Ilia Notin

InstaSeer. Report

Introduction

I was always interested in behavioral psychology. It is curious what makes people make certain decisions, why they behave one way or another, sometimes against the logic. Of course, it is hard to answer these questions in all situations. Still, one of them, which seems comparatively simple, is determining why people like what they see. But is it possible to build a model that could predict if an image will be pleasant for most people?

Dataset description

Instagram has an enormous image database “labeled” by the number of likes; hence, I decided to use it. Instagram API does not allow scraping images and other post metadata. However, there is a library called Instascrape based on Beautiful Soup which has the required functionality.

I obtained a raw dataset which had various features, among which I initially chose:

Id	a unique post identifier	used to remove duplicates
height	image height	used to calculate image ratio
width	image width	used calculate image ratio
display_url	image/video url	used to download images for subsequent object detection and color data extraction
is_video	video indicator	used to filter the images
tagged_users	names of tagged profiles	used to calculate their number
caption	text under the post	used for text vectorization
upload_date	timestamp of post upload	used to define day of the week
hashtags	list of hashtags	used to calculate their number
likes	number of likes	used to calculate target variable
numOfSubs	number of followers	used to calculate target variable

Then I cleaned the dataset and generated new features. The final dataset which was used for modeling contained the following variables:

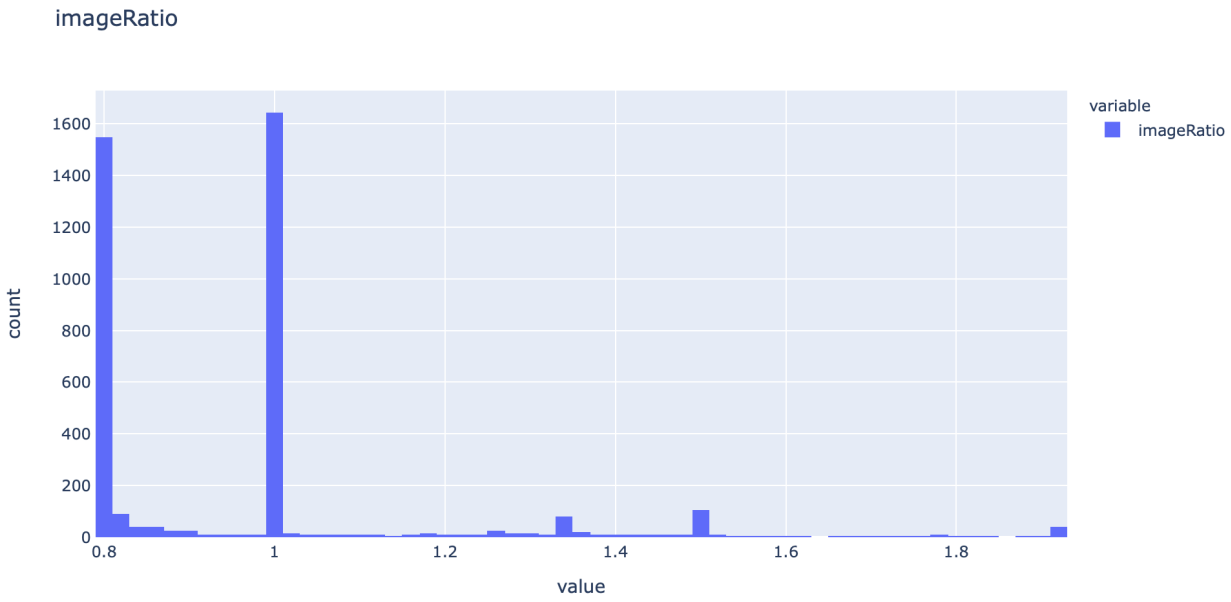
imageRatio	float	width to height image ratio
numOfHashtags	int	number of hashtags
numOfTaggedUsers	int	number of tagged users
lenOfCaption	int	number of characters in caption
uploadDayOfWeek	int	day of week number of post upload

airplane…zebra	int	one hot encoded objects present in the image
b_0…b_9	float	binned histograms of blue channel (10 bins)
g_0…g_9	float	binned histograms of green channel (10 bins)
r_0…r_9	float	binned histograms of red channel (10 bins)
Target variable		
weightedLikes	float	the number of likes divided by the number of followers, assuming that an equal percentage of followers tend to click like button

Analysis of variables

First, let's plot histograms for each variable to get some understanding of the data.

