Predicting the number of Likes on Instagram Posts

Rachita Saha (2019082) Dipankar Jiwani (2019037) Isha Sehrawat (2019046)

1. Introduction

Increasing popularity of social media has made it a platform for promoting different kinds of businesses. We attempt to perform multimodal analysis using the post image, caption and some additional data about the user to predict the number of likes the post will receive. Predicting how well a post will perform will be highly beneficial for users and help them curate better content.

2. Related Works

Owing to the growing popularity of social media, regular users are highly interested in predicting the popularity of their posts in social networks and recently, it has received a great deal of attention from researchers.

Dugue et al [1] explored this problem by developing a base XGBoost model on basic user data and then added features generated using NLP (on user bio, caption, and location) and CNN (on image). They found that the most important feature was the average likes on previous posts of the user. They demonstrated that training CNN on the images alone, extracting some features from its output, and integrating it with the joint model decreased the RMSE to 2837. Carta et al [2] proposed an original definition of popularity for Instagram posts, by modeling it as a binary classification problem. They introduced a new set of enriched features, by analyzing the sentiment score, emojis, and the popularity of the chosen hashtags. Priego [3] developed a three-branch neural network to train the mixed data by defining three different model architectures to process each type of data. After this, the three branches were concatenated to make a final prediction. Zamorano et al [4] analyed different aspects of image data such as the presence of a human subject, a group of people, and smiling faces. The best results were delivered by the Multilayer Perceptron (MLP) with Mean Absolute Percentage Difference of 55.40%, RMSE of 0.0807 and two-layer neural networks (Mean Absolute Percentage Difference: 55.22%, RMSE: 0.008021).

3. Dataset and Evaluation

3.1 Dataset

The dataset used [5] consists of data of popular Instagram users featuring on the Inconosquare Index Influencers [6], a list of 2000+ Instagram influencers.

It provides the metadata from the profile and 17 latest posts from the users in separate JSON files for each user. In order to perform multimodal analysis, we downloaded images from the post URL by crawling each post. The scraper was coded using Selenium, which allowed us to crawl webpages and download images from them. Instagram has recently restricted the number of login requests to 5. Hence, we used cookies to remain logged in to the app while downloading images. We created a CSV file for training consisting of 1016 images from 371 users. We used a 75-25 train-test split for Linear Regression, SVM and RandomForestRegressor. For Neural Network (train-test into 75-25) and further train-validation as 80-20.

3.2 Feature Extraction

The new features extracted from the dataset were average and standard deviation of the likes on most recent posts, weekday on which the image was posted, average no. of posts per week by the user, interaction (avgPrevious10Likes/numberFollowers)*100), and follower-following ratio.

The flattening of the image and passing it to the models was a basic approach and didn’t prove to be giving good results. The Kaze algorithm is used to extract image features, which is a state-of-the-art computer vision algorithm. Using this, we extracted a feature vector of size 2560 for every image to represent it and appended it to the other data associated with the image.

3.3 Evaluation Metrics

We used Root Mean Squared Error (RMSE) and Mean Absolute Percentage Error to evaluate the models.

4. Methodology

To understand the data better, we plot a heatmap (Fig 1) to check the correlation between different variables. As expected, the numberFollowers, mean, and std of the likes positively correlates with the numberLikes on posts. The metadata of the post, such as the presence of a caption, location, length of caption, number of tags, and mentions do not have a significant impact on the number of likes. On plotting the histograms (Fig 2) and box plots (Fig 4-5) of the different features, we observe that all the distributions are highly right-skewed. All histograms have a clear single peak. We see a large number of outliers in all the plots. Some outlier values are quite far away from the median due to the presence of users (such as celebrities) with an abnormally high number of followers and consequently numberLikes. The scatter plots (Fig 6) show that for most users with follower count in the range 0 - $5*10^6$, the number of likes on their post lies in the range 0 - $2*10^5$ with a few outliers. A clear positive linear relationship is observed between avgPrevious10Likes and numberLikes.

