Facebook Analysis Report

1. Introduction

Facebook is one of the largest social media platforms on the internet, where users can interact with each other and produce content. Content can come in the form of text posts, images, videos, or other forms of media.

As the popularity of social media continues to grow at an exponential rate, so does the need for businesses and influencers to take advantage of the various platforms available to grow their brand. A large amount of reach on social media yields a large fanbase, which in turn increases the brand influence. Brands and businesses can achieve this by finding ways to improve their reach through increasing engagements.

'Engagements' is the system used by the platform to gauge traction gained by posts. The total number of engagements is the total number of likes, comments, and shares. The more engagements a post gains, the more likely it is to be viewed by other accounts, and the larger the reach gained by the user that posted it.

The easiest and most effective way to gauge the popularity of a post is by looking at the number of 'likes' it receives. Therefore, a brand or business that finds a systematic method to improve the number of likes per post, will be able to increase its reach, popularity, and value much more than a brand that doesn't take advantage of the system.

Social media growth experts have theorized that there are multiple ways to increase the popularity of a page, by increasing the number of engagements its posts receive.
Some of the variables that can be used include: The rate at which a page produces content/posts, the type of the post, the time of the day at which the post is put out, and the number of comments. Evidently, we can theorize that a page/user with an already large following is more likely to gain traction on their posts compared to a smaller account.

Due to the sheer variety of variables available that can be manipulated to improve engagements, it should be possible for a business to formulate an algorithm that predicts the number of engagements or likes that a post receives. We will attempt to create such a model, as we believe that a prediction algorithm is an excellent way to increase the popularity of an account, and consequently, the reach of the brand/business that runs it.

The method used to produce this prediction model will be KNN-regression, as all of the variables suggested are quantitative. We will be able to use a mixture of pre-posting variables as well as variables determined after the post is produced, to predict the number of likes that it receives.

To produce this model, we will use a dataset obtained from the machine learning database https://archive.ics.uci.edu/ml/datasets/Facebook+metrics. The dataset contains a random selection of Facebook posts from a multitude of accounts. The latter vary in lifetime likes and visibility. Furthermore, the posts are a selection of images, status updates, videos, and links.

Variable descriptions.

1. Page total likes: The total amount of likes the Facebook page has.
2. Type: describes whether it is a photo, video, or status.
3. Category:
4. Post Month: The month in which the post was made (1-12).
5. Post Weekday: The day of the week in which the post was made (1-7).
6. Post Hour: The hour of the day in which the post was made (1-24).
7. Paid: Whether the post is monetized or not.
8. Lifetime Post Total Reach: refers to the total number of people who saw the post and clicked on it.
9. Lifetime Post Total Impressions: refers to the total number of times the post was displayed on the feed.
10. Lifetime Engaged Users: The number of unique users who clicked anywhere on the post.
11. Lifetime Post Consumers: The number of unique users who clicked anywhere on the post.
12. Lifetime Post Consumptions: Total number of clicks anywhere on the post
13. Lifetime Post Impressions by people who have liked your Page: refers to the total number of times the post was displayed on the feed of the people that have liked the page.
14. Lifetime Post reach by people who like your Page: refers to the total number of people who have liked the page who saw the post and clicked on it.
15. Lifetime People who have liked your Page and engaged with your post: The number of unique people who have liked the page and clicked anywhere on the post.
16. Comment: number of comments on the post.
17. Like: number of likes on the post.
18. Share: number of times the post has been shared.
19. Total Interactions: sum total of all likes, shares, comments, and reactions on the post.

The model relies on processing the information from several predictor variables. Collecting both retrospective and prospective data.

