IT461: Practical Machine Learning

Information Technology Department

TikTok Data Analysis

PREPARED BY:

RAMLAH ALMUSALM|438200676

REEM ALGHANNAM|438200291

Table of Contents

Table of Figures

1) Introduction

Introducing the topic and why is it useful to use ML algorithms on it.

Trends and popularity are largely driven by social media in the modern age. In addition, TikTok is top-rated among teenagers. Despite the spread of Tiktok, we did not see anyone work on its data, we are interested in working and exploring the impact of this application. To be able to predict the popularity of a TikTok post the first thing we need is to collect data about TikTok videos. Unfortunately, there is no available dataset for Tiktok. So, we collect it using Rapid API.

What is the problem you want to solve and what are you proposing to solve it with.

TikTok receives a large portion of their income from advertisement, These advertisements show throughout their application as banners in posts, to attract companies to use TikTok as a way to display their ads and carefully choose a video to embed the ads in. Therefore, predicting the popularity of posts published is a challenging problem.

First of all, the number of times a video played and the number of times the video has been shared among social media both play an important role in the distribution patterns of the post. So, we decided to build models to work in this area, to discover and answer the following questions: what is the relationship between playCount, shareCount with other variables? Given the other variables, can playCount and shareCount be predicted?

2) Data Preprocessing

- **Show what preprocessing techniques you will be using**

1- Drop columns

2- Encode 'statistics.shareCount' parameter

3- Data normalization

- **Why did you choose to apply these techniques?**

1- Drop columns

Because these columns do not contribute to our models, for example: description needs a natural language processing, video_id just identifies the video etc. So, we will drop these columns.

2- Encode 'statistics.shareCount' parameter

To apply classification on 'statistics.shareCount' parameter, we have to transform it to categorical parameter, we will encode it into 0 and 1.

3- Data normalization

Data normalization is the process of putting the numeric data values into a common scale [1], we needed to apply data normalization on the numeric parameters due the high difference of ranges between them. As shown in Figure 1 and Figure 2, there is a high difference between the values, for instance, we can see the max of author.videoCount is 11300, where the max of statistics.playCount is 2.86e+8.

	author.following	author.followers	author.heartCount	author.videoCount	author.diggCount
count	1406.000000	1.406000e+03	1.406000e+03	1406.000000	1406.000000
mean	878.125178	2.035612e+07	1.149892e+09	509.640825	5847.596017
std	1689.049503	3.132031e+07	2.383804e+09	740.100451	17985.477550
min	0.000000	3.690000e+02	4.172000e+03	2.000000	0.000000
25%	24.000000	1.841000e+05	2.900000e+06	54.000000	766.000000
50%	135.000000	3.200000e+06	7.780000e+07	216.000000	3790.000000
75%	1061.000000	1.950000e+07	3.514000e+08	633.000000	7173.000000
max	9265.000000	9.370000e+07	7.200000e+09	11300.000000	436200.000000

Figure 1 Data description

	video.duration	statistics.diggCount	statistics.commentCount	statistics.playCount
count	1406.000000	1.406000e+03	1406.000000	1.406000e+03
mean	19.325747	4.054729e+06	48158.226885	2.638274e+07
std	13.031492	5.068527e+06	72298.956894	3.632508e+07
min	6.000000	1.356000e+03	0.000000	4.230000e+04
25%	13.000000	2.118000e+05	2582.000000	2.300000e+06
50%	15.000000	2.300000e+06	16800.000000	1.230000e+07
75%	18.000000	6.475000e+06	56000.000000	3.940000e+07
max	59.000000	2.240000e+07	531600.000000	2.682000e+08

Figure 2 Data description

• Provide examples of preprocessing steps

1- Drop columns

Before building the models, we get familiar with the data we have. The dataset contains 38 attributes which give information about TikTok videos. we specified the columns that will not add a value and help us to accomplish our task, which are: video_id, create_time, author.verified, description, author.avatarLarger, author.signature, author.private, author.secUid, video.height, video.width, video.ratio, video.cover, video.originCover, video.dynamicCover, video.playAddr, music.id, music.title, music.playUrl, music.coverThumb, music.coverMedium, music.coverLarge.

Since these columns are not useful for getting the desired results, we dropped them.