Image ratio

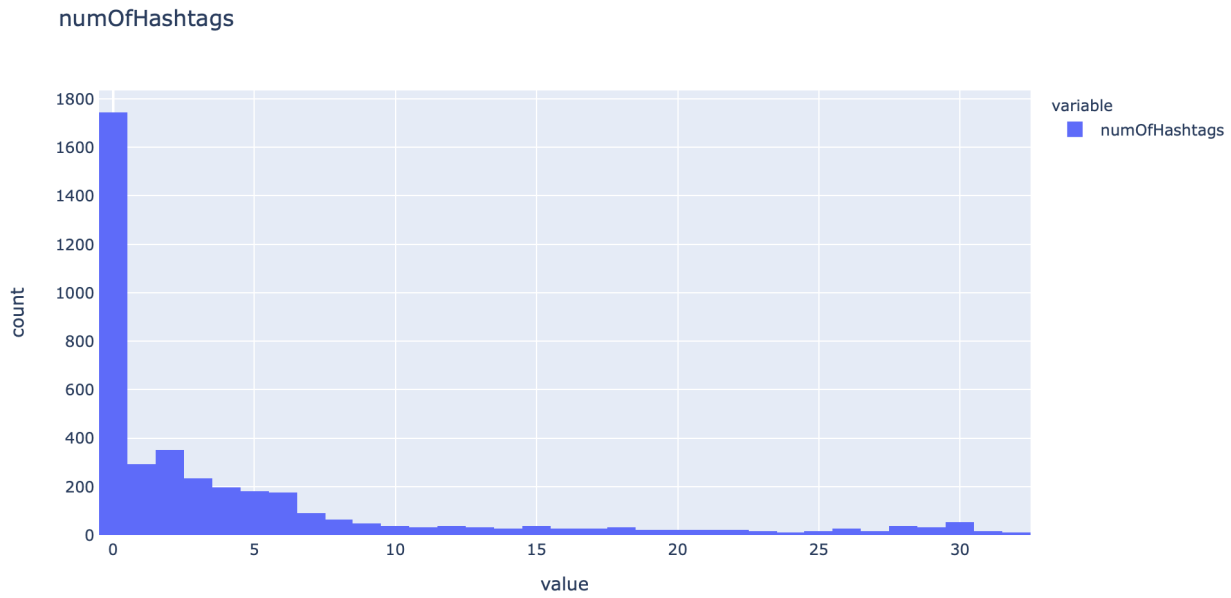


It can be concluded that there mainly 3 types of images:

- Square (ratio=1)
- Rectangular with ratio=0.8
- Having other ratios

Probably I would need to convert this variable categorical with these three categories

numOfHashtags



Most of the posts do not have hashtags. Those which have them mostly contain 2 hashtags (up to 6).

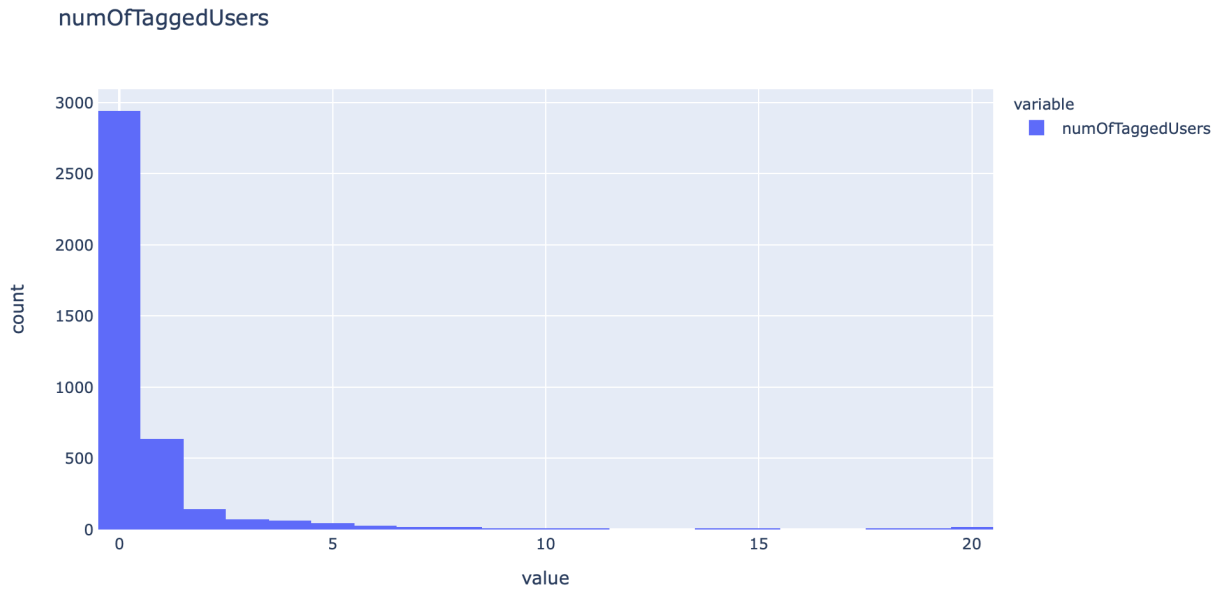
This variable could also be converted to a categorical with values:

0 – no hashtags

1 – from 1 to 6

2 – more than 6

numOfTaggedUsers



Similar situation as with the previous variable.

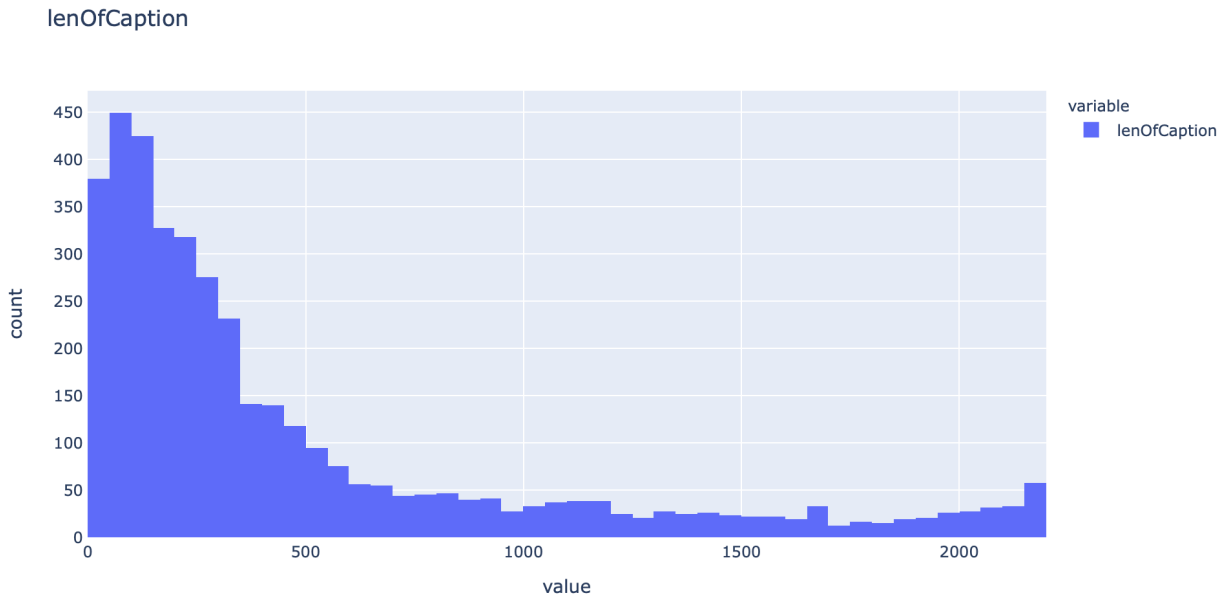
I would try to convert to categorical with 3 categories:

0 – no tags

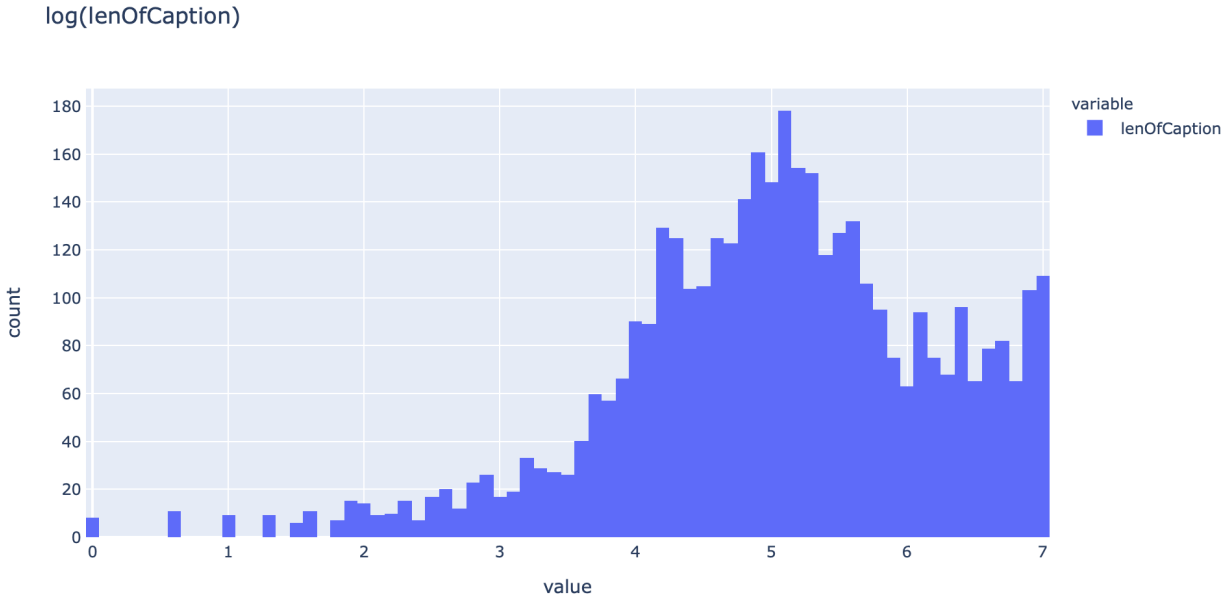
1 – 1 tag

2 – more than 1 tag

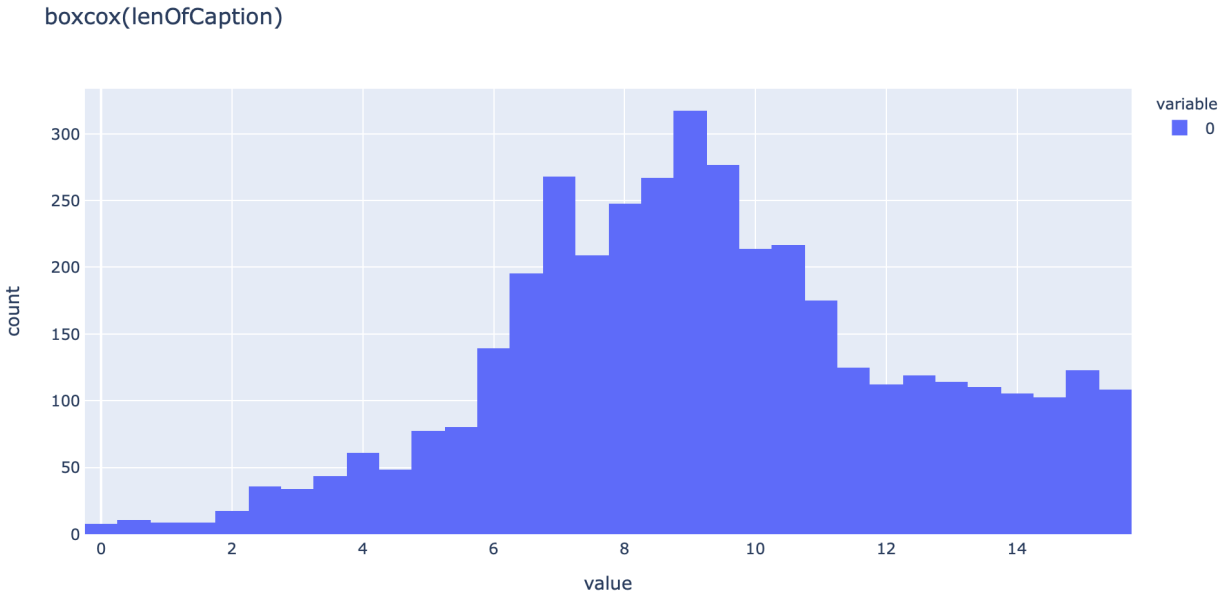
lenOfCaption



This variable looks like having a distribution similar to log-normal. Probably log-transformation should be applied.