Classifications:

like (dependent variable/y-axis)

(chosen variables based on plots and correlation coefficients):

Independent variables/x-axis:

1 - prospective \ 5 - retrospective \ 3 - retrospective \ 9 - retrospective

10 - retrospective

where the retrospective variables contain historical information attributed to the accounts

Steps to use K-NN regression to conduct a data analysis for predictions:

1- Identify 2 variables to be used \ 2- Split the data into training and testing data \ 3- Create a scatterplot of the 2 variables \ 4- \ (a) identify and use n neighbouring points to the point of interest \ (b) take the mean value of the points to predict the number of likes a post from an account will get \ (c) Plot predicted value for likes from the KNN-regression model for several different values for K, see which one fits best for our training observations \ 5- Use ggpairs() to compile all scatter graphs in one matrix plot and colour it by type of post

Methods:

1 - load the following packages:
tidyverse, tidymodels, forcats, gridExtra, GGally

2 - read the data into R using the appropriate read_* function

We will be using read_csv2 as the data is using ‘,’ as a decimal and ‘.’ as a grouping mark.

```
facebook <- read_csv2("data/data_facebook.csv")
colnames(facebook) <- make.names(colnames(facebook))
```

3 - Clean up and wrangle the data:
remove the N/A values, and make sure that the data is “tidy”

```
facebook<- facebook %>%
na.omit()
dim(facebook)
```

4 - (summary stats)

Page total likes, type, Category, Post Month, Post Weekday, Post hour, Paid, 'Lifetime Post Total Reach', 'Lifetime Post Total Impressions', `Lifetime Engaged Users`, `Lifetime Post Consumers', `Lifetime Post Consumptions`, `Lifetime Post Impressions by people who have liked your Page`, `Lifetime Post reach by people who like your Page`, `Lifetime People who have liked your Page and engaged with your post`, comment, like, share, `Total Interactions`

5 - Remove binary columns (or columns that represent categorical variables)

```
subset(facebook, select= -c(...))
```

6 - The best variables (according to <source>) to use as predictors in the model are:

Post.Hour : The hour at which the post was made
Comment: the number of comments the post has received (which can gauge the number of replies/the number of replies made)
Type: Whether the post is a status update (text), picture, video, link

However, Post.Hour and Type are categorical, and cannot be used as predictors.

Additional variables that could be useful include:

Total.Interactions: The total amount of interactions a users page gets. Intuitively, this would be a good variable to use because more interactions should result in more likes.

7 - Find correlation coefficients for all the quantitative variables

8 - Create a table with the values of each correlation coefficient
9 - Arrange the table values in descending order

10 - Slice the top 5 rows, to get the predictors with the highest correlation coefficients. These variables will be our most likely predictors for the model.

11 - Split the original data into training and testing

```
split <- initial_split(facebook, prop= 0.7)
facebook_train <- training(split)
facebook_test <- testing(split)
```

12 - Subtract all the columns apart from the top 5, from the training data

```
subset(facebook_train, select= -c(...))
```

13 - Create visualisations of each of the top 5 variables compared to likes

Example:

```
Like_and_engaged_plot <- ggplot(facebook_train_scaled, aes(x =
Lifetime.People.who.have.liked.your.Page.and.engaged.with.your.post, y = like)) +
  geom_point(aes(colour=Type)) +
  labs(x = "People that liked your Page and engaged with post (standardized)", y = "Likes on post
(standardized)") +
  scale_x_log10(labels = scales::comma) +
  scale_y_log10(labels = scales::comma) +
 theme(text = element_text(size = 15))
Like_and_engaged_plot
```

14 - Choose variable that is best fit for KNN-regression

By looking at the graphs plotted above and the correlation coefficients, we chose total.interactions as our best predictor variable. This is because the variable has a high correlation coefficient and the graph is more linear than the others, meaning that it has a stronger relationship with the 'like' variable, which will result in more accurate predictions.

15 - Create the recipe to center and scale the training data, keep the testing data untouched

```
data_recipe <- recipe(yaxis ~ xaxis, data = data_train) %>%

step_scale(all_predictors()) %>%

step_center(all_predictors())
```

16 - Create the model specification, using the kknn engine, and regression mode. Make sure to tune the 'neighbours' argument in order to let R try multiple values of K

```
spec <- nearest_neighbor(weight_func = "rectangular", neighbors = tune()) %>%
  set_engine("kknn") %>%
  set_mode("regression")
```

17 - Perform the cross folding algorithm using 5 folds

```
sacr_vfold <- vfold_cv(sacramento_train, v = 5, strata = price)
```