2- Encode 'statistics.shareCount' parameter

Figure 3 represents the data before encoding, we set a threshold to encode the data upon it, the threshold is the mean value of 'statistics.shareCount' parameter, the values greater than the mean value is encoded as 1, 0 otherwise (see Figure 4). If the video encoded as 1, that means it is shared more than the average times videos shared, so we can say it is shared widely. The number of data points that belong to class 1 is 358, and the rest (1048) belong to class 0. See Figure 5 for the encoded data.

statistics.shareCount
104400
3667
103400
1083
17900
325000
3221
1112
4247
2658

Figure 3 'statistics.shareCount' parameter before encoding

```
meanValue= dataset['statistics.shareCount'].mean()
print(meanValue)
```

```
83586.5960170697
```

```
dataset['statistics.shareCount'].values[dataset['statistics.shareCount'].values <= meanValue] = 0
```

```
dataset['statistics.shareCount'].values[dataset['statistics.shareCount'].values > meanValue] = 1
```

Figure 4 Code to encode 'statistics.shareCount' parameter

statistics.shareCount
1
0
1
0
0
1
0
0
0
0

Figure 5 'statistics.shareCount' parameter after encoding

3- Data normalization

First: We specified the columns we will normalize, which are all numeric columns except 'statistics.shareCount' (categorical now).

Second: We split the data into training/ testing data, 80%, 20% respectively, to apply the normalization on each set separately, Figure 6 and Figure 7 shows a sample of the unnormalized data and Figure 8 shows the code used to split the data.

author.following	author.followers	author.heartCount	author.videoCount	author.diggCount
59	173800	873800	19	1392
833	18600	547100	42	5061
4	112000	1200000	73	1821
79	184100	798900	324	4604
16	37800000	828300000	568	1298

Figure 6 Sample of unnormalized data

video.duration	statistics.diggCount	statistics.commentCount	statistics.playCount
8	838400	10300	23500000
59	541600	19400	2000000
15	738200	13100	5500000
15	138000	3162	1700000
14	6700000	24200	44500000

Figure 7 Sample of unnormalized data

```
X_train, X_test, y_train, y_test = train_test_split(X, y,test_size=0.2, random_state=2)
print (X_train.shape,X_test.shape,y_train.shape, y_test.shape)
```

Figure 8 Splitting data code

Third: We normalized the data using the code in Figure 9. A sample of the data after the normalization process shown in Figure 10 and Figure 11.

```
from sklearn import preprocessing
dataset = pd.DataFrame(preprocessing.scale(X_train.drop(['statistics.shareCount'], axis=1)),
                       columns = ['author.following','author.followers','author.heartCount', 'author.videoCount',
                                  'author.diggCount','video.duration', 'statistics.diggCount', 'statistics.commentCount'])
```

Figure 9 Normalization code

author.following	author.followers	author.heartCount	author.videoCount	author.diggCount
-0.485135	-0.644614	-0.482182	-0.663174	-0.247821
-0.026726	-0.649571	-0.482319	-0.632086	-0.043751
-0.517709	-0.646587	-0.482045	-0.590185	-0.223960
-0.473290	-0.644285	-0.482213	-0.250921	-0.069169
-0.510602	0.557149	-0.134955	0.078881	-0.253049

Figure 10 Sample of normalized data

video.duration	statistics.diggCount	statistics.commentCount	statistics.playCount
-0.869415	-0.634795	-0.523821	-0.079388
3.045574	-0.693373	-0.397910	-0.671476
-0.332064	-0.654571	-0.485079	-0.575089
-0.332064	-0.773030	-0.622585	-0.679738
-0.408828	0.522087	-0.331495	0.498931

Figure 11 Sample of normalized data

3) Methodology

- **Explain the ML algorithm used**

Linear regression: We chose to use linear regression for this task, because we want to predict a continuous variable (response) using another continuous variable (predictor). Moreover, linear regression requires a linear relationship between the predictor and the response and that is the same as we have between the parameters (will be shown later), these reasons prompted us to use linear regression.

Logistic regression: Logistic Regression is a method used to predict categorical variable using one or more continues variables. We chose to use logistic regression for this task, because the response is categorical and the predictor is continuous, also, there is a linear relationship between the predictor and the response (as will be shown later), these reasons prompted us to use logistic regression.