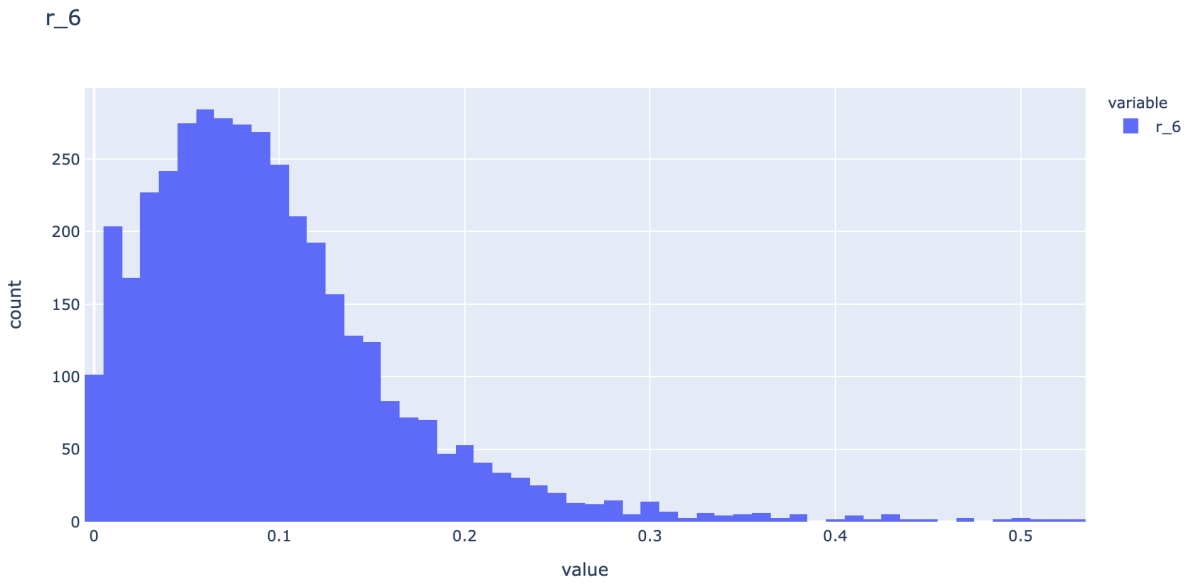
Not ideal, but better. Let's try the Box-Cox transformation:



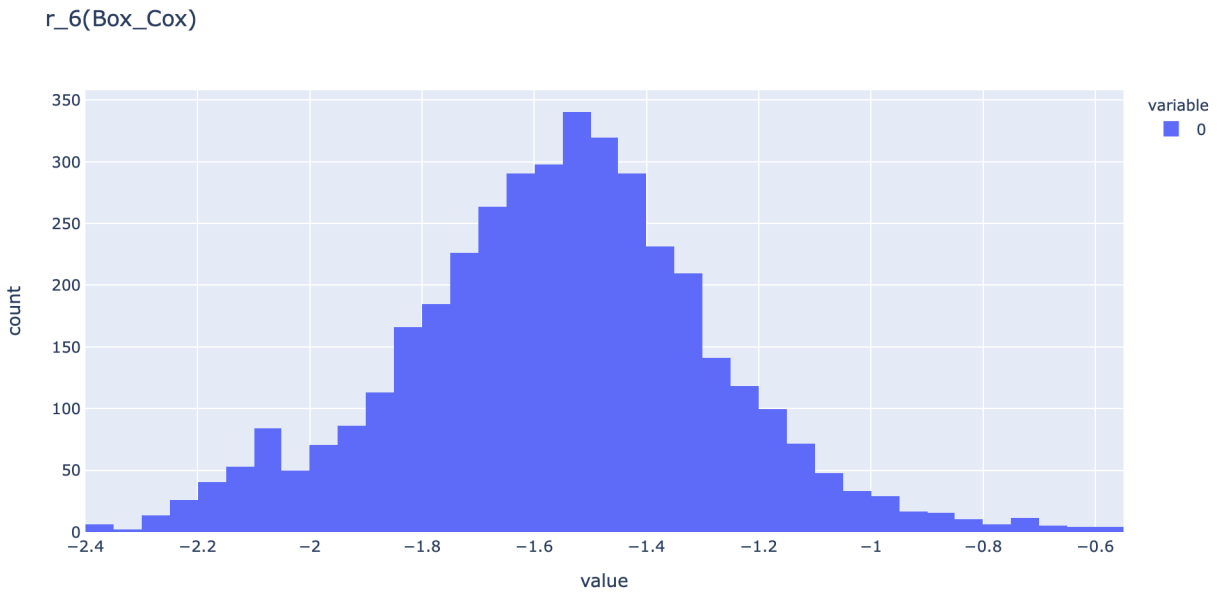
A little bit better.

b_0…b_9, g_0…g_9, r_0…r_9

Most of these variables have distributions close to log-normal one.