18 - Create a workflow to complete our prediction mode;

```
sacr_wkflw <- workflow() %>%
  add_recipe(recipe) %>%
  add_model(spec)
```

19 - Create a tibble with a sequence equivalent to the number of nearest neighbours that we want to try (values of K). We will try K = 200

```
gridvals <- tibble(neighbors = seq(1, 200))
```

20 - Find the results of the mean RMSPE of the model:

```
sacr_results <- sacr_wkflw %>%
  tune_grid(resamples = sacr_vfold, grid = gridvals) %>%
  collect_metrics()
```

21 - Find the minimum RMSPE, then filter to find the minimum mean

```
data_min <- data_results %>%

filter(.metric == "rmse") %>%

filter(mean == min(mean))
```

23 - The ideal K value will be the value corresponding to the lowest rmse

24 - Retrain the model using the chosen value of K

```
set.seed(1234)
kmin <- sacr_min %>% pull(neighbors)
sacr_spec <- nearest_neighbor(weight_func = "rectangular", neighbors = kmin) %>%
  set_engine("kknn") %>%
  set_mode("regression")
```

```r
sacr_fit <- workflow() %>%
  add_recipe(sacr_recipe) %>%
  add_model(sacr_spec) %>%
  fit(data = sacramento_train)
```

25. Evaluate the model on the test set, if the RMSPE is not a lot higher than the one we calculated on the training data, our prediction model is good.

```r
sacr_summary <- sacr_fit %>%
  predict(sacramento_test) %>%
  bind_cols(sacramento_test) %>%
  metrics(truth = price, estimate = .pred)

sacr_summary

set.seed(1234)
sacr_preds <- sacr_fit %>%
  predict(sacramento_train) %>%
  bind_cols(sacramento_train)
```

Final code

1. Load relevant packages

```
library(tidyverse)
library(tidymodels)
library(GGally)

library(repr)
library(readxl)
library(DBI)
```

#Read the dataset_Facebook csv file into R

```
facebook <- read_csv2("dataset_Facebook.csv")
```

# Clean up, and remove the NA values to prevent errors when calculating summary statistics.

```
colnames(facebook) <- make.names(colnames(facebook))
facebook<- facebook %>%
na.omit()
dim(facebook)
```

2. View data to see what we are working with

```
Facebook
```

3. Correlation coefficients

We will create a correlation matrix for the facebook dataset, this will give us each variable's correlation coefficient. We will pick a variable that has a good coefficient (close to 1). The better the correlation, the stronger the relationship between that variable and 'like' which will fit better as our predictor for our K-NN regression model.

- **First we remove non-numerical variables such as 'type' and 'category' because the cor() function only takes in data with numerical variables.**

- We use the 'pearson' correlation coefficient, this is the default method that computes the linear dependency between two variables.

```
cor_facebook <- subset(facebook, select= -c(Type, Category, Post.Month, Post.Weekday,
Post.Hour, Paid)) %>%
            cor(use = "all.obs", method = c("pearson"))
```

- From the correlation matrix, extract rows 1-13 because they are the variables we would like to calculate the correlation coefficients for
- Extract column 11-12 because it includes the variable we are predicting (like).
- Create a tibble with rownames of possible predictors as "variable", this way we have a neat and organized table of correlation coefficients
- Taking columns 11-12 from the matrix has 2 columns, one of which we don't want, so we select the 2 we WANT to look at (variable and like)
- Arrange in descending order of correlation coefficient, this way we will have the best variables at the top

```
cor_facebook[1:13, 11:12] %>%
as_tibble(rownames="variable") %>%
select(variable, like) %>%
arrange(desc(like))
```

4. TOP 5 predictors

These are the best predictors because their correlation coefficients are near 1. Meaning, that they have the strongest relationship with ‘like’, the variable that we are predicting.

