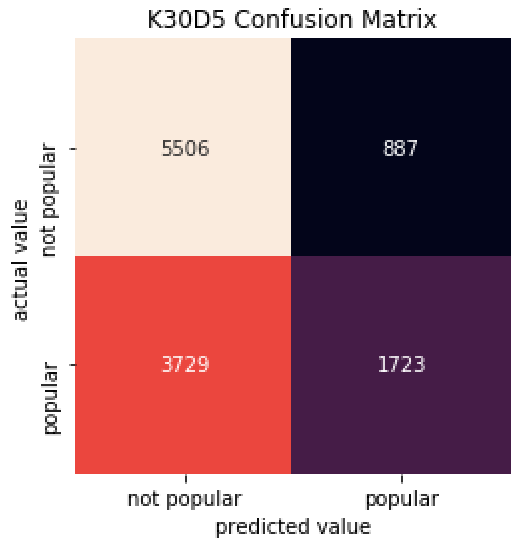
f1-score=0.62

Comment:

Both XGB and LGBM Classifiers performed best

6) K=30, delta=0.5

RandomForestClassifier:

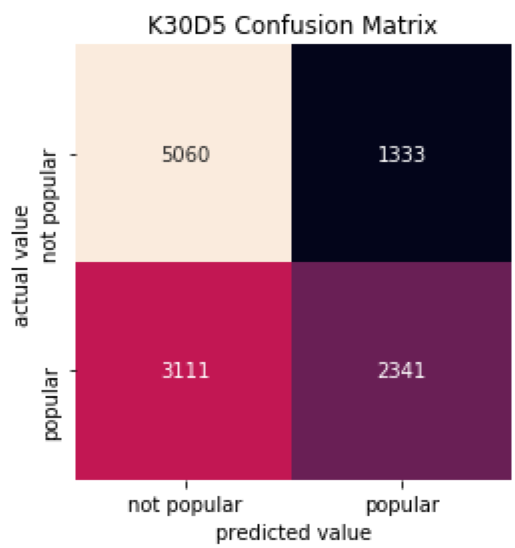


Recall=0.31

Precision=0.66

f1-score=0.58

XGBClassifier:

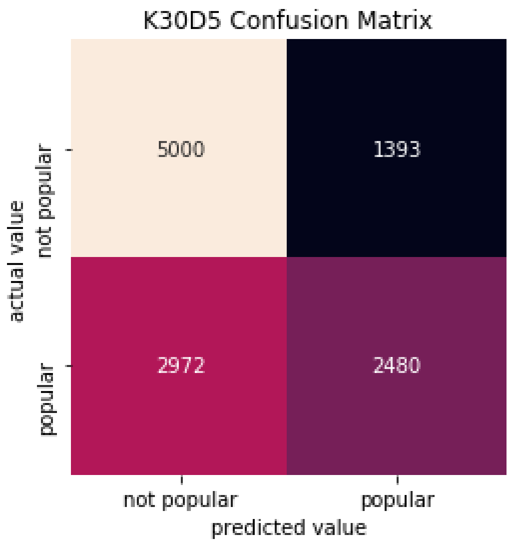


Recall=0.43

Precision=0.64

f1-score=0.61

LightGBM:



Recall=0.45

Precision=0.64

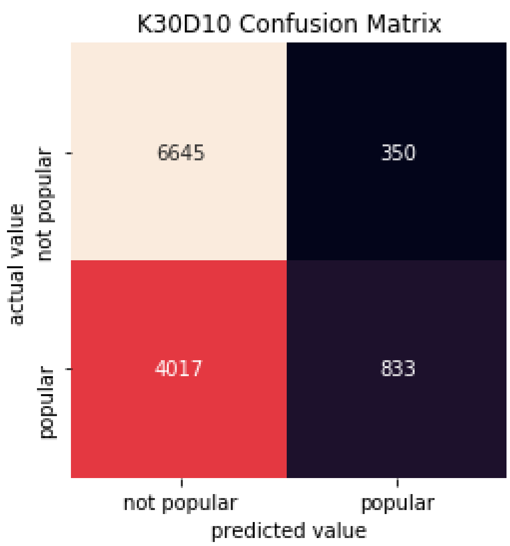
f1-score=0.62

Comment:

LGBM performed best for this particular target vector

7) K=30, delta=0.1

<u>*RandomForestClassifier:*</u>

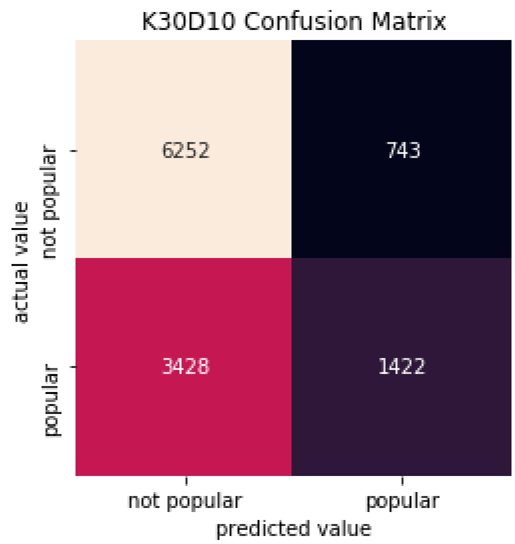


Recall=0.17

Precision=0.7

f1-score=0.56

XGBClassifier:

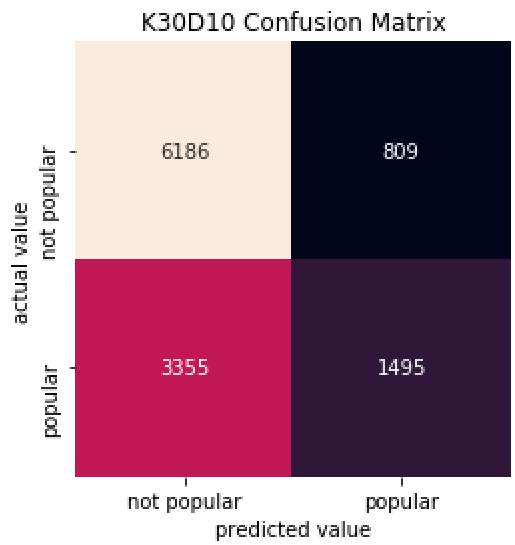


Recall=0.3

Precision=0.66

f1-score=0.61

LightGBM:



Recall=0.31

Precision=0.65

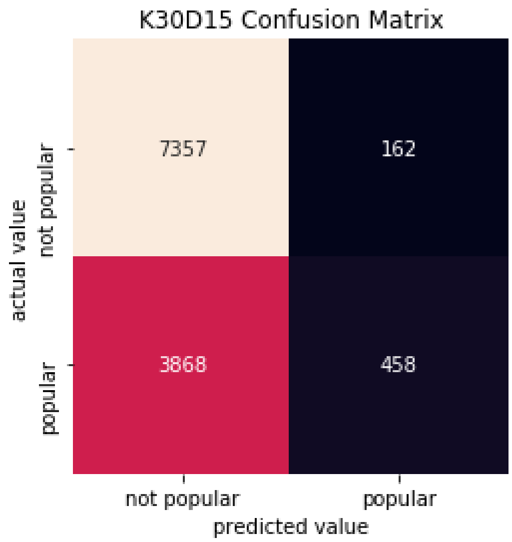
f1-score=0.61

Comment:

XGB Classifier performed better than the other two models

8) K=30, delta=0.15

RandomForestClassifier:

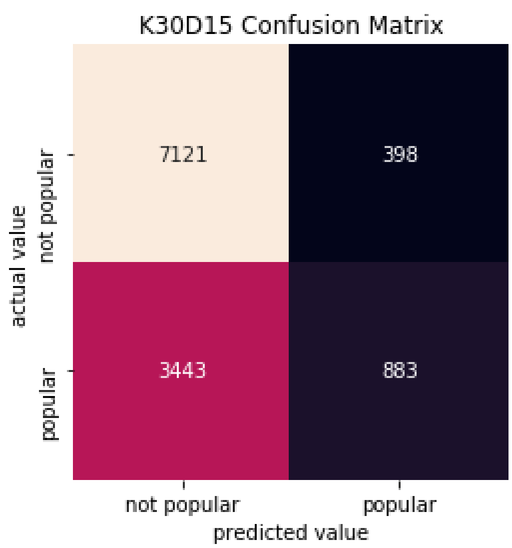


Recall=0.1

Precision=0.74

f1-score=0.57

XGBClassifier:

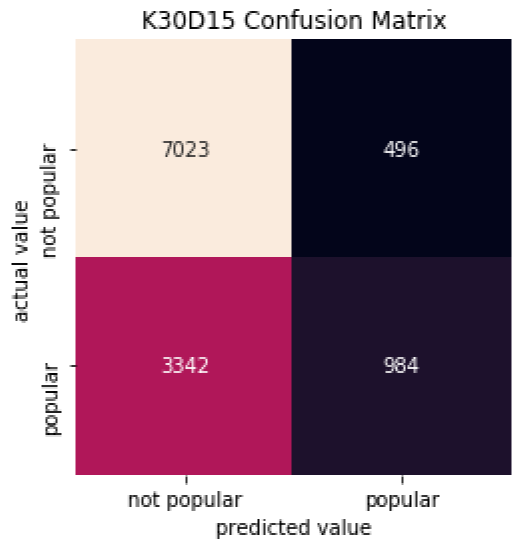


Recall=0.2

Precision=0.7

f1-score=0.62

LightGBM:



Recall=0.23

Precision=0.66

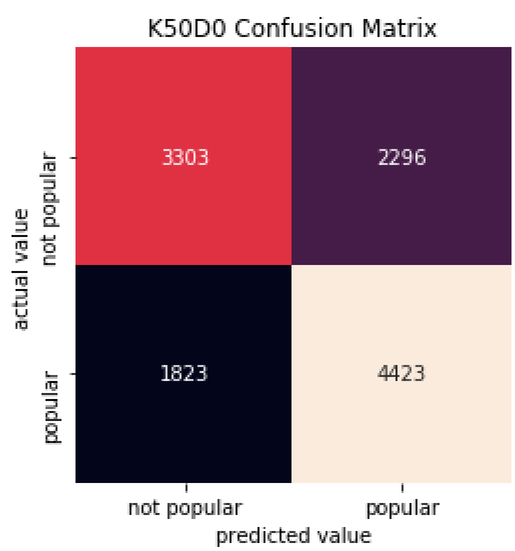
f1-score=0.62

Comment:

XGB Classifier performed better than the other two models

9) K=50, delta=0

RandomForestClassifier:

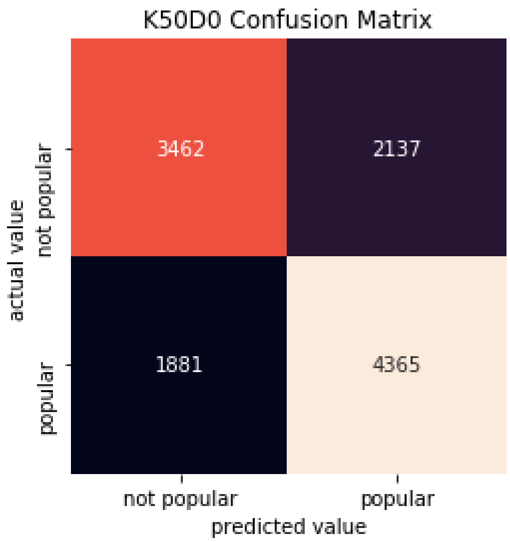


Recall=0.71

Precision=0.66

f1-score=0.65

XGBClassifier:

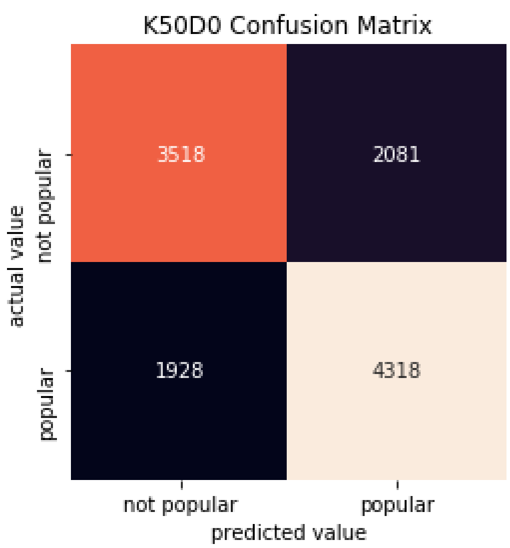


Recall=0.7

Precision=0.67

f1-score=0.66

LightGBM:



Recall=0.7

Precision=0.67

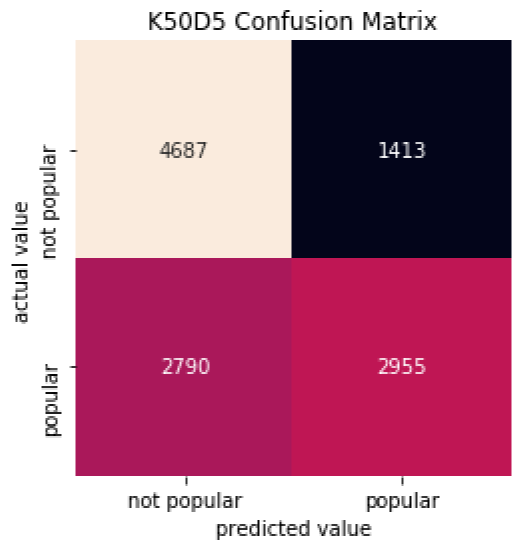
f1-score=0.66

Comment:

Both XGB and LGBM performed best for this particular target vector

10)K=50, delta=0.5

RandomForestClassifier:

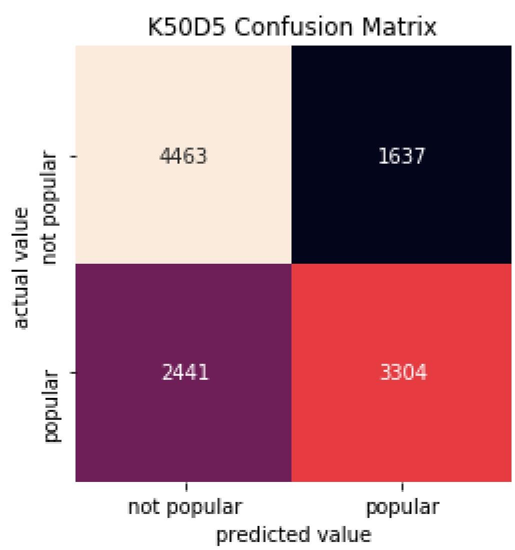


Recall=0.51

Precision=0.68

f1-score=0.64

XGBClassifier:

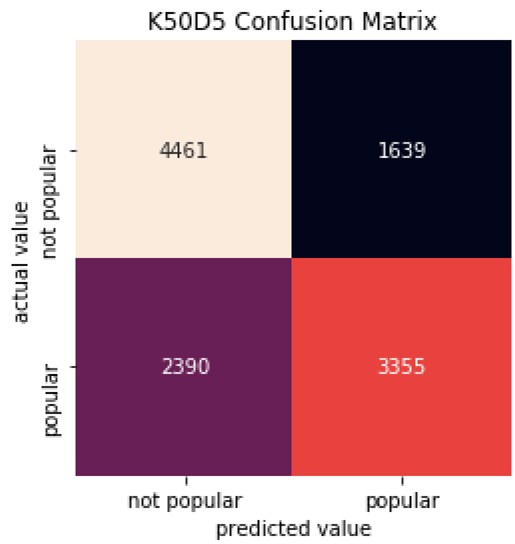


Recall=0.58

Precision=0.67

f1-score=0.65

LightGBM:



Recall=0.58

Precision=0.67

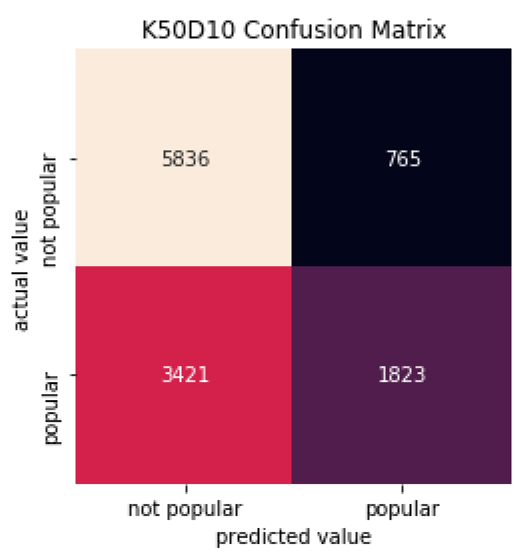
f1-score=0.66

Comment:

LGBM performed better than the other two models

11)K=50, delta=0.1

RandomForestClassifier:

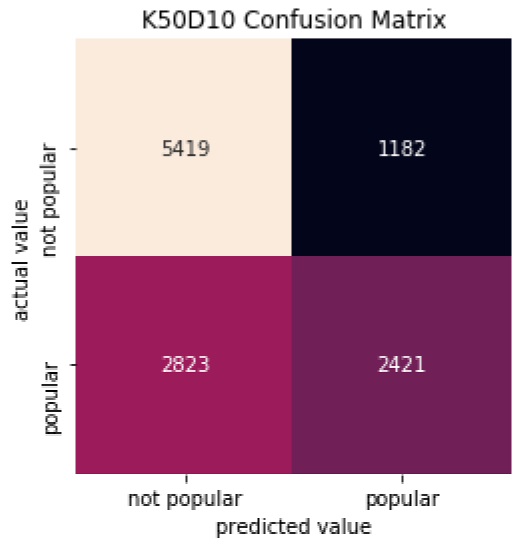


Recall=0.35

Precision=0.7

f1-score=0.62

XGBClassifier:

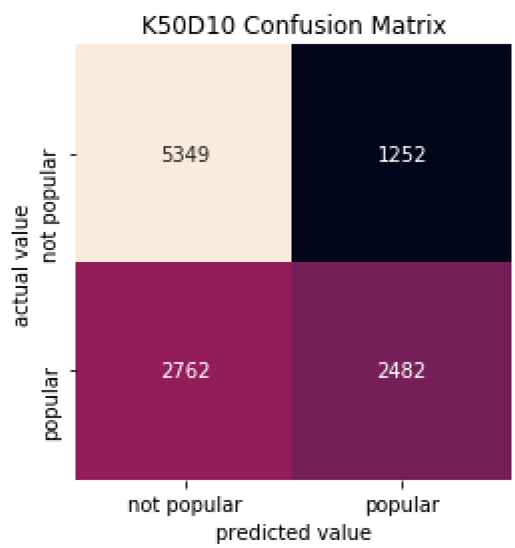


Recall=0.46

Precision=0.67

f1-score=0.65

LightGBM:



Recall=0.47

Precision=0.66

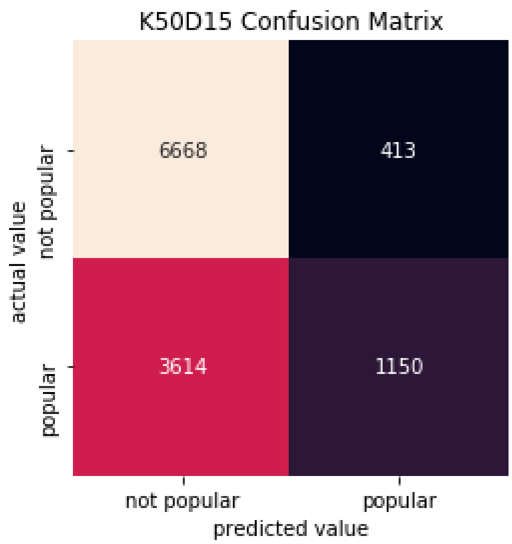
f1-score=0.65

Comment:

Both XGB and LGBM performed best for this target vector

12)K=50, delta=0.15

RandomForestClassifier:

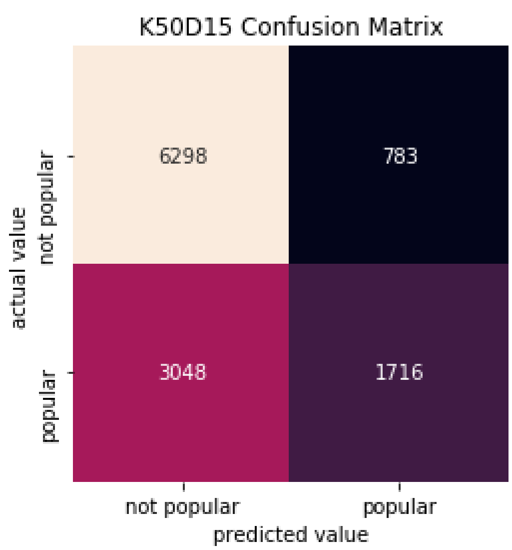


Recall=0.24

Precision=0.74

f1-score=0.6

XGBClassifier:

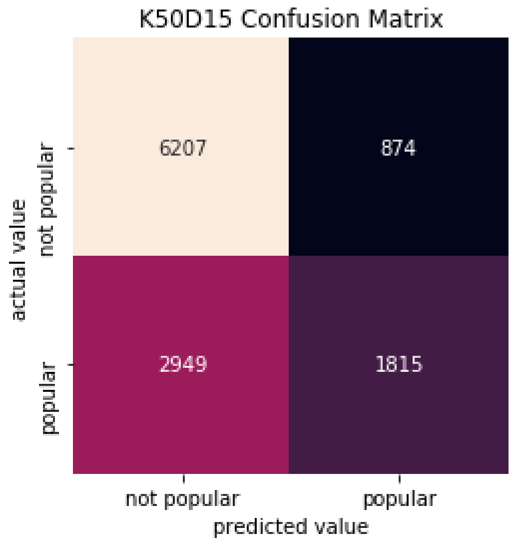


Recall=0.36

Precision=0.69

f1-score=0.65

<u>LightGBM:</u>



Recall=0.38

Precision=0.67

f1-score=0.65

Comment:

LGBM performed better than the other two models

How does the performance of the models change based on the choice of hyperparameters?