Our problem is predicting the number of likes which burns down to a regression problem. Our most intuitive approach for the baseline model was Linear Regression. After some research, we decided to use SVM because we had limited data samples. RandomForestRegressor as an ensemble method would help with overfitting prevention, and CNN (Fig 9) is ideal for dealing with images.

Our analysis of the results obtained in the Interim Project Evaluation showed overfitting in the Linear Regression model with very high errors hinting that the bias is high and the model is not complex enough to capture the data, which suggests that the data is linearly inseparable. SVM showed significant overfitting as well. However, RandomForestRegressor worked well because it uses a voting strategy.

4.1 Increasing the size of the dataset

Initially, the training dataset consisted of 1000 samples, which is one of the reasons why Linear Regression overfitted the dataset. To avoid overfitting, our first approach was to increase the size of the dataset. We added 58% more samples (1580 total samples) and, even without normalization or PCA, we obtained results that were comparable to the best metrics obtained in the interim evaluation results.

4.2 Normalization of Input and Output variables

We normalized the input and output variables with the maximum value for each feature. The mean and standard deviation of the ‘numberLikes’ after normalization were 0.023091 and 0.068992 respectively. We observed significant improvement in the results obtained after normalization.

4.3 Principal Component Analysis

Earlier the dataset size was 1000, after increasing it to 1580, we still did not achieve good enough results on the train-test-val set. The reason is the scarcity of data samples. We used the Kaze algorithm to generate image features leading to a total of 2680 features which is way larger than 1580, the number of data samples. So, we decided to use PCA to reduce the number of features to an optimal number. For the same, we plotted the metrics vs the number of features curve for different models. The same plot for the Linear Regression model is shown below (Fig 10) and 800 is the best value for the final feature size. The results obtained after doing PCA on different models improved significantly.

4.4 Hyperparameter tuning

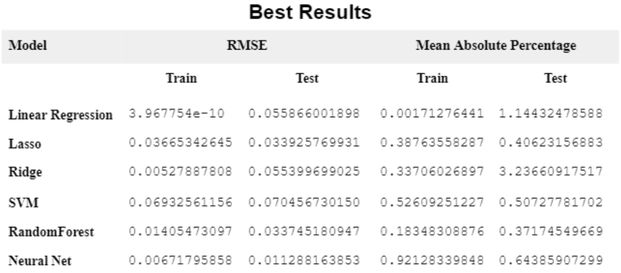
To ensure we get good results, we tuned the parameters for all our models. For the linear regression model, we passed normalized inputs. The model was overfitting, so we decided to use regularization. Ridge and Lasso performed better. For choosing the appropriate value of alpha, we plotted the RMSE vs alpha plot for training and testing datasets (Fig 11). Alpha equal to 0.0001 seemed to perform the best, so we chose it to do the training for L1 and L2 regularization. For SVM, because of the data being scarce and complex, ‘rbf’ filter is used with gamma equal to 0.001 and C equal to 1e3 which was found after trying out certain values. The number of estimators in the Random Forest Regressor is 150 which helps it with overfitting.

4.5 NN architecture

The neural network consists of 2 networks, one a CNN to deal with the images and one MLP to deal with the metadata. The CNN consists of 3 sets of 3X3 convolution filters with relu activation function, batch normalization, and max 2D pooling of size 2X2. Its output is flattened and passed through a dense network of 16 nodes (with batch normalization and dropout=0.5), then 4 nodes(dropout=0.5), and finally a single output node, each with relu activation function. The MLP consists of 3 layers with 8, 4, and final 1 output node with activations as relu, relu and linear respectively. We finally concatenate the output of the two models and pass them into the dense neural network with 4 nodes and finally one output node with activations as relu and linear. The loss vs epoch plot below shows that the training was done correctly (Fig 12). We did an early stopping at 100 epochs to avoid overfitting.