#1. Total.Interactions
#2. share
#3. comment
#4. Lifetime.Post.reach.by.people.who.like.your.Page
#5. Lifetime.Engaged.Users

5. Training and Testing split

We split the data up into 70% training and 30% testing. We build the classifier using only the training set and evaluate the accuracy of the classifier using the testing data. By following this practice we get a good idea of the classifier’s accuracy.
We chose a ratio of 70 to 30 as this is the perfect balance that allows that model to not be over reliant on the training data, while preserving a higher level of accuracy when evaluating the model using the testing set.

```
set.seed(1234)
split <- initial_split(facebook, prop= 0.7)
facebook_train <- training(split)
facebook_test <- testing(split)
```

The following columns have been removed as they are discrete or categorical variables. We cannot perform K-NN regression with either.

```
facebook_train<- subset(facebook_train, select= -c(Category,Post.Month, Post.Weekday,
Post.Hour, Paid))
facebook_train
```

6. Training set with just the top 5 predictors

We remove all the columns beside the top 5 predictors which had the highest Pearson correlation coefficients.

```
subset(facebook_train, select=
-c(Lifetime.People.who.have.liked.your.Page.and.engaged.with.your.post,
                                  Lifetime.Post.Impressions.by.people.who.have.liked.your.Page,
                                  Lifetime.Post.Consumptions,
                                 Lifetime.Post.Consumers,
                                 Lifetime.Post.Total.Impressions,
                                 Lifetime.Post.Total.Reach,
                                 Page.total.likes))
```

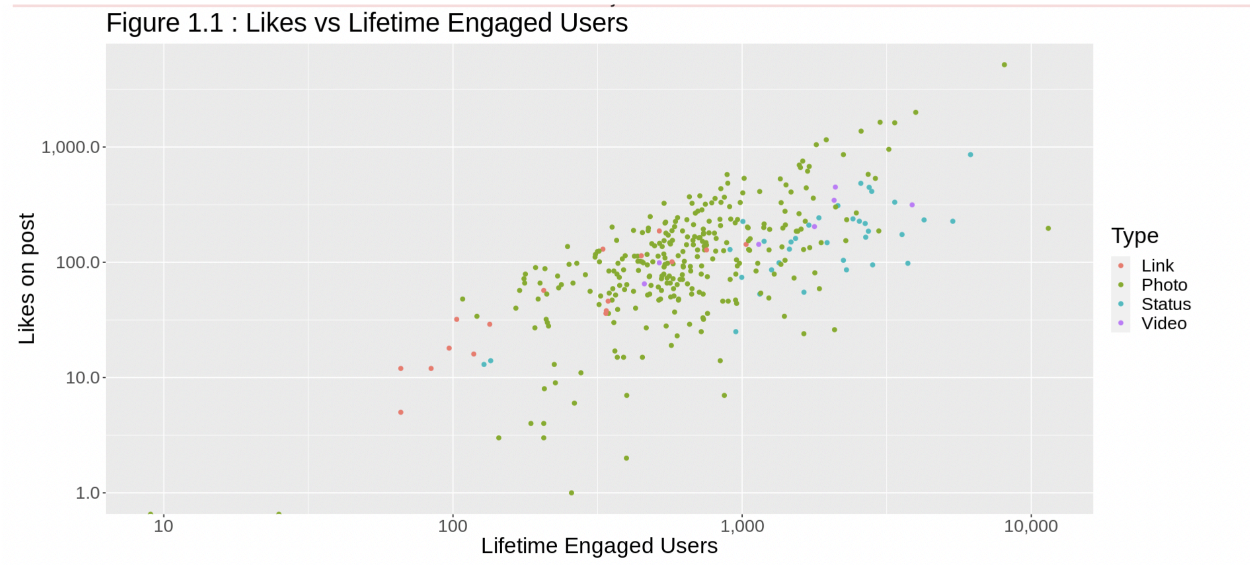
7. Extra visual - Using ggpairs to create a compact visualization of all variables
This helps us view all predictor variables at once.

```
plot_pairs <- facebook_train %>%
  select(like, comment, share, Total.Interactions, Lifetime.Engaged.Users,
Lifetime.Post.reach.by.people.who.like.your.Page) %>%
  ggpairs()
plot_pairs
```

8. Plots to visualize the top 5 predictors vs. likes