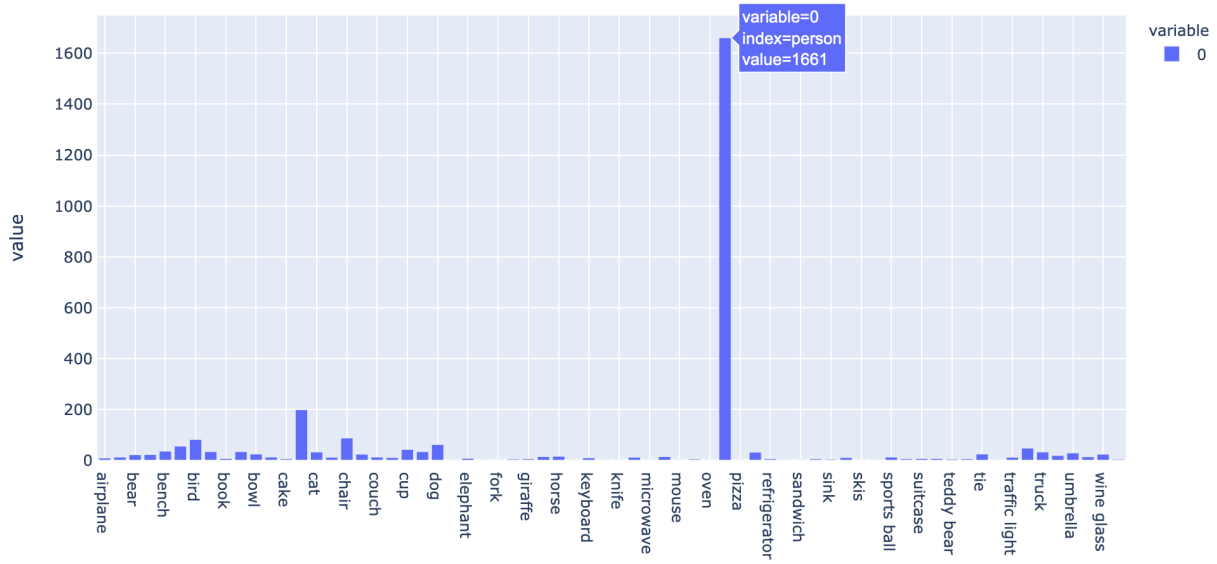


I tried to log transform it, but it doesn't help much. The Box-Cox transformation could be helpful, but it requires non-zero positive values. What I can do, is to add a tiny fraction to 0 values so that it does not affect the whole picture, but it will allow the transformation.



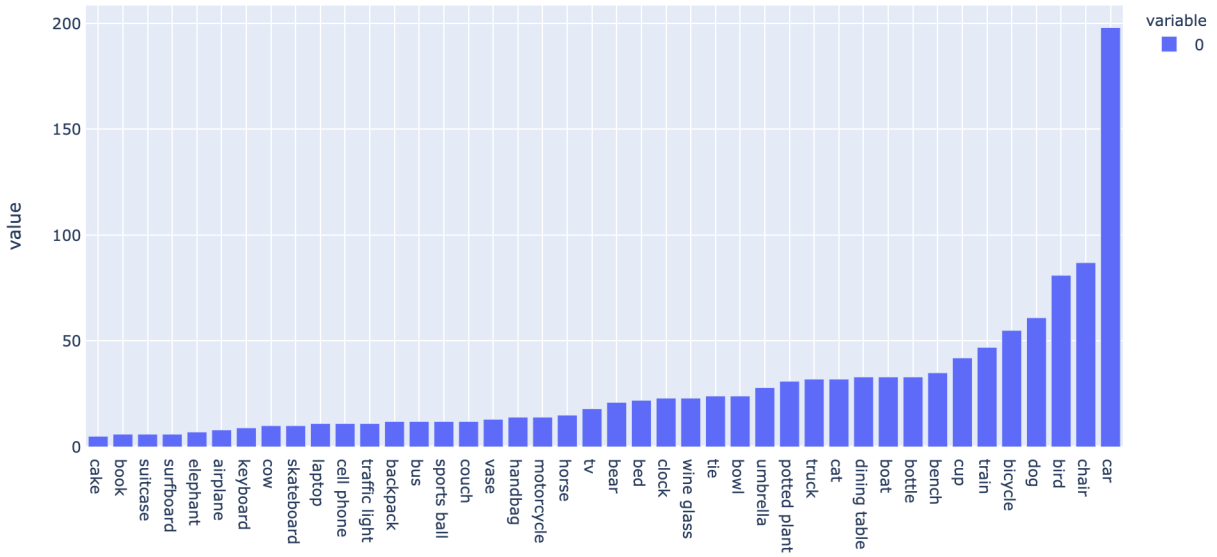
Much better.