5. Results

The results of various models have been shown through the Table 1.



Best Results

Model	RMSE		Mean Absolute Percentage	
	Train	Test	Train	Test
Linear Regression	3.967754e-10	0.055866001898	0.00171276441	1.14432478588
Lasso	0.03665342645	0.033925769931	0.38763558287	0.40623156883
Ridge	0.00527887808	0.055399699025	0.33706026897	3.23660917517
SVM	0.06932561156	0.070456730150	0.52609251227	0.50727781702
RandomForest	0.01405473097	0.033745180947	0.18348308876	0.37174549669
Neural Net	0.00671795858	0.011288163853	0.92128339848	0.64385907299

Table 1: The best results obtained on baseline and advanced models

The mean of the numberofLikes after normalizing the data is 0.023091 and the standard deviation is 0.068992. Based on this, we observe the RMSE is well less than the standard deviation of the numberofLikes for all our models on the test set. Another insight into the result is given by the mean absolute percentage error given by:

$$M = \frac{1}{n} \sum_{t=1}^{n} \left| \frac{A_t - F_t}{A_t} \right|$$

Where A_t is the actual value, F_t is the predicted value, and n is the number of data samples.

When comparing the results on the test set, we observe the NN performs the best (Fig 13-15). The neural network with the help of early stopping of the training and extensive use of dropout was able to achieve less bias and variance for the model. The RandomForestRegressor, which is an ensemble method is also producing sufficiently good results after the NN. The train and test errors are reasonably close indicating the model is not very overfitting. The performance of the RandomForestRegressor is followed by the Linear Regression. However, the Linear Regression models clearly overfit, owing to the lack of data and a high number of features (even after applying PCA), with a big gap between training and test metrics. The SVM model seems too simple to capture the data relationship since the bias is high, this could be due to the number of data points and the complex relation between them. The memory required for storing of kernel matrix for SVM increases significantly with the number of data points so it is not feasible for a dataset like ours. Our dataset also had a lot of outliers as it consisted of very famous influencers with less popular ones too. The figure (Fig 16) shows the comparison of the bias and variance of the different models we used.

On analyzing the cases where the absolute percentage error was greater than 1.5 times the test data’s mean absolute percentage error, we did a separate analysis. On plotting bar graphs, scatter plots, to analyze the distribution of the metadata of wrong cases, we observe these samples were roughly from the same distribution of training and test samples and sampled from the area around the mean of the test set. However, the problem might have been caused due to images of such samples which included animated characters, memes with text written on them, random colored images, etc. Due to the small training data, the model might have not learned these features of starkly different images. These images and the plots of their distribution are given below (Fig 17-18).

When comparing our best result with the one in Dugue et al [1], which used a similar approach of multimodal analysis to predict likes, our RMSE values of the NN model (Fig 19) were better. However, when compared to the works of Zamorano et al [4], who analysed images with detection of humans, faces, etc., had results slightly better than us, with RMSE and MAPE a bit less than what we have achieved.

We observe that a central problem of our whole project is the lack of data. We had achieved concrete results which can still be improved if the number of data samples is increased. Caption analysis using NLP can also be included. Different Neural Network architectures can be implemented to fit the use case.

6. Contributions

6.1 Deliverables

Initially, we planned on performing analysis only on the post metadata. However, after discussing the project with the TA, we decided to perform multimodal analysis on the post image as well as metadata. Hence, we ended up achieving much more than was planned in the proposal, and the deliverables were redistributed among the team members.

Rachita wrote the code to download the images from the post URL, generated a dataset of 1580 images, and extracted additional features from the data such as frequency of posting in a week, average and std of likes on recent posts, etc. She preprocessed the data and performed exploratory data analysis to identify the most important features (histogram, correlation heatmap, scatter plots, bar plots).