The allocated data for training is 80% and the remining 20% for testing. The models will be trained for one time (no iterations) using the previously split data, to find the best line/curve fits that data. For the testing, the models will be tested using test set (unseen data) for one time (no iterations), and the results will be evaluated using some metrices such as for the linear: MSE, MAE and RMSE, R squared and for the logistic: accuracy, precision, and recall.

For both models, the parameters will be chosen depending on the exitance and strength of the correlation between the predictors and response.

- **Implementation (with screenshots)**

The first step in the model building was feature selection, we identified all the variables that have a strong/moderate linear relationship with the response, to do that we used the heatmap shown in Figure 12. To indicate the strength of the correlation, we will use the following ranges:

weak $0 \leq |r| \leq 0.5$

strong $0.8 \leq |r| \leq 1$

moderate otherwise

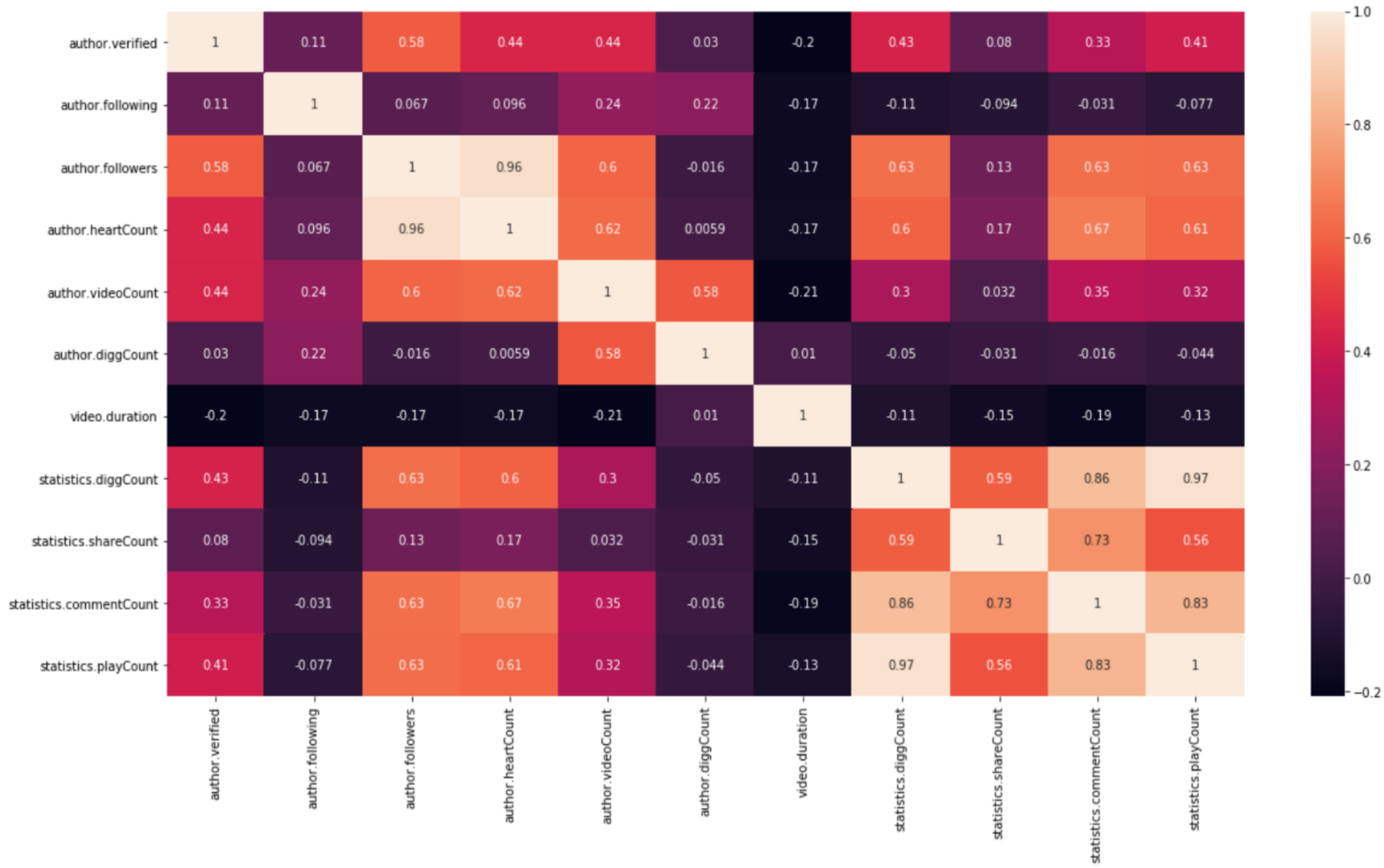


Figure 12 The heatmap

A) Linear Regression

First: To build the model, we will select the predictors, and as shown in the heatmap (Figure 12), we can indicate that there is a strong correlation between the response 'statistics.playCount’ with 'statistics.diggCount', also we can determine the correlation using a scatter plot, as shown in Figure 13.

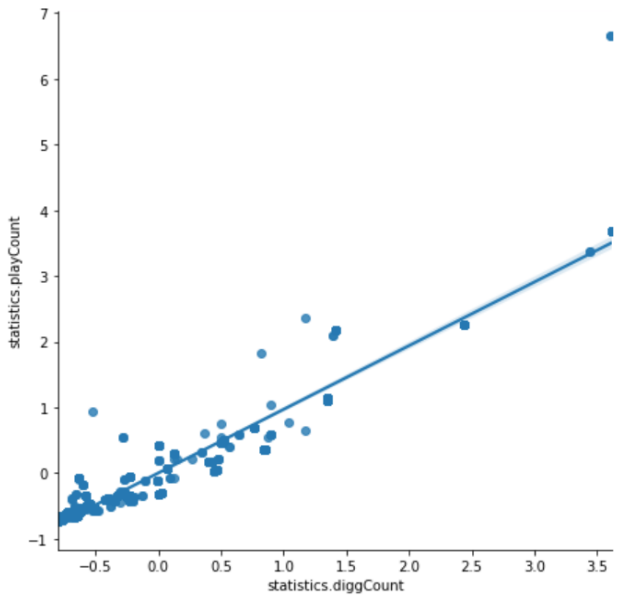


Figure 13 Scatter plot shows to correlation between 'statistics.playCount’ and 'statistics.diggCount'