Objects

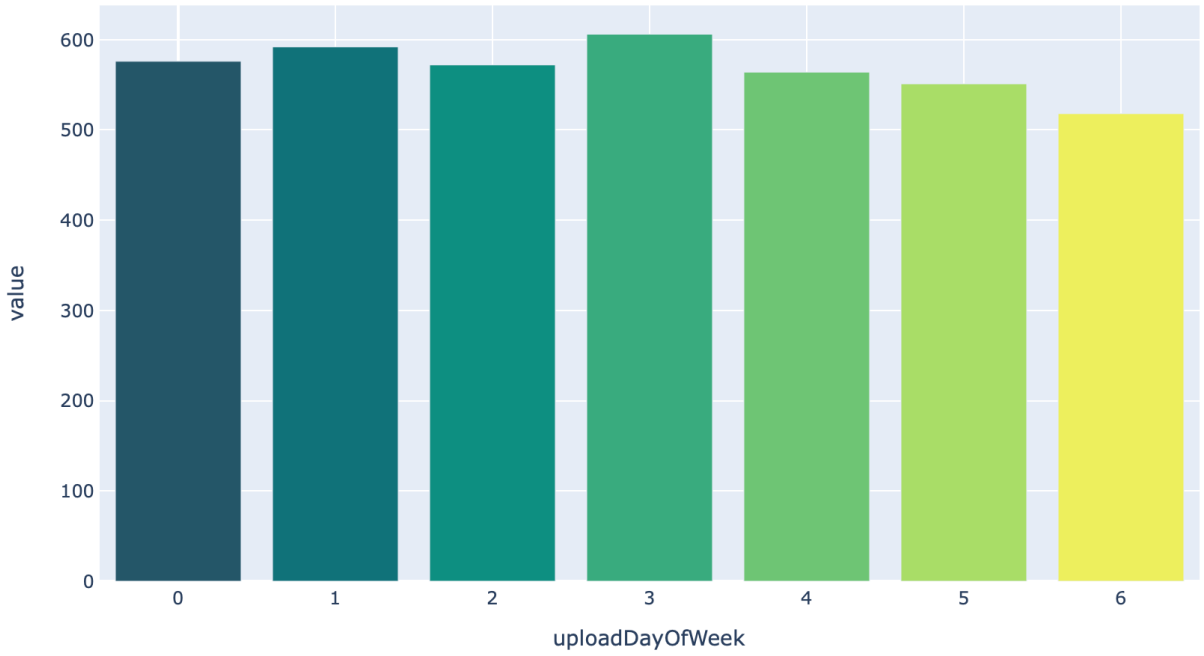


Most of the posts contain people; therefore, I removed this column.

This is how top-25 most frequent objects look like:



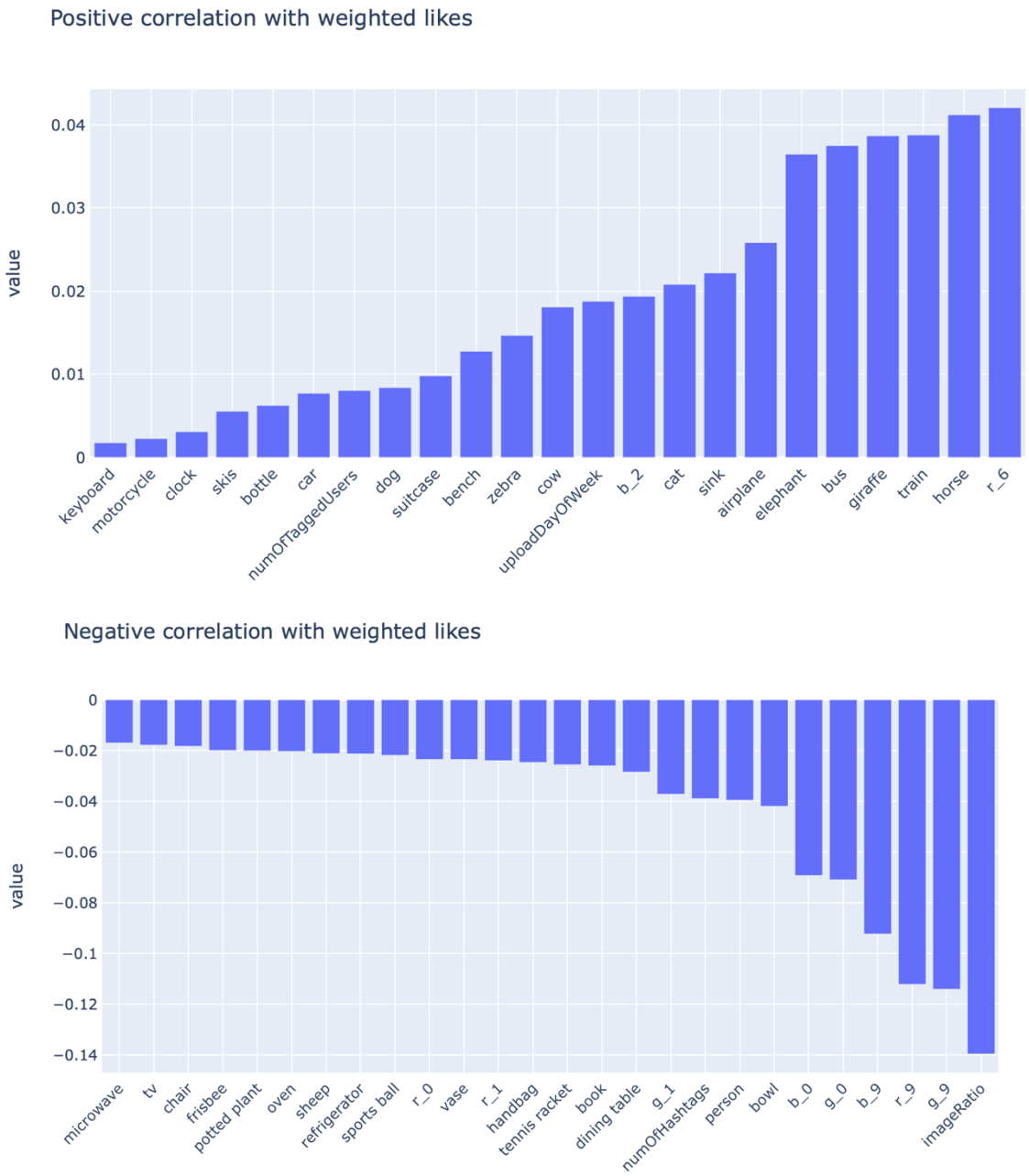
uploadDayOfWeek



This variable is evenly distributed. The only noticeable thing is that there are not a lot of posts during the weekend.