Dipankar extracted image features using the Kaze algorithm for Non-DL models, implemented the Neural Network model, and experimented with its architecture to improve performance. He compared the results and performed error analysis for all the models as well as for the specific error cases.

Isha was also involved in feature extraction and engineering, she ran the basic models for linear regression, SVM, and RandomForestRegressor, did hyperparameter tuning, generated the plots for analysis and comparison of models, and performed error analysis.

6.2 Individual Contributions

A brief description of the contributions by the individual members is given in the subsection above. The code files contributed by each member are as follows:

Rachita:

- img_download.py
- generate_csv.py
- remove_folders.py
- ML_Project_Rachita.ipynb

Dipankar:

- ML_Project_Dipankar.ipynb

Isha:

- ML_Project_Isha.ipynb

Figures

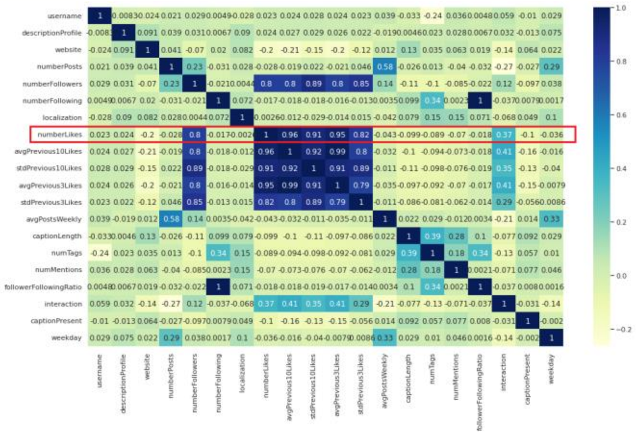


Figure 1: Correlation Heatmap

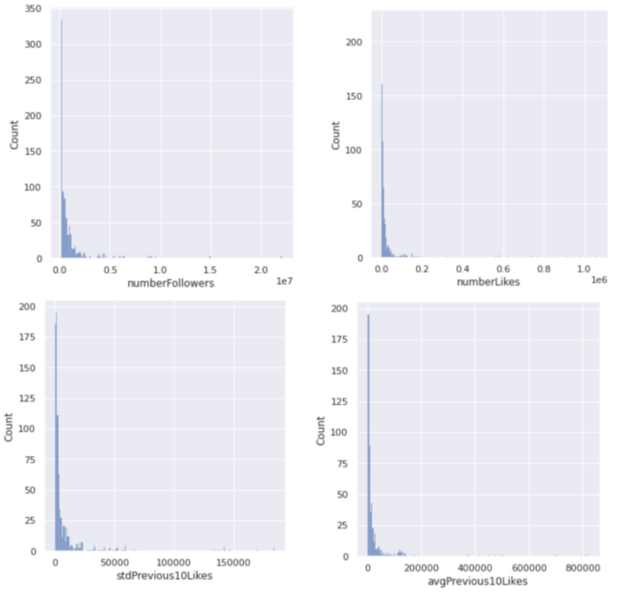


Figure 2: Histogram of numberFollowers, numberLikes, avgPrevious10Likes, stdPrevious10Likes

