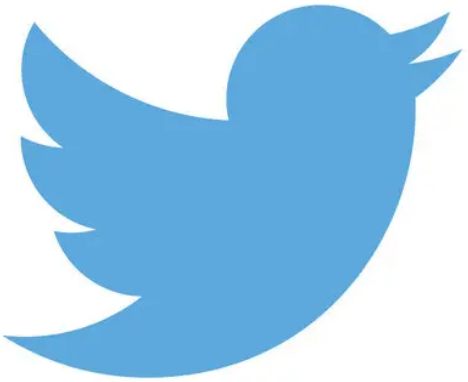
Maximizing Tweet Outreach and Predicting Tweet Performance



Angad Pfuetzner

Table of Contents

Maximizing Tweet Outreach and Predicting Tweet Performance

Problem Statement

One of the biggest challenges for musical artists is effectively using social media to reach as many people as possible. The more people that see an artist on social media, the more people check out their music, the more people become fans and support the artist.

The social media platform I will be looking at is Twitter. It is very common for musical artists to tweet promotional content for upcoming projects, links to songs, or even just random thoughts they have. All of these types of tweets can help an artist reach more people if enough people see it and engage with it through likes or retweets. Typically, the higher the number of people that see a tweet, the higher the number of likes and retweets the tweet gets.

This project seeks to determine if there is an optimal time of day for artists to tweet in order to reach the most people, and subsequently what time that would be. This will help ensure that artists can tweet their promotional content or whatever else they choose at the optimal time of day to maximize their reach.

The main idea is to determine if there is any correlation between time of tweet and the amount of engagement. Subsequently, I will try to build a predictive model to determine how a certain tweet will perform (based on favorites) using data on the tweet time and day as well as the words contained in the tweet itself.

Data Access

To access Twitter data, it is necessary to make a Twitter developer account and start a new project. Once a project is created you receive: a consumer key, a secret consumer key, an access token and a secret access token. These keys can then be used with the user-friendly tweepy python library (https://www.tweepy.org/) to pull tweets and their data.

To determine which twitter accounts to look at, I used Spotify, https://www.music-map.com/ and my own personal knowledge of the electronic music scene. The requirements were that the fan bases were similar, the account had at least 5000 tweets and the account was verified. I picked 10 artistics in the electronic genre because of my personal knowledge of the genre and the specific artists. The artists and their twitter screen names can be seen in the table below:

Artist	Twitter Screen Name
Alison Wonderland	awonderland
Galantis	wearegalantis
San Holo	sanholobeats
The Knocks	theknocks
Peking Duk	pekingduk
Porter Robinson	porterrobinson
What So Not	WhatSoNot
Louis The Child	LouisTheChild
NGHTMRE	NGHTMRE
Jai Wolf	JaiWolfx

The next step was to determine which attributes of the tweet data would be useful to pull. After going through the list of possible attributes to call, the final list of attributes was as follows:

- screen_name : artist twitter name
- reply_to: screen_name of twitter user the tweet is in response to
- is_quote_status: boolean indicating if the tweet quotes another tweet
- text: tweet text
- created_at: date and time the tweet was tweeted
- favorites: number of favorites the tweet has at the time the data is pulled
- retweets: number of retweets the tweet has at the time the data is pulled
- follower_count: number of followers the account has at time the data is pulled
- entities: contains info about hashtags, images, links etc.
- is_truncated: boolean indicating if the full tweet text has not been included
- statuses_count: number of tweets the account has at time of tweeting

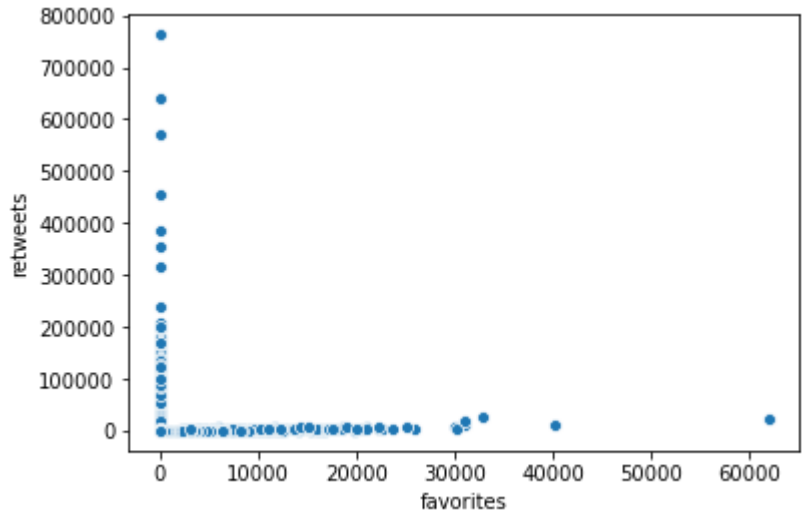
The Twitter API allows approximately 3200 tweets to be pulled from an account's timeline at once. For some unknown reason, the Tweepy call pulled only 31 680 tweets but it was decided that should still be a sufficient amount of data to explore.