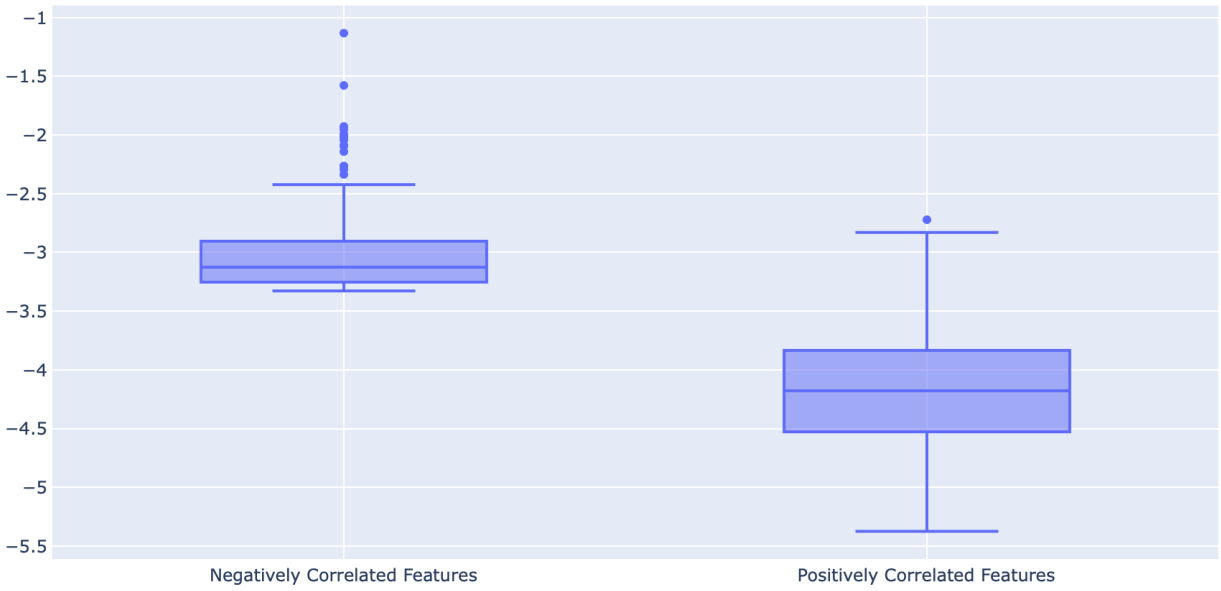
Target variable

Feature correlation



Correlation values are pretty low, but it's interesting that wider images with horses, trains and giraffes with no bright colors should gain more likes. Here it's interesting to compare samples with opposite characteristics to see how significant the difference between mean log-weighted likes values is.

For that purpose, I took 2 samples with 300 records in each. The samples were taken from dataframe sorted by top-10 columns from positive and negative correlations. Their boxplots are provided below.



Initially I wanted to conduct one-way ANOVA but the samples didn't meet its requirements: they had different variance, not normally distributed residuals and values. Therefore, I conducted the Kruskal-Wallis H-test which requires only independence of samples (the equality of distribution shapes can be ignored if null hypothesis is rejected [http://rcompanion.org/handbook/F_04.html]).

The result of Kruskal_Wallis H-test showed that these two groups are stochastically different (H>Hc, p<<0.05):

```
1 print(scipy.stats.kruskal(negSample, posSample))
2 print('Critical value:',chi2.ppf(0.95, 1))
```

```
KruskalResult(statistic=412.95826257493144, pvalue=8.321359029491769e-92)
Critical value: 3.841458820694124
```

Therefore, the elimination of negatively correlated features can lead to increase of number of likes.

The research questions

- Is it possible to predict number of likes based on image data?
- What are the factors that affect most the number of likes?

There are two approaches for solving this task which I will try to implement.

In this case I will apply following models:

- Random forest regressor
- XGBoost regressor
- K-neighbors Classifier
- Decision Tree Regressor
- Voting Regressor

The first one is regression analysis, which is based on numerical values of likes.

The second one is classification analysis with previously binned likes into several categories.

In this case I will apply following models:

- Random forest Classifier
- XGBoost Classifier
- K-neighbors Classifier
- Voting Classifier
- Convolutional Neural Network

Regarding model selection for all models except CNN, I implemented Grid Search in case of a few model parameters or Randomized Search when I had to check wide ranges of parameters.

Regarding train and test split I will use a standard 80/20 ratio for train and test sets respectively.

In the tables below there are the obtained results.

Regression models

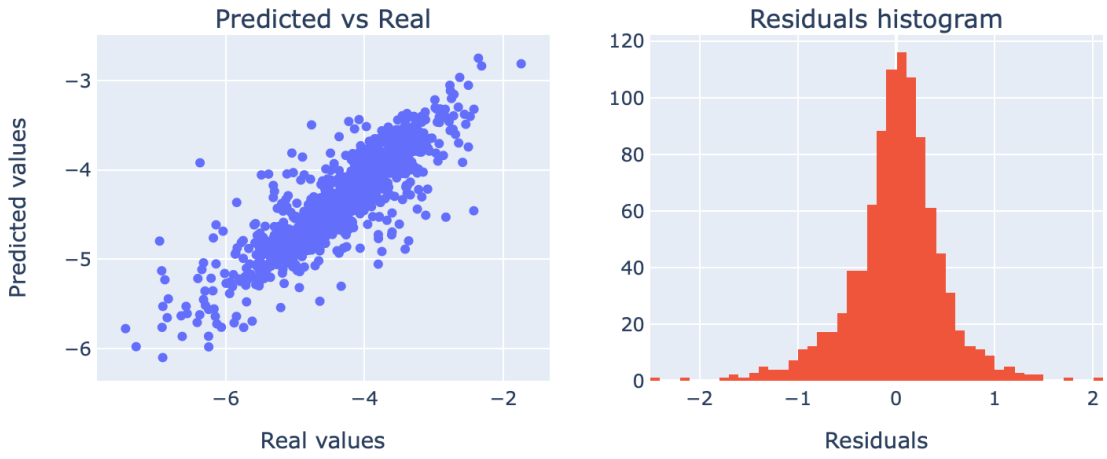
Model name	Model selection method	Train set		Test set	
		RMSE	R2	RMSE	R2
Random forest regressor	RandomizedSearchCV	0.477	0.663	0.469	0.684
XGBoost regressor	RandomizedSearchCV	0.598	0.471	0.574	0.525
K-neighbors Classifier	RandomizedSearchCV	0.814	0.0206	0.824	0.0248
Decision Tree Regressor	RandomizedSearchCV	0.755	0.156	0.763	0.162
Voting Regressor	None	0.272	0.890	0.724	0.247