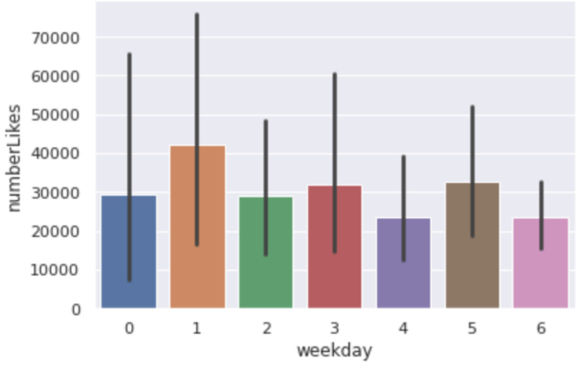


Figure 3: Bar plot of average number of likes on each weekday

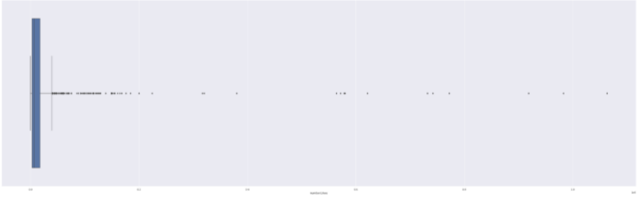


Figure 4: Box plot of number of likes

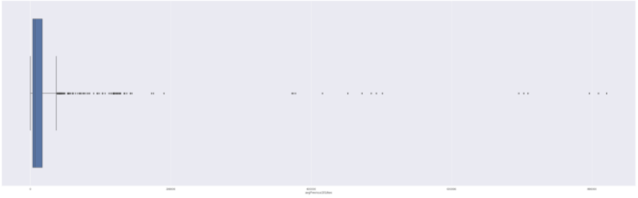


Figure 5: Box plot of averagePrevious10Likes

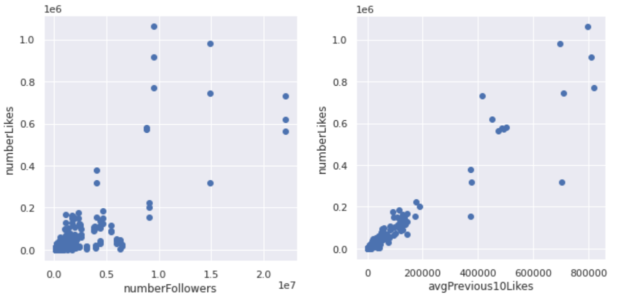


Figure 6: Scatter plots of numberFollowers and avgPrevious10Likes with numberLikes

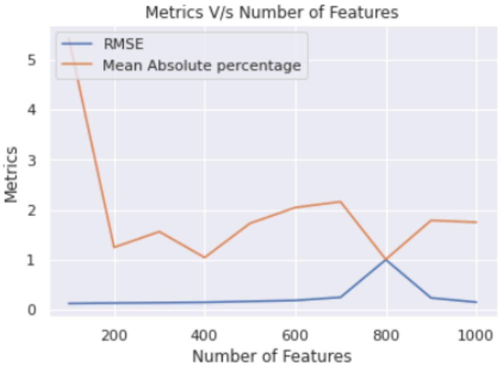


Figure 7: Metrics vs Number of Features (PCA)

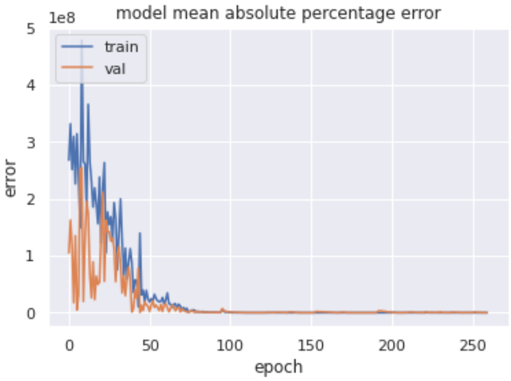


Figure 8: MAPE vs Epoch (Neural Network)