Data Wrangling

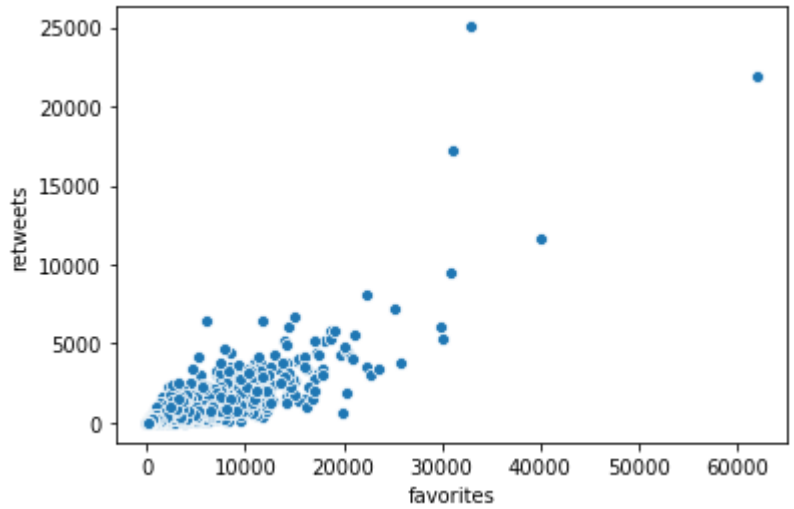
There were several issues with the data that needed to be rectified in order to be able to explore the data.

1. Retweeted Tweets

The first thing I looked at was the relationship between favorites and retweets. I predicted there should be a reasonably strong linear correlation between the two and as therefore quite surprised when the following scatter plot resulted.



The plot showed an alarming number of tweets with hundreds of thousands of retweets and 0 favorites, which I have never encountered on Twitter. After digging deeper, I was able to determine that retweeted tweets, which can be identified by the “RT @” at the beginning of the text, all had 0 favorites but kept the actual number of retweets. Filtering out the retweeted tweets provided a much more reasonable plot.



2. Reply Tweets

I know from personal experience that reply tweets often do not appear on the timeline and therefore perform significantly worse in terms of favorites and retweets. For this reason I removed any reply tweets.

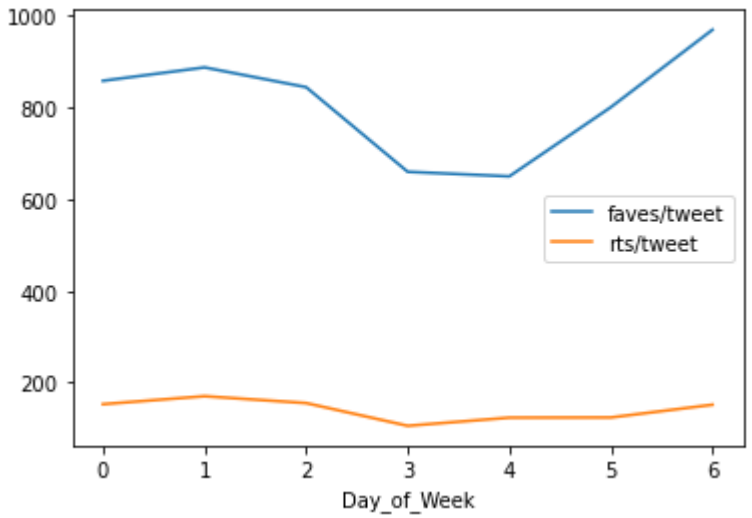
3. Adding Day of Week and Time of Day

Since the created_at column was a series of datetime objects, it was quite simple to extract the time of day (as an hour of day from 0 to 23) and the day of the week from 0 (Monday) to 6 (Sunday). These values were added to the dataframe for each tweet.