So, we will use 'statistics.playCount’ and 'statistics.diggCount' to build the model.

Second: Create and fit a model.

As shown in Figure 14, we create an instance of the class LinearRegression, which will represent the linear regression model.

```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
```

Figure 14 Building linear regression model part1

As shown in Figure 15, we call the fit() method along with our training data to fit the model.

```
1 model.fit(x_train, y_train)
```

```
LinearRegression()
```

Figure 15 Building linear regression model part2

The linear regression model finds the best value for the intercept and slope, which results in a line that best fits the data. In Figure 16, we see the value of the model intercept and slop.

```
1 print(model.coef_)
```

```
[0.97190082]
```

```
1 print(model.intercept_)
```

```
-0.0033292527371409954
```

Figure 16 Linear regression model intercept and slope

As shown in Figure 17, we visualize the regression line, which is:

statistics.playCount= 0.9719 + -0.0033292527371409954 statistics.diggCount*

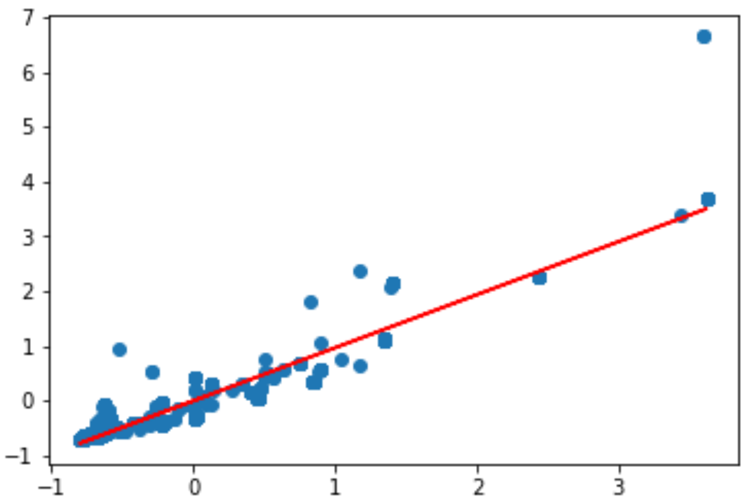


Figure 17 Linear regression visualization

B) Logistic Regression

First: To build the model, we will select the predictor, and as shown in the heatmap (Figure 12), we can indicate that there is no strong correlation between the response 'statistics.shareCount' and any other variable, but there is a moderate correlation with 'statistics.commentCount', so we will use it to build the model.