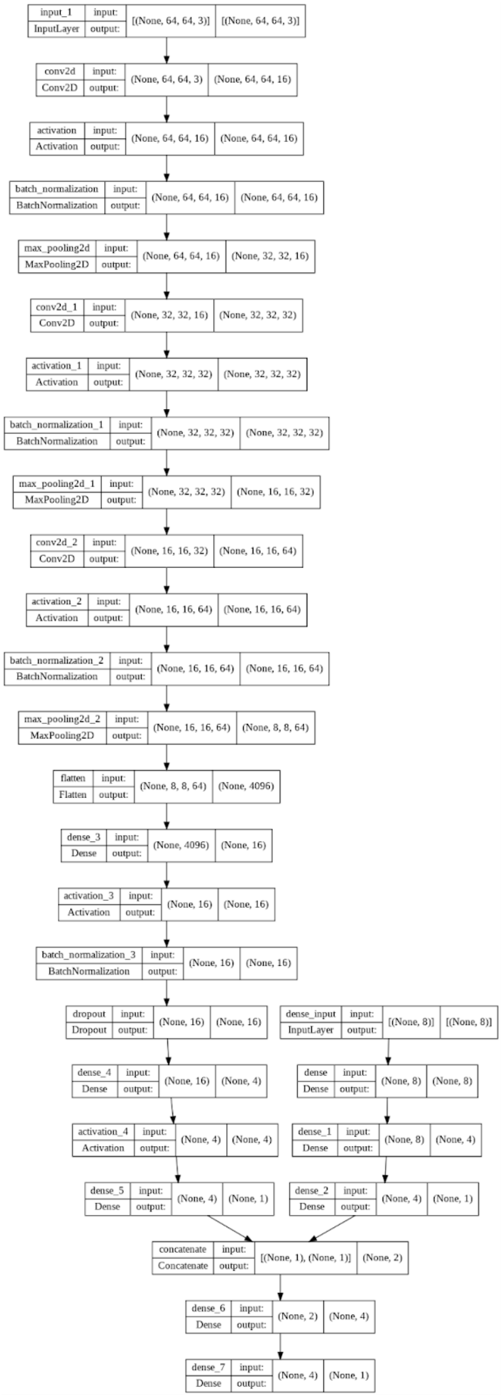

input_1 InputLayer input: output: [(None, 64, 64, 3)] [(None, 64, 64, 3)]
conv2d Conv2D input: output: (None, 64, 64, 3) (None, 64, 64, 16)
activation Activation input: output: (None, 64, 64, 16) (None, 64, 64, 16)
batch_normalization BatchNormalization input: output: (None, 64, 64, 16) (None, 64, 64, 16)
max_pooling2d MaxPooling2D input: output: (None, 64, 64, 16) (None, 32, 32, 16)
conv2d_1 Conv2D input: output: (None, 32, 32, 16) (None, 32, 32, 32)
activation_1 Activation input: output: (None, 32, 32, 32) (None, 32, 32, 32)
batch_normalization_1 BatchNormalization input: output: (None, 32, 32, 32) (None, 32, 32, 32)
max_pooling2d_1 MaxPooling2D input: output: (None, 32, 32, 32) (None, 16, 16, 32)
conv2d_2 Conv2D input: output: (None, 16, 16, 32) (None, 16, 16, 64)
activation_2 Activation input: output: (None, 16, 16, 64) (None, 16, 16, 64)
batch_normalization_2 BatchNormalization input: output: (None, 16, 16, 64) (None, 16, 16, 64)
max_pooling2d_2 MaxPooling2D input: output: (None, 16, 16, 64) (None, 8, 8, 64)
flatten Flatten input: output: (None, 8, 8, 64) (None, 4096)
dense_3 Dense input: output: (None, 4096) (None, 16)
activation_3 Activation input: output: (None, 16) (None, 16)
batch_normalization_3 BatchNormalization input: output: (None, 16) (None, 16)
dropout Dropout input: output: (None, 16) (None, 16)
dense_input InputLayer input: output: [(None, 8)] [(None, 8)]
dense_4 Dense input: output: (None, 16) (None, 4)
dense Dense input: output: (None, 8) (None, 8)
activation_4 Activation input: output: (None, 4) (None, 4)
dense_1 Dense input: output: (None, 8) (None, 4)
dense_5 Dense input: output: (None, 4) (None, 1)
dense_2 Dense input: output: (None, 4) (None, 1)
concatenate Concatenate input: output: [(None, 1), (None, 1)] (None, 2)
dense_6 Dense input: output: (None, 2) (None, 4)
dense_7 Dense input: output: (None, 4) (None, 1)