Classification models

Model name	Model selection method	Train set		Test set	
		Accuracy	Weighted F1-score	Accuracy	Weighted F1-score
Random forest Classifier	RandomizedSearchCV	0.85	0.85	0.50	0.51
XGBoost Classifier	RandomizedSearchCV	0.92	0.92	0.52	0.52
K-neighbors Classifier	GridSearchCV	0.57	0.58	0.35	0.36
Voting Classifier	-	1	1	0.49	0.49
Convolutional Neural Network	-	0.41	0.44	0.24	0.31

From the analysis of the obtained results, I can conclude that the metrics are not outstanding. The best regression model which is Random Forest regressor demonstrates R2 score of 0.684 though in psychology domain it is considered high. The plot below shows predicted vs real values scatter plot which is close to diagonal line.

```
True mean value          : -4.3392412778049
Predicted mean value     : -4.3392412778049
MSE                      : 0.21995278949104724
RMSE                     : 0.4689912467104767
3sigma                   : 1.4077055860795578
Var. coeff.              : 0.10809721714069728
Var. coeff. (3 sigma)    : 0.3242916514220918
R2-score                 : 0.6838492567397988
```



Feature importance table below states that among the most important variables are numOfHashtags, numOfTaggedUsers, and it makes sense as hashtags and tagged people lead to attraction of the audience.

feature	importance
imageRatio	0.083863
numOfHashtags	0.065383
numOfTaggedUsers	0.043436
person	0.038311
bench	0.027216
bowl	0.026880
r_9	0.021013
g_9	0.020162
cup	0.019892
bird	0.019716
lenOfCaption	0.019696
r_2	0.018836
dining table	0.018650
cake	0.018000
car	0.017774

Regarding classification models the best results are obtained by XGBoost Classifier. However, it looks overfitted which should be fixed. In the table below there is a confusion table and classification report of that model.

```
Scores for train set:
              precision    recall  f1-score   support

           0       0.95      0.90      0.92      1384
           1       0.88      0.94      0.91      1188
           2       0.93      0.92      0.92      1273

    accuracy                           0.92      3845
   macro avg       0.92      0.92      0.92      3845
weighted avg       0.92      0.92      0.92      3845

Scores for test set:
              precision    recall  f1-score   support

           0       0.62      0.52      0.57       355
           1       0.37      0.43      0.40       279
           2       0.57      0.59      0.58       328

    accuracy                           0.52       962
   macro avg       0.52      0.52      0.52       962
weighted avg       0.53      0.52      0.52       962
```

Conclusion

The models created within this project did not show outstanding results. I suppose that number of records should be increased which I will implement later. However, Random Forest regressor can be used for comparative choice of the posts.

Conducted Kruskal-Wallis H-test showed significant differences between Instagram posts with dominating positively and negatively correlating features which allows to determine factors affecting success of a post.

As for the future work, I plan to increase number of records, fine tune neural networks and apply clusterization to generate new features. I would also pay some attention to tuning of models which demonstrate overfitting.

Appendix A. Model parameters

Regression models

Model name	Model parameters
Random forest regressor	{'n_estimators': 175, 'min_samples_split': 10, 'min_samples_leaf': 4, 'max_features': 'auto', 'max_depth': 110, 'bootstrap': True}
XGBoost regressor	{'min_child_weight': 3, 'max_depth': 8, 'learning_rate': 0.1, 'gamma': 0.1, 'colsample_bytree': 0.4}
K-neighbors Classifier	{'leaf_size': 20, 'n_neighbors': 100, 'weights': 'uniform'}
Decision Tree Regressor	{'splitter': 'best', 'min_weight_fraction_leaf': 0.1, 'min_samples_split': 4, 'min_samples_leaf': 1, 'min_impurity_decrease': 1e-07, 'max_leaf_nodes': None, 'max_features': None, 'max_depth': 240, 'ccp_alpha': 0.001}
Voting Regressor	Random forest regressor XGBoost regressor K-neighbors Classifier Decision Tree Regressor All with default parameters

Classification models

Model name	Model parameters
Random forest Classifier	{'n_estimators': 1047, 'min_samples_split': 5, 'min_samples_leaf': 4, 'max_features': 'sqrt', 'max_depth': 10, 'bootstrap': False}
XGBoost Classifier	{'min_child_weight': 7, 'max_depth': 8, 'learning_rate': 0.15, 'gamma': 0.5}
K-neighbors Classifier	{'leaf_size': 20, 'n_neighbors': 25, 'p': 1, 'weights': 'distance'}
Voting Classifier	Random forest Classifier XGBoost Classifier K-neighbors Classifier

	All with default parameters

Convolutional Neural Network	model = Sequential()
	# convolutional layer
	model.add(Conv2D(25, kernel_size=(3,3), strides=(1,1), padding='valid', activation='relu', input_shape=(128,128,3)))
	model.add(MaxPool2D(pool_size=(1,1)))
	# flatten output of conv
	model.add(Flatten())
	# hidden layer
	model.add(Dense(100, activation='relu'))
	# output layer
	model.add(Dense(1, activation='softmax'))
	# compiling the sequential model
	model.compile(loss='categorical_crossentropy', metrics=['accuracy'], optimizer='adam')