Approach:

Tuning the hyperparameters of our models either causes the models to be more or less accurate. This is the point of performing a grid search. We feed the grid search method the grid of hyperparameters that are to be tried on the training data, and the combination of parameters that achieve the best score (in our case, accuracy) is returned. In our RandomForest Classifier, we tuned the n_estimators and max_depth hyperparameters. In our XGBoost Classifier, we tuned the learning_rate, n_estimators, and max_depth hyperparameters. In our LightGBM Classifier, we tuned the n_estimators, min_data_in_leaf, and learning_rate hyperparameters.

Results:

Our approach to training the models was the same as what we suspect the paper's[1] authors did: run grid searches for each of the 3 feature matrices and their 4 respective target vectors, record the results of each grid search run, and then use the mode of each hyperparameter for instantiating the final model that we will score on. In doing this, it is certain that in some cases the model scores will not be as good as they possibly can be. For example if an XGBoost model returned optimal hyperparameters of [learning_rate = 0.1, n_estimators = 750, max_depth = 5] but our final hyperparameters (the mode of each hyperparameter) were [learning_rate = 0.01, n_estimators = 1000, max_depth = 5], we would not expect that specific model to be at its best performance. The grids used for hyperparameters were:

- RandomForest Classifier:
    - n_estimators: [500, 750, 1000]
    - max_depth: [5, 10 ,12]
- XGBoost Classifier:
    - learning_rate: [0.01, 0.1]
    - n_estimators: [750, 1000]
    - max_depth: [5, 10]
- LightGBM Classifier:
    - n_estimators: [750, 1000, 1250]
    - min_data_in_leaf: [10, 20, 30]
    - learning_rate: [0.001, 0.01 , 0.1]

Below is the summary of grid searches run on the original data:

			Original Data			
	RandomForest Grid Search				XGBoost Grid Search	
	K = 10				K = 10	
	max_depth	n_estimators		learning_rate	n_estimators	max_depth
D0 Result	12	500	D0 Result	0.01	1000	5
D5 Result	12	750	D5 Result	0.01	750	10
D10 Result	12	750	D10 Result	0.01	1000	5
15 Result	12	500	15 Result	0.01	1000	5
	RandomForest Grid Search				XGBoost Grid Search	
	K = 30				K = 30	
	max_depth	n_estimators		learning_rate	n_estimators	max_depth
D0 Result	12	750	D0 Result	0.1	750	5
D5 Result	12	750	D5 Result	0.01	1000	5
D10 Result	12	500	D10 Result	0.1	750	5
15 Result	12	750	15 Result	0.01	1000	5
	RandomForest Grid Search				XGBoost Grid Search	
	K = 50				K = 50	
	max_depth	n_estimators		learning_rate	n_estimators	max_depth
D0 Result	12	1000	D0 Result	0.1	750	5
D5 Result	12	1000	D5 Result	0.1	750	5
D10 Result	12	750	D10 Result	0.01	1000	5
15 Result	12	1000	15 Result	0.01	1000	10
	final max_depth	final n_estimators		final learning_rate	final n_estimators	final max_depth
	12	750		0.01	1000	5

Below is the summary of grid searches run on the extended data with RandomForest and XGBoost:

Original + New Data (Extended Dataset)

	RandomForest Grid Search K = 10	
	max_depth	n_estimators
D0 Result	12	500
D5 Result	12	750
D10 Result	12	1000
15 Result	12	1000

	XGBoost Grid Search K = 10		
	learning_rate	n_estimators	max_depth
D0 Result	0.01	1000	10
D5 Result	0.01	750	5
D10 Result	0.01	750	5
15 Result	0.01	1000	5

	RandomForest Grid Search K = 30	
	max_depth	n_estimators
D0 Result	12	750
D5 Result	12	1000
D10 Result	12	1000
15 Result	12	500

	XGBoost Grid Search K = 30		
	learning_rate	n_estimators	max_depth
D0 Result	0.01	750	5
D5 Result	0.01	1000	10
D10 Result	0.01	1000	5
15 Result	0.01	1000	10

	RandomForest Grid Search K = 50	
	max_depth	n_estimators
D0 Result	12	1000
D5 Result	12	750
D10 Result	12	500
15 Result	12	500

	XGBoost Grid Search K = 50		
	learning_rate	n_estimators	max_depth
D0 Result	0.01	1000	5
D5 Result	0.01	750	10
D10 Result	0.1	750	5
15 Result	0.01	1000	5

final max_depth	final n_estimators
12	1000

final learning_rate	final n_estimators	final max_depth
0.01	1000	5

Below is the summary of grid searches run on the extended data with LightGBM:

Original + New Data (Extended Dataset)			
LightGBM Grid Search			
K = 10			
	learning_rate	min_data_in_leaf	n_estimators
D0 Result	0.01	20	1000
D5 Result	0.01	30	1250
D10 Result	0.01	30	750
15 Result	0.01	10	1250
LightGBM Grid Search			
K = 30			
	learning_rate	min_data_in_leaf	n_estimators
D0 Result	0.01	30	750
D5 Result	0.01	30	1250
D10 Result	0.01	10	1250
15 Result	0.01	10	750
LightGBM Grid Search			
K = 50			
	learning_rate	min_data_in_leaf	n_estimators
D0 Result	0.01	10	750
D5 Result	0.01	10	1000
D10 Result	0.01	10	1250
15 Result	0.01	20	1250
	final learning_rate	final min_data_in_leaf	final n_estimators
	0.01	10	1250

Since all final models were scored on the final hyperparameters, it stands to reason that some of these models performed worse than they should have, if their optimal parameters were used.

In the remainder of this section, I will show how changing the parameters of certain models can affect their fit and performance.

Below is the RandomForest Classifier trained on the extended data, for the K50D0 matrix and target vector, using different hyperparameters:

	Training Data	RandomForest Comparison Results for K50D0 Data				
	Accuracy	Accuracy	Balanced Accuracy	F1	Precision	Recall
Model 1	72.19	65.23	64.90	65.09	65.83	70.81
Model 2	61.06	61.03	60.49	60.62	61.37	70.45
Model 1 Hyperparameters			max_depth = 12	n_estimators = 1000		
Model 2 Hyperparameters			max_depth = 5	n_estimators = 500		

It is evident here that changing the hyperparameters has a negative effect on the model. It becomes severely underfitted, and all scoring metrics drop significantly except for recall.

Below is the XGBoost Classifier trained on the extended data, for the K50D0 matrix and target vector, using different hyperparameters:

	Training Data	XGBoost Comparison Results for K50D0 Data				
	Accuracy	Accuracy	Balanced Accuracy	F1	Precision	Recall
Model 1	66.67	66.08	65.86	66.02	67.13	69.88
Model 2	95.97	64.77	64.58	64.74	66.02	68.44
Model 1 Hyperparameters			learning_rate = 0.01	n_estimators = 1000	max_depth = 5	
Model 2 Hyperparameters			learning_rate = 0.1	n_estimators = 750	max_depth = 10	

It is evident here that changing the hyperparameters has a negative effect on the model. It becomes severely overfitted, and all scoring metrics are decreased.

Below is the LightGBM Classifier trained on the extended data, for the K50D0 matrix and target vector, using different hyperparameters:

	Training Data	LightGBM Test Data Results				
	Accuracy	Accuracy	Balanced Accuracy	F1	Precision	Recall
Model 1	67.33	66.15	65.98	66.12	67.48	69.13
Model 2	74.79	66.04	65.85	66.00	67.26	69.34
Model 1 Hyperparameters		learning_rate = 0.01	min_data_in_leaf = 10	n_estimators = 1250		
Model 2 Hyperparameters		learning_rate = 0.1	min_data_in_leaf = 30	n_estimators = 750		

It is evident here that changing the hyperparameters actually led to a better fit for the model, even though the scoring metrics dropped slightly, all except for recall.

Evidenced by all the points above, we are able to change the fit and scores of the models by tinkering with their hyperparameters. If we had less limitations (time, cluster lifespan), we would ultimately have preferred to train all these models to have proper fitting such that they would truly be optimal. Working with the restrictions we had, we are very pleased with our results.

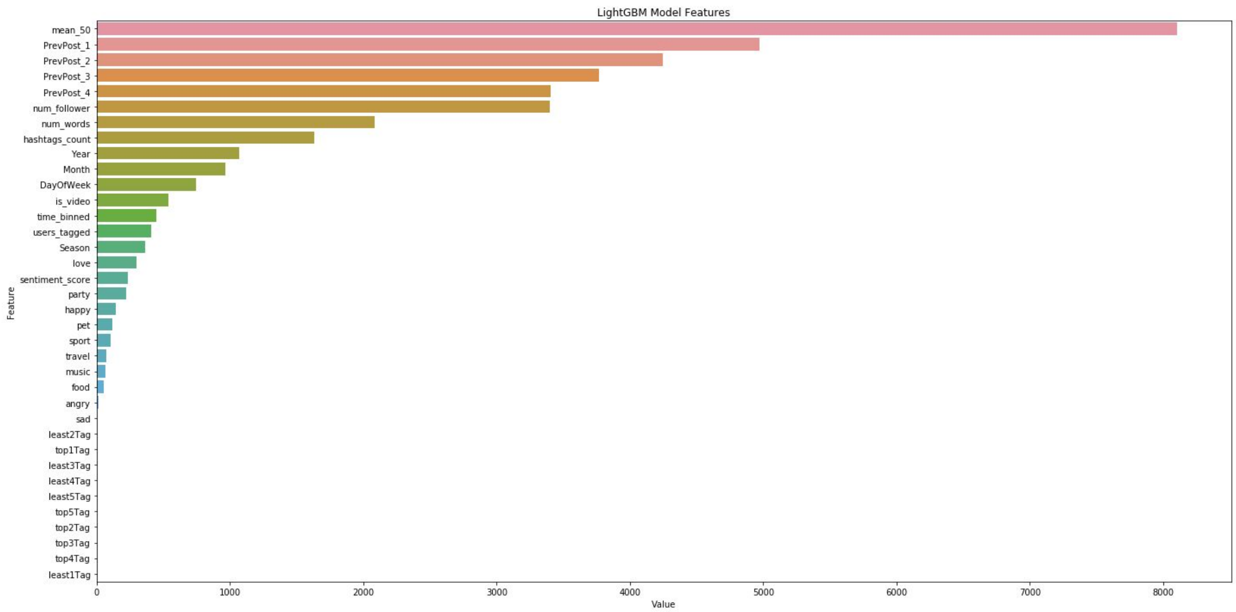
How are the misclassifications of the best performing model distributed?

Approach:

Our best performing model on the extended data was the LightGBM classifier, on the K50D0 matrix and vector. To figure this problem out, we'll have to also look at feature importance.

Results:

Below is a plot of feature importance for the LightGBM model on the extended data, for the K50D0 matrix and vector:



From the above figure, we can see that the features holding the greatest influence over the model are:

- Mean_50
- PrevPost_1
- PrevPost_2
- PrevPost_3
- PrevPost_4
- num_follower

The first five features make a lot of sense. If the average likes of the last 50 posts is low, it could be easier to make a popular post because the competition between posts is low. However, it could also be harder, as it may be difficult to keep improving each post. If the average likes of the last 50 posts is high, it could be harder to make a popular post because it is difficult to keep improving each post. However, it could also be easier, as it could be a reflection that the user is becoming more popular and receiving more likes in general. The above rationale also applies to PrevPost_1 to PrevPost_4. When it comes to num_follower, a larger following could make it easier to receive more likes and actually continue expanding their viewership, leading to a positive domino effect. The opposite applies to a smaller following, where it would likely be more difficult to consistently appease a large portion of their audience, and it would be more difficult to expand their viewership, leading to a negative bottleneck effect.

The following data has been manually labelled. It consists of 6 above features, along with the true popularity vector, and the predicted popularity vector for the K50D0 data. Each misclassification is labelled as follows:

- PT = Popularity Trend
    - This means that all 4 previous posts followed an upward or downward trend that the model likely picked up on and made its decision in conjunction with the mean_50 column
- CP = Cyclical Popularity
    - This means that all 4 previous posts didn’t necessarily follow a trend, but were all up and down compared to each other and there was no consistency
- HM50 = High mean_50
    - This means that the mean_50 was high enough such that consistently achieving that high amount of likes was unrealistic
- SF = Small Following
    - This means that the user held a small following (~300 or less) and was not able to engage that small amount of followers
- PE = Popularity Extremes
    - This means that one of the PrevPost columns had an extremely high or extremely low value compared to the other three
- LE = Low Engagement
    - This means that the user had a large following (~1000 or higher) but consistently got very low likes on the four previous posts and had a low mean_50

The labelled misclassified results are as follows:

1	mean_50	PrevPost_1	PrevPost_2	PrevPost_3	PrevPost_4	num_follower	Actual Outcome	Predicted Outcome	Likely Misclassification Reason
2	85.06	141	88	60	48	993	0	1	PT
3	25.26	40	24	43	25	546	0	1	CP
4	83.78	60	60	110	116	3939	0	1	PT
5	91.66	89	157	112	77	1222	0	1	CP
6	220.28	196	179	255	131	2063	0	1	HM50
7	13.2	10	19	18	12	308	0	1	SF
8	122.38	56	230	213	93	1364	1	0	CP
9	48.52	20	26	30	29	1779	1	0	PT
10	115.1	117	106	132	211	1493	1	0	PT
11	63.62	58	39	73	51	993	0	1	CP
12	31.52	35	21	35	16	2340	1	0	CP
13	15.58	8	18	10	8	540	1	0	CP
14	69.06	64	30	106	71	1066	1	0	CP
15	34.08	43	27	59	38	879	1	0	CP
16	1201.84	992	1294	1285	1464	19400	0	1	PT
17	104.98	131	148	152	172	1220	0	1	PT
18	32.32	29	36	28	28	612	1	0	CP
19	16.86	16	21	14	17	928	0	1	CP
20	64.52	71	11	61	76	2399	1	0	PE
21	210.36	239	175	279	150	2873	0	1	CP
22	2476.76	1508	1503	3071	2586	16400	0	1	CP
23	53.72	48	62	54	36	563	1	0	CP
24	71.92	46	59	73	54	1237	1	0	CP
25	28.04	27	28	29	30	467	0	1	PT
26	102.52	105	124	91	97	1077	0	1	CP
27	97.9	148	67	109	114	1727	1	0	PE
28	132.62	136	108	168	123	3300	1	0	CP
29	114.04	164	188	110	149	1064	0	1	CP
30	95.02	122	122	132	88	1211	0	1	CP
31	19.68	15	28	13	15	278	1	0	SF
32	42.78	25	32	67	28	2767	1	0	PE
33	15.26	21	19	14	10	323	0	1	PT
34	32.38	34	32	41	20	395	0	1	CP
35	277.08	325	236	349	251	2987	1	0	CP
36	70.04	72	49	73	105	1567	0	1	CP
37	25.46	39	22	25	28	1161	0	1	PT
38	32.72	16	53	12	26	879	1	0	CP
39	12.48	8	14	15	8	930	1	0	CP
40	457.22	580	1062	877	268	11300	0	1	CP
41	22.26	48	48	55	11	2316	0	1	PE
42	14.08	16	17	11	18	511	0	1	LE
43	235.26	185	326	207	278	2769	0	1	CP
44	3.54	2	4	4	2	1510	1	0	LE
45	44.26	78	41	31	39	1467	0	1	PE
46	33.7	32	43	32	23	368	1	0	CP
47	670.1	1391	367	473	736	5226	0	1	CP
48	76.58	73	65	49	64	1124	1	0	CP
49	19.8	17	22	11	30	2416	1	0	LE
50	514.26	229	403	509	529	16500	1	0	PT
51	24.6	26	28	24	31	306	0	1	SF
52	42.62	49	38	58	49	1216	0	1	LE
53	22.1	9	30	15	19	781	0	1	LE
54	8.32	4	4	8	12	276	1	0	SF
55	216.14	248	335	213	166	2181	0	1	CP
56	29.88	32	31	53	34	854	0	1	PE
57	18.54	27	24	47	18	601	0	1	PE

58	22.28	29	25	20	18	270	0	1	PT
59	25.72	32	13	10	44	321	0	1	CP
60	82.88	65	51	79	130	622	1	0	PE
61	96.9	105	93	90	105	4175	1	0	LE
62	90.58	112	45	88	72	746	1	0	CP
63	249.04	225	135	232	332	2952	0	1	CP
64	25.18	39	40	12	31	599	0	1	PE
65	110.7	115	134	124	141	2064	0	1	CP
66	168.54	194	204	100	497	1064	0	1	PE
67	153.22	177	76	96	16	12800	1	0	PE
68	296.14	438	338	293	292	5196	0	1	PT
69	145.94	104	130	231	218	12800	1	0	CP
70	66.4	57	62	62	80	518	1	0	PT
71	449.66	278	210	129	177	6601	1	0	CP
72	1118.68	1676	1308	1403	1107	24700	1	0	PT
73	204.08	276	157	290	213	1415	0	1	PE
74	47.06	38	71	47	47	524	0	1	PE
75	59.1	61	76	37	69	1606	1	0	CP
76	26.86	23	24	36	16	475	1	0	CP
77	42.24	39	31	37	52	1375	0	1	PE
78	147.54	97	52	136	155	4178	1	0	CP
79	107.28	196	46	93	146	1464	0	1	CP
80	186.06	180	235	249	211	2296	0	1	CP
81	141.18	127	84	117	109	2091	1	0	CP
82	202.28	198	286	225	193	2134	0	1	CP
83	128.86	130	154	96	147	1548	0	1	PE
84	82.42	85	45	50	49	4227	1	0	PE
85	60.52	77	66	79	69	1202	0	1	CP
86	43.68	51	51	18	37	970	0	1	PE
87	140.96	148	74	204	151	1378	0	1	PE
88	261.22	275	387	375	492	5290	0	1	PT
89	25.16	28	37	29	32	256	0	1	SF
90	82.06	131	93	128	101	692	0	1	CP
91	174.22	242	274	167	242	2873	0	1	PE
92	137.66	98	203	259	145	1456	0	1	PE
93	42.1	43	81	20	54	1238	0	1	PE
94	472.26	323	617	553	393	21500	0	1	CP
95	49.12	60	62	78	51	1201	0	1	CP
96	134.16	173	139	143	117	1393	0	1	PT
97	27.26	31	27	31	38	598	0	1	LE
98	87.9	66	76	88	77	975	1	0	CP
99	64.04	81	86	38	106	765	0	1	PE
100	43.04	33	57	45	47	1250	0	1	CP
101	123.3	112	182	104	142	1661	0	1	CP

Discussions

Patrick Pickard:

A possible scenario where this model could be used is for gathering information on users/clientele that may be interacting with your product from an Instagram account. Example; if your product was a website, or any form of software that allowed users to interact with your product with their Instagram account, you would be able to use a model such as this to get some information on your market/users. Using the model to determine what your users posted leading to “more popular” posts may allow you to adjust your business model to either capture additional market segments, better understand your market (possibly popular posts have common traits in them that your product could utilize to increase success), or determine where you could spend your money to drive up interactions/sales/etc.

Models such as this can offer valuable insight into sects or groups of similar people, or give better broad “market” sentiment on what is popular at the current moment. This can give products a potential competitive advantage who might use this information effectively.

Ziad Chemali:

Another scenario, a random instagram user who is looking to grow his profile and wants to have more engagement with his/her followers . He/she would use this model and give it all the inputs required (caption, publish time, and all other features mentioned in the pre-processing/ feature engineering step) and the model will predict the posts popularity.What they can do is keep changing the caption and publishing time and see what kind of result they would obtain from the model.

Joshua Posyluzny:

With social media becoming so prevalent these days, it seems that the race to become an “influencer” is as popular as ever. An influencer is a person who uses their social media influence in order to affect the purchasing decisions of their audience. With our model, users with a smaller audience who wish to become influencers with larger audiences could runtest posts that will be analyzed and determined with relatively decent accuracy whether it will be popular or not. By consistently doing this, smaller users could substantially grow their audience, achieving influencer status.

Conclusions

Overall, our project was able to replicate the referenced papers results, and improve on the evaluation metrics used to score the models created with our extended dataset. We were also able to find another potentially well functioning model with the LGBM model used, as it was seen to perform comparably with the best performing XGBoost model also used in the referenced paper.

The potential for expansion on this project exists in the sense that additional features beyond the raw meta-data used could be considered such as derived features such as binned post time of day, frequency of posts, interaction/activity of profile, comment numbers, etc. Additionally, more complicated analysis could be used like image processing to determine if patterns in the images posted had any potential correlation with the classification outcome.

In summary, we replicated the paper's models (XGBoost/RandomForest) by using their original data set, then scraped additional 1000 new data points from a total of 10 public ordinary users. After extracting the new data, we followed the paper's steps in preprocessing/feature engineering; later, we retrained the models (XGBoost/RandomForest) and tested another potential model (LGBM).
The results showed that the XGBoost and LGBM models are our best performing models when classifying the popularity of a post using K=50 and delta=0 where K is the rolling average window of previous posts and delta is the tolerance.

References

1. Salvatore Carta, Alessandro Sebastian Podda, Diego Reforgiato Recupero, Roberto Saia, and Giovanni Usai. 2020. Popularity Prediction of Instagram Posts *Information* **2020**, *11*(9), 453; **https://doi.org/10.3390/info11090453**
2. Instagram-scraper (open source API). https://github.com/arc298/instagram-scraper
3. PySpark LightGBM https://github.com/Azure/mmlspark/blob/master/docs/lightgbm.md