Second: Figure 18 shows how we fitted the model using the training set (X_train /y_train). The threshold used with the log odds to classify the data points into 0 or 1 is the default (0.5).

```
from sklearn.linear_model import LogisticRegression

logreg = LogisticRegression()
logreg.fit(X_train, y_train)
```

Figure 18 Building Logistic regression model

Figure 19 shows visualization of the model.

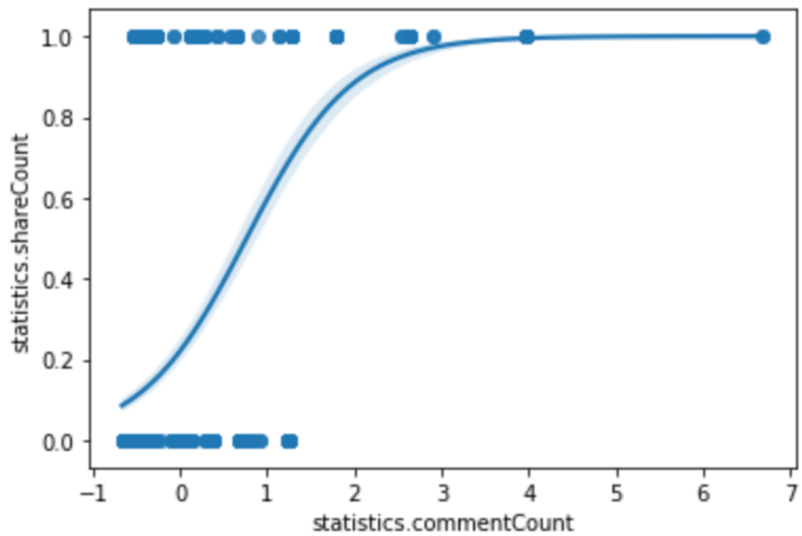


Figure 19 Logistic regression model

- **What are the initial parameters of the model if any?**

  None

4) Evaluation and Results

A) Linear Regression

- **Show your results (tables, figure, confusion matrices …etc)**

  After using the model to predict the test data, here we will use three common evaluation metrics for linear models, which are: Mean Absolute Error (MAE) as shown in Figure 20 , Mean Squared Error (MSE) as shown in Figure 21, and Root-Mean-Square Error (RMSE) as shown in Figure 22.

```
#print result of MAE
metrics.mean_absolute_error(y_test, predictions)
```

```
0.14482263103191442
```

Figure 20 Evaluation metric (MAE)

```
#print result of MSE
metrics.mean_squared_error(y_test, predictions)
```

```
0.047492398250426594
```

Figure 21 Evaluation metric (MSE)

```
#print result of RMSE
np.sqrt(metrics.mean_squared_error(y_test, predictions))
```

```
0.21792750686966203
```

Figure 22 Evaluation metric (RMSE)

In Figure 23, we can see that this model has a much higher R-squared value 0.94 meaning that this model explains 94% of the variance in our response variable, which clearly reveals that our model is doing well and can be used for real-world cases for solving problems.

OLS Regression Results

Dep. Variable:	statistics.playCount	**R-squared (uncentered):**	0.940
Model:	OLS	**Adj. R-squared (uncentered):**	0.940
Method:	Least Squares	**F-statistic:**	2.184e+04
Date:	Fri, 27 Nov 2020	**Prob (F-statistic):**	0.00
Time:	15:42:31	**Log-Likelihood:**	-22.311
No. Observations:	1406	**AIC:**	46.62
Df Residuals:	1405	**BIC:**	51.87
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
statistics.diggCount	0.9693	0.007	147.791	0.000	0.956	0.982

Omnibus:	1260.239	**Durbin-Watson:**	2.032
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	102285.787
Skew:	3.780	**Prob(JB):**	0.00
Kurtosis:	44.095	**Cond. No.**	1.00