Exploratory Data Analysis

The next step was to begin exploring the data. To answer the first question of the optimal time to tweet, I executed the following steps. First, I grouped the data by day of the week, summing the favorites and retweets fields and counting the text field to get a count of how many tweets were tweeted each day. Then I divided both the favorites and retweets columns by the text count to obtain favorites/tweet and retweets/tweet values for each day.

Day_of_Week	favorites	retweets	text	faves/tweet	rts/tweet
0	1415779	252249	1650	858.047879	152.878182
1	1557118	298754	1755	887.246724	170.230199
2	1523742	280253	1805	844.178393	155.264820
3	1195871	191575	1813	659.608935	105.667402
4	1324702	251706	2039	649.682197	123.445807
5	1194485	184755	1492	800.593164	123.830429
6	1294075	202431	1335	969.344569	151.633708

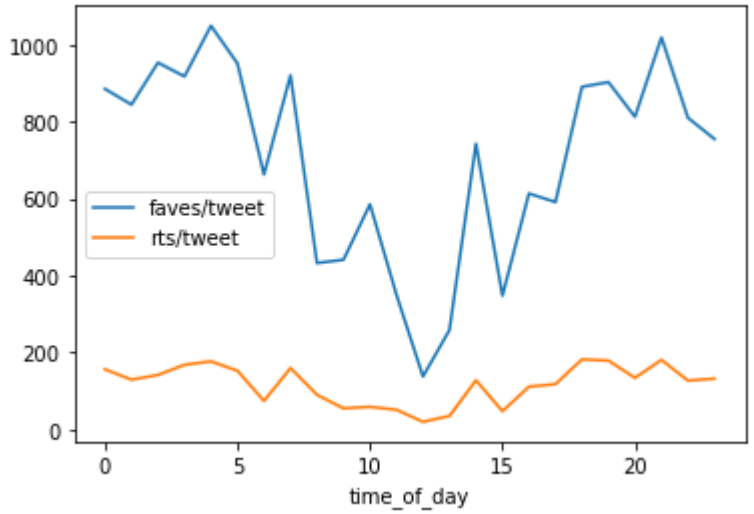


As we can see in the plot above, the retweets/tweet seem to remain similar throughout the week, with the worst day being thursdays. For the favorites/tweet column there is a larger spread. Sundays (6) are the best day while Thursdays (3) and Fridays (4) are the worst by a significant margin. This could be attributed to the fact that Thursdays and Fridays are the

most common weekdays for people to go out and socialize at night, therefore spending less time on social media. People are typically working during the day and out at night so there is much less time to read and interact with tweets. Sundays are often lazy days spent at home browsing social media so it makes sense to me that they have the highest performance tweets on average.

The above steps were then repeated using the time of day to group the data. Grouping by the time of day gave the following results: MAKE SCATTER PLOTS

time_of_day	favorites	retweets	text	faves/tweet	rts/tweet
4	410384	69251	391	1049.575448	177.112532
21	920363	163560	903	1019.228128	181.129568
2	483543	71923	507	953.733728	141.859961
5	281629	45210	296	951.449324	152.736486
7	184121	31975	200	920.605000	159.875000
3	368077	67465	401	917.897756	168.241895
19	951819	188942	1054	903.054080	179.261860
18	946090	193573	1062	890.856874	182.272128
0	663202	117307	749	885.449933	156.618158
1	541396	83051	641	844.611544	129.564743
20	847093	140012	1042	812.949136	134.368522
22	692085	108642	854	810.403981	127.215457
23	598183	104684	792	755.281566	132.176768
14	112915	19457	152	742.861842	128.006579
6	183021	20541	276	663.119565	74.423913
16	369926	67172	603	613.475954	111.396352
17	572425	114231	968	591.348140	118.007231
10	66155	6658	113	585.442478	58.920354
9	55604	6942	126	441.301587	55.095238
8	77041	16073	178	432.814607	90.297753
11	24173	3547	69	350.333333	51.405797



The plot above shows that the evening (18-20) and late night (0-4) times have the highest tweet performance, while tweets perform significantly worse during the day (8-17). To me this is easily explainable by the fact that most people are working during the day and are much more active on social media at night.

Pre-Processing

Having answered the first question of the optimal time of day and day of the week to tweet, I moved on to begin constructing a model of predicting tweet performance. The first step was to perform some common NLP data processing.