Figure 9: Neural Network Architecture

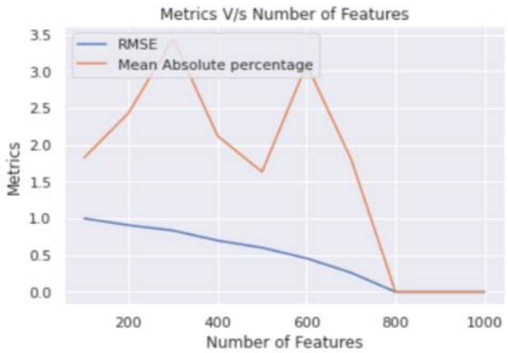


Figure 10: Metrics vs Number of features for linear regression

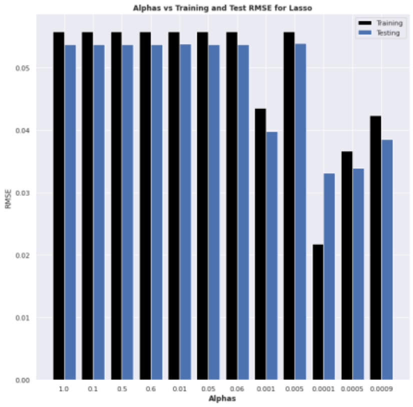


Figure 11: RMSE vs alpha plot for train-test sets for LASSO

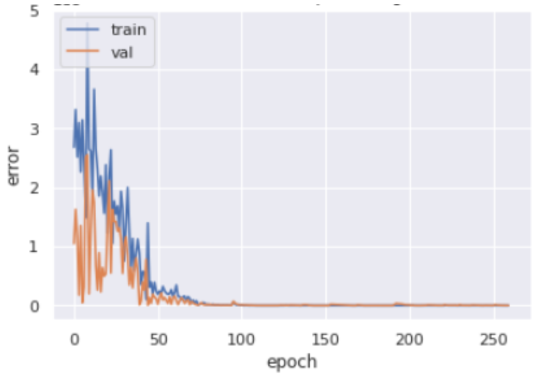


Figure 12: Mean Absolute Percentage Error vs epoch for Neural Network

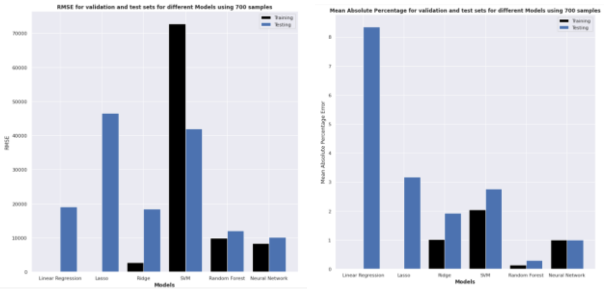


Figure 13: Metric comparison on different models for 700 samples

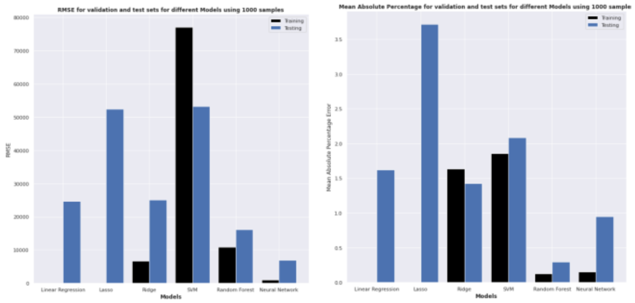


Figure 14: Metric comparison on different models for 1000 samples

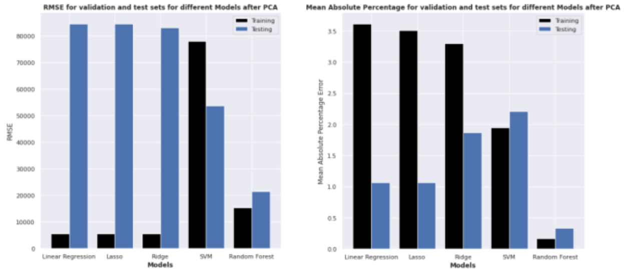


Figure 15: Metric comparison on different models after PCA

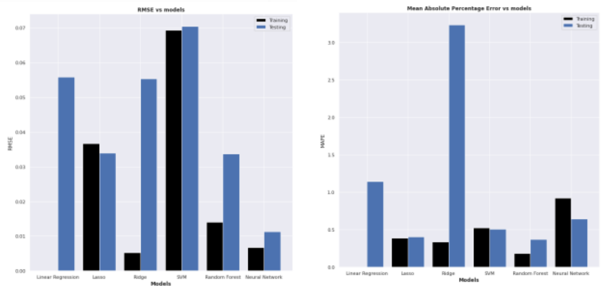


Figure 16: Bias and Variance Analysis

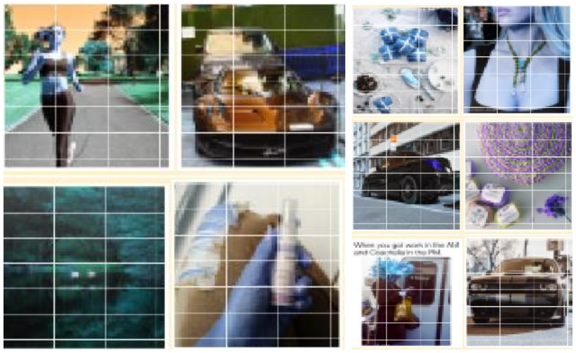


Figure 17: Images with high error

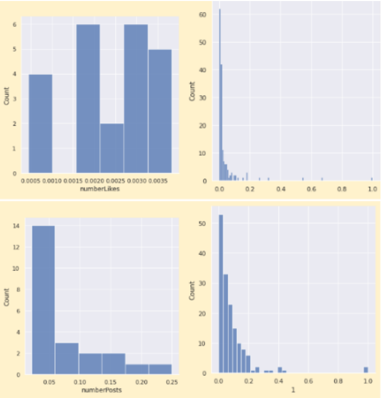


Figure 18: Plots to analyze the cases with high error

References

[1] Dugué, C., 2022. *Predicting the number of likes on Instagram*. [online] Medium. Available at: <https://towardsdatascience.com/predict-the-number-of-likes-on-instagram-a7ec5c020203> [Accessed 14 April 2022].

[2] Carta, S., Podda, A., Recupero, D., Saia, R. and Usai, G., 2020. Popularity Prediction of Instagram Posts. *Information*, 11(9), p.453.

[3] Priego, J., 2022. [online] Upcommons.upc.edu. Available at: <https://upcommons.upc.edu/bitstream/handle/2117/339937/152579.pdf?sequence=1&isAllowed=y> [Accessed 14 April 2022].

[4] Zamorano, G., 2022. *Predicting the Popularity of Instagram Posts*. [online] Medium. Available at: <https://towardsdatascience.com/predicting-the-popularity-of-instagram-posts-deeb7dc27a8f> [Accessed 14 April 2022].

[5] GitHub. 2022. *instagram-like-predictor/profiles at master · gvsi/instagram-like-predictor*. [online] Available at: <https://github.com/gvsi/instagram-like-predictor/tree/master/profiles> [Accessed 14 April 2022].

[6] Iconosquare - Instagram, TikTok, LinkedIn, Facebook and Twitter Analytics and Management Platform. 2022. *Iconosquare - Instagram, TikTok, LinkedIn, Facebook and Twitter Analytics and Management Platform*. [online] Available at: <https://influence.iconosquare.com/> [Accessed 14 April 2022].