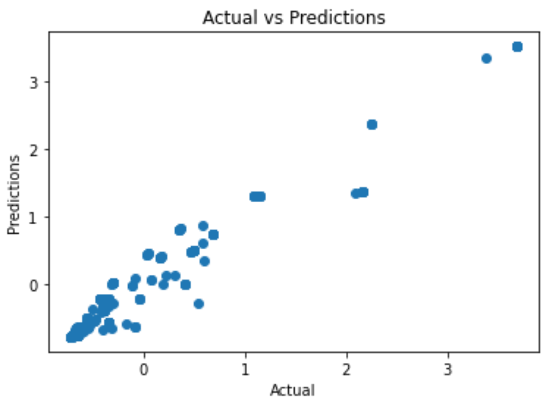
Figure 23 Linear regression evaluation metrics

- **Which evaluation techniques is better and why?**

  We can’t compare the value of different metrics with each other. In comparison, the only thing that make sense is to compare the same measure of error. For example, compare MAE for Method 1 to MAE for Method 2. But when we talk about what’s the more effective matrix to use in our model, we think MAE is a better option because its gives answer about “How far were your off in your predictions, on average?” which is determine the distance between actual values and predictor values.

- **Plot your output (visualization of error)**

  A predicted against actual plot shows the effect of the model and how predicted values close to the actual values. As shown in Figure 24, we can see a low difference and strong linear relation between the actual and predicted values.



	Actual	Predicted
	−0.725116	−0.780310
	−0.580597	−0.606303
	−0.690753	−0.756498
	−0.721401	−0.777176
	−0.718256	−0.778347
	...	...
	−0.109680	−0.013827
	1.099281	1.309733
	−0.569582	−0.493378
	1.148851	1.309733
	−0.704779	−0.774338

Figure 24 Prediction vs actual

B) Logistic Regression

- **Show your results (tables, figure, confusion matrices …etc)**

After using the model to predict the test data, we can evaluate the model, Figure 25 shows the confusion matrix of the model, where it represents the number of correctly/incorrectly classified data points for both classes, 0 and 1.

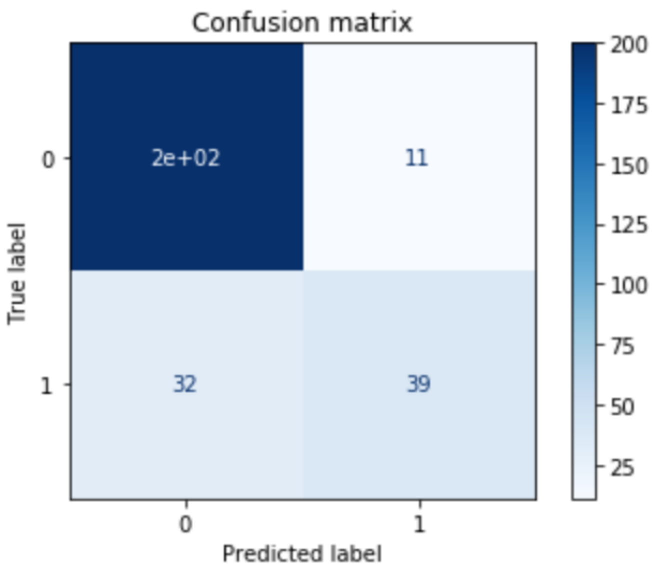


Figure 25 Logistic regression confusion matrix

From the confusion matrix we can calculate accuracy, precision, recall and other performance measures as shown in Figure 26.

	precision	recall	f1-score	support
0	0.86	0.95	0.90	211
1	0.78	0.55	0.64	71
accuracy			0.85	282
macro avg	0.82	0.75	0.77	282
weighted avg	0.84	0.85	0.84	282

Logistic regression evaluation metrics26 Figure

The model accuracy is 85% which means the model performs well at all. The model precision and recall with respect to class 1 are lower than class 0, which indicates the model efficiency in identifying class 1 is lower than for class 0.

- **Plot your output (visualization of error)**

  The scatter plot shown in Figure 27 visualizes the predicted data points, the correctly classified data points colored by green, and the misclassified data points colored by blue.