1. NLP Data Processing

To prepare the tweet text field for use in a predictive model some common NLP techniques had to be used. The first thing I noticed was that any tweets that included an image or video contained a link in the text beginning with “http.”. To standardize any type of media included I replaced all of these links with the phrase “containsmedia”. Next I used a CountVectorizer on the tweet texts to determine how many unique words are present. Without including stop words there were 10 160 unique words present in the data, which is far too many to include as features for this model. To try and mitigate this I filtered out stopwords and the words that were used less than 100 times. This gave a much more reasonable length list of 95 words which can be seen here:

'containsmedia', 'u', 'music', 'new', 'love', 'today', 'see', 'first', 'thank', 'get', 'like', 'live', 'song', 'good', 'one', 'much', 'amp', 'time', 'show', 'day', 'tomorrow', 'set', 'want', 'virtual', 'people', 'na', 'miss', 'really', 'coming', 'album', 'going', 'tonight', 'happy', 'guys', 'ever', 'year', 'tour', 'back', 'years', 'play', 'know', 'ur', 'life', 'last', 'best', 'night', 'video', 'got', 'right', 'make', 'everyone', 'worlds', 'amazing', 'remix', 'next', 'shows', 'week', 'playing', 'let', 'things', 'seafoxnation', 'listen', 'friends', 'gon', 'go', 'made', 'soon', 'wait', 'world', 'full', 'us', 'come', 'friday', 'san', 'weekend', 'special', 'vegas', 'summer', 'tune', 'big', 'tickets', 'galantis', 'spotify', 'sale', 'la', 'sold', 'ca', 'excited', 'x', 'thanks', 'pm', 'w', 'tix', 'check', 'party'

The counts of each of the above words in the text field were added as features and I moved on to general pre-processing for the other features.

2. General Pre-Processing

The first pre-processing step was to deal with the categorical variables: screen_name, Day_of_Week and time_of_day. I used the pandas get_dummies function to make dummy variables and then dropped the first value to avoid collinearity. I then removed the irrelevant features: 'is_quote_status', 'text', 'created_at', 'follower_count', 'is_truncated' and 'statuses_count'. The features that remained alongside the words above were:

'screen_name_LouisTheChild', 'screen_name_NGHTMRE', 'screen_name_WhatSoNot', 'screen_name_awonderland', 'screen_name_pekingduk', 'screen_name_porterrobinson', 'screen_name_sanholobeats', 'screen_name_theknocks', 'screen_name_wearegalantis', 'Day_of_Week_1', 'Day_of_Week_2', 'Day_of_Week_3', 'Day_of_Week_4', 'Day_of_Week_5', 'Day_of_Week_6', 'time_of_day_1', 'time_of_day_2', 'time_of_day_3', 'time_of_day_4', 'time_of_day_5', 'time_of_day_6', 'time_of_day_7', 'time_of_day_8', 'time_of_day_9', 'time_of_day_10', 'time_of_day_11', 'time_of_day_12', 'time_of_day_13', 'time_of_day_14', 'time_of_day_15', 'time_of_day_16', 'time_of_day_17', 'time_of_day_18', 'time_of_day_19', 'time_of_day_20', 'time_of_day_21', 'time_of_day_22', 'time_of_day_23'

Including the word count features there were still 133 features, which is too many for the number of data points I have. To reduce the number of features I manually looked through the list of words and selected the ones that I thought would be most relevant. Certain words like ‘x’, ‘got’, ‘u’, etc. were easy to remove because of their irrelevance to the tweet’s main message while others like ‘seafoxnation’ and ‘galantis’ were removed because they were only really used in one artist’s tweets (galantis). The final list of features (70) for the model can be seen here:

'containsmedia', 'music', 'new', 'love', 'today', 'first', 'like', 'live', 'song', 'good', 'time', 'show', 'day', 'tomorrow', 'set', 'virtual', 'people', 'miss', 'album', 'going', 'tonight', 'happy', 'guys', 'ever', 'year', 'tour', 'play', 'life', 'last', 'best', 'night', 'video', 'screen_name_LouisTheChild', ‘screen_name_NGHTMRE', 'screen_name_WhatSoNot', 'screen_name_awonderland', 'screen_name_pekingduk', 'screen_name_porterrobinson', 'screen_name_sanholobeats', ‘screen_name_theknocks', 'screen_name_wearegalantis', 'Day_of_Week_1', ‘Day_of_Week_2', 'Day_of_Week_3', 'Day_of_Week_4', 'Day_of_Week_5', 'Day_of_Week_6', 'time_of_day_1', 'time_of_day_2', 'time_of_day_3', 'time_of_day_4', 'time_of_day_5', 'time_of_day_6', 'time_of_day_7', 'time_of_day_8', 'time_of_day_9', 'time_of_day_10', 'time_of_day_11', 'time_of_day_12', 'time_of_day_13', 'time_of_day_14', 'time_of_day_15', 'time_of_day_16', 'time_of_day_17', 'time_of_day_18', 'time_of_day_19', 'time_of_day_20', 'time_of_day_21', 'time_of_day_22', 'time_of_day_23'

The favorites and retweets columns were used as the y variables and the features above were used as X. The data was then split into training and test sets with a 75%/25% split, to then be used for model building.

Modeling

1. Model Selection

The first step in modeling was to determine whether I should use the favorites or the retweets values as the y variable for my model. After comparing linear regression models using favorites and using retweets, the model using favorites showed a significantly better R2 value. Based on these findings I decided to proceed using the favorites value as the variable to predict.

R2 using favorites: 0.2764
R2 using retweets: 0.1559

To determine what type of model to build, 3 different algorithms were compared. A linear regression model, a K nearest neighbors model and a random forest model were compared on the basis of training time (seconds), prediction time (seconds), train mean squared error and test mean squared error. For simple comparison purposes, the default out-of-the-box sklearn algorithms were used. The results are seen in the following table:

	model	train_mse	test_mse	train_time	pred_time
0	Linear Regression	3.078481e+06	3.491299e+06	0.045973	0.007995
1	KNN	4.304366e+06	4.993207e+06	0.559841	4.347990
2	Random Forest	1.609311e+06	5.880136e+06	65.291681	53.656602

From the above table I decided the linear regression model was the most promising, although none of the models provided great results in terms of the mean squared error. The random forest model performed well on the training set but was clearly overfit as the test set error was very large. The training and prediction time were also quite long and therefore the random forest model was eliminated from contention. The KNN model performed better but was still worse than the linear regression model in all aspects.

2. Hyperparameter Tuning

To better tune the model I decided to use an elastic net model and tune the alpha, l1_ratio and normalize parameters. The parameter values I used were:

alpha: (0.01, 0.1, 1)
l1_ratio: (0, 0.5, 1)
normalize: (True, False)

Since there were not too many parameter combinations to test and training times were quite fast for the linear regression model I decided to use GridSearchCV to determine the optimal parameters. Fitting the GridSearchCV and calling the best_estimator_ attribute returned ElasticNet(alpha=0.1, l1_ratio=1, normalize=True).

3. Final Model

Finally, I built and trained an ElasticNet model using the parameters above. I then looked at the top 10 coefficients and their values which can be seen in the following table.

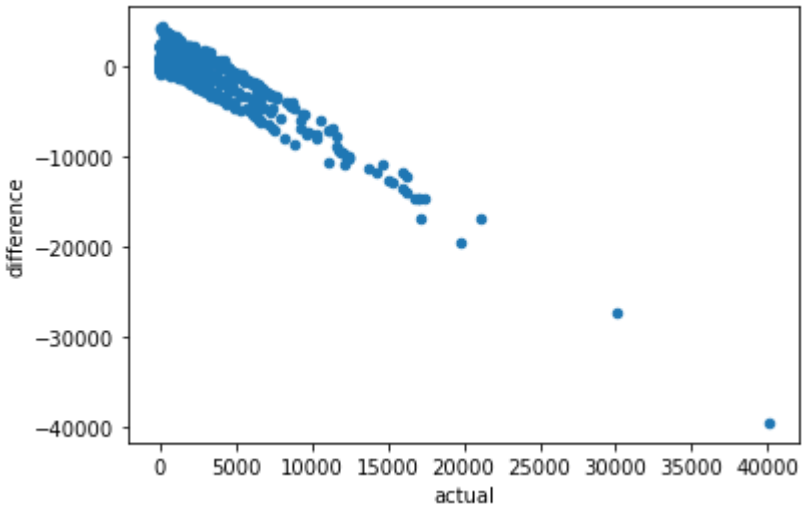
	coeff
screen_name_awonderland	3847.854521
screen_name_porterrobinson	1988.782060
music	635.842530
screen_name_sanholobeats	550.346278
going	519.619575
album	503.997065
virtual	483.015332
miss	420.031050
song	360.222659
first	275.698762

The top 2 coefficients correspond to tweets from awonderland and porterrobinson which makes sense as they have the largest followings and therefore typically receive the most favorites on their tweets. Words like music, album and song also made the top 10 which can easily be explained by fan excitement for new works from the artist.