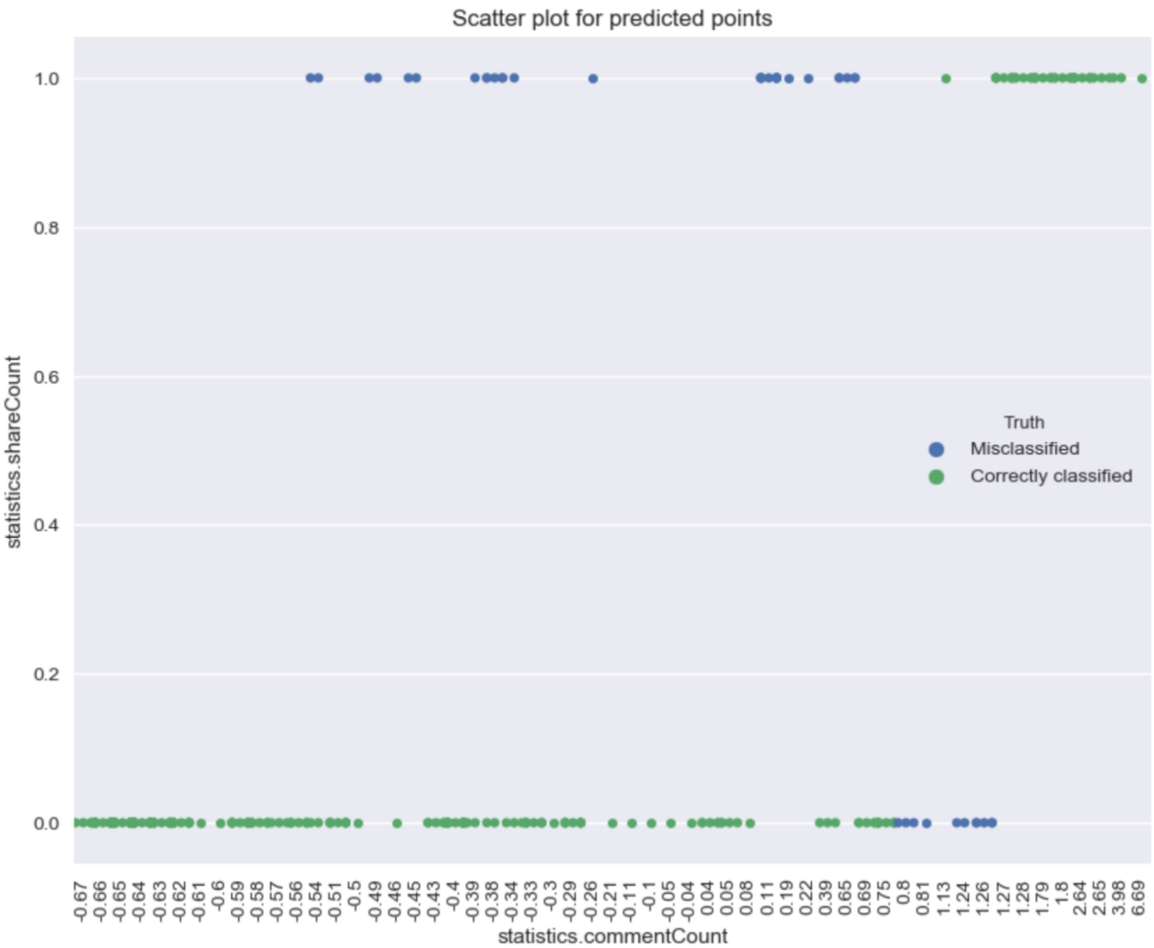


Figure 27 Scatter plot for predicted points

- **Which evaluation technique is better and why?**

  As we are interested in class 1 and care about correctly identifying it, we think precision will give us a good view and perception of the model performance in classifying the points that belong to class 1. The second reason prompts us to prefer precision rather than the other measurements, especially the accuracy, because we can see there is kind of an imbalance in the class labels. Figure 28 shows how the number of points belong to class 0 higher than class 1, and as mentioned in [2], using precision is probably useful when the number of negative data points is larger than the positive, since the precision will not be affected by the negative ones.

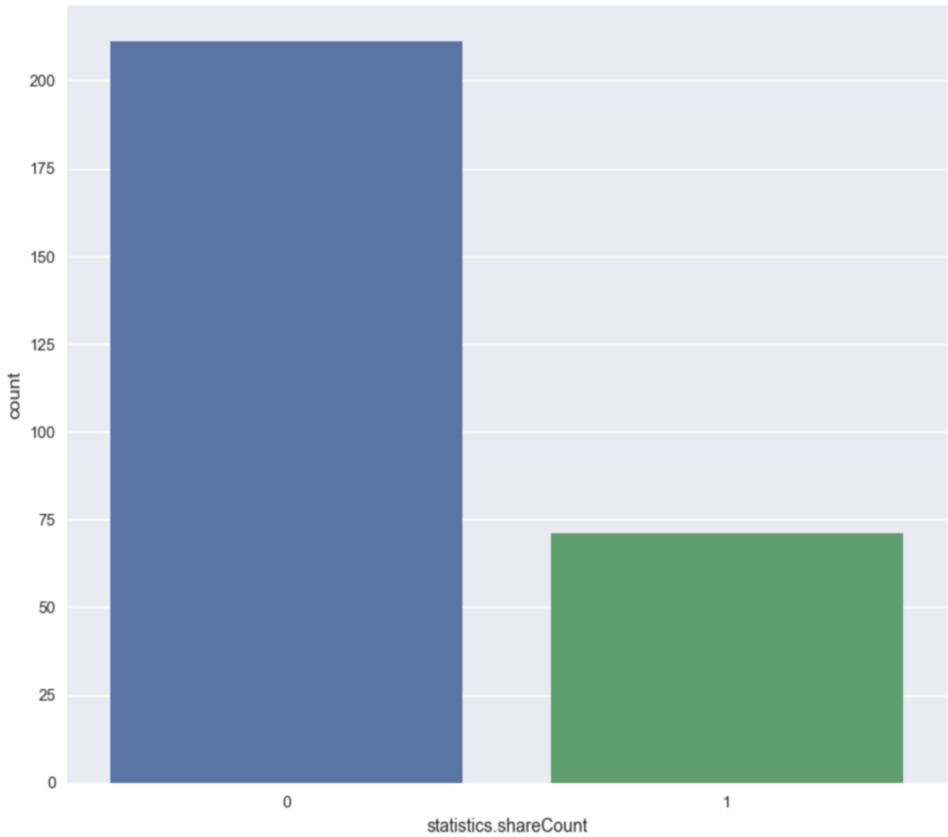


Figure 28 Count of class 1 and 0 instances

5) Analysis and conclusion

- **Explain your results in analytical way instead of numbers. Meaning, why do you think one algorithm performed that way and the other didn't?**

After we have built the two models and evaluate them using evaluation methods, we can conclude in some sentences and talk about their performance and results.

For the linear regression model which identifies the strength of the effect that the dig count has on a play count. In particular, we have proposed a predictor based on the correlation between a response, and train and construct the model based on only the variable that has a strong correlation, As the results showed, we can indicate we reach a powerful and highly persuasive way of demonstrating relationships between the response variable and predictor variable. Moreover, the evaluation techniques that we applied to our model gives powerful and persuasive proof.

For the logistic regression model which predicts if a video is shared widely or not, depending on the numbers of comments in that video, and as the results showed, we can mostly rely on the model for prediction.

The used models were sufficient to perform the prediction tasks and that due to the nature of the data used and its suitability to the algorithm features and capabilities.

Finally, we can say both models can be used as it intended, to help commercial companies deciding where to put their ads.

- **How to improve the models**

Collecting more data will improve the models and gives us a better and accurate results.

For the linear regression model, applying gradient decent to minimize the cost function and optimize our model.

As the logistic regression model affected by the imbalanced data, where the class 0 in more than class 1, we can improve the model by Under-sampling the majority class (class 0) and that's done by reducing the instances from class 0. Another way is by Over-sampling the minority class (class 1) by increasing the size of class 1.

6) References

[1] Medium. 2020. Why Data Normalization Is Necessary For Machine Learning Models. [online] Available at: <https://medium.com/@urvashilluniya/why-data-normalization-is-necessary-for-machine-learning-models-681b65a05029> [Accessed 26 November 2020].

[2] Medium. 2020. What Metrics Should Be Used For Evaluating A Model On An Imbalanced Data Set?. [online] Available at: <https://towardsdatascience.com/what-metrics-should-we-use-on-imbalanced-data-set-precision-recall-roc-e2e79252aeba> [Accessed 27 November 2020].