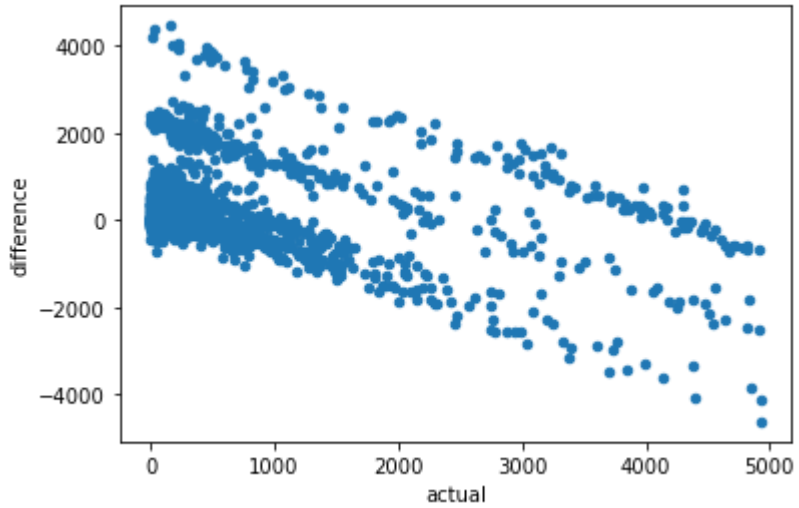
Results

1. Error Analysis

The final tuned linear regression model returned a mean squared error of 3 490 805, which is not much of an improvement from the out-of-the-box linear regression model. This means that on average tweets are 1868 favorites away from the real value. When plotting the distribution of the number of favorites a tweet received vs. the difference of favorites predicted and the actual favorites received we can see the difference increasing as the number of favorites increases. The model largely under-predicts high performing tweets.



When looking at tweets with favorites up to 5000, we can see that the model over-predicts favorites at low values and gradually begins under-predicting as favorites increase.



The model clearly does not make very accurate predictions. This is not a complete surprise as there are certainly many more factors involved in a tweet's performance than the time of the tweet, the artist and the count of commonly used words in the tweet.

2. Separating Models by Artist

I decided to do a little bit more investigation to see if the fact that there are 10 different artists' tweets played a significant role in the model's poor performance. I repeated the model building process after separating the data by artist and built 10 linear regression models. The results of the model errors can be seen in the table below.

	artist	train_size	train_mse	test_mse
0	awonderland	694	6.768748e+06	8.230582e+06
1	wearegalantis	1594	2.295202e+06	3.260180e+06
2	sanholobeats	662	8.744734e+05	1.487674e+06
3	theknocks	1484	7.858945e+03	3.970866e+03
4	pekingduk	817	6.285289e+03	3.717030e+03
5	porterrobinson	1219	1.753097e+07	1.706559e+07
6	WhatSoNot	1585	9.097937e+04	4.301338e+04
7	LouisTheChild	1054	8.730527e+05	6.794481e+05
8	NGHTMRE	1099	6.438722e+05	4.584188e+05
9	JaiWolfx	1681	3.349253e+05	2.277984e+06

It looks like certain artists are much more predictable by the model than others. The Knocks and Peking Duk have mean squared errors a factor of 4 less than Porter Robinson. Further investigation would have to be done to determine the cause of the disparity. It could be that The Knocks and Peking Duk have had less viral tweets so all of their tweets perform similarly well and thus lead to less error with even a simple mean value prediction. Overall however, the models are still not amazing even when separated by artist.

Further Study

A number of improvements and modifications can be made on the model to try and improve its performance. Some of the things that can be investigated in further research include:

- The effect of time on tweet performance. The tweet data spans a period of 4 years for some of the artists and tweets naturally reach larger audiences and have more engagement as their follower count increases. Adding some sort of time dependent variable could improve the model performance.
- More advanced NLP analysis. The NLP used in this report was very basic. Further study using more advanced NLP methods such as bigrams/trigram counts, sentiment analysis, etc. could lead to improved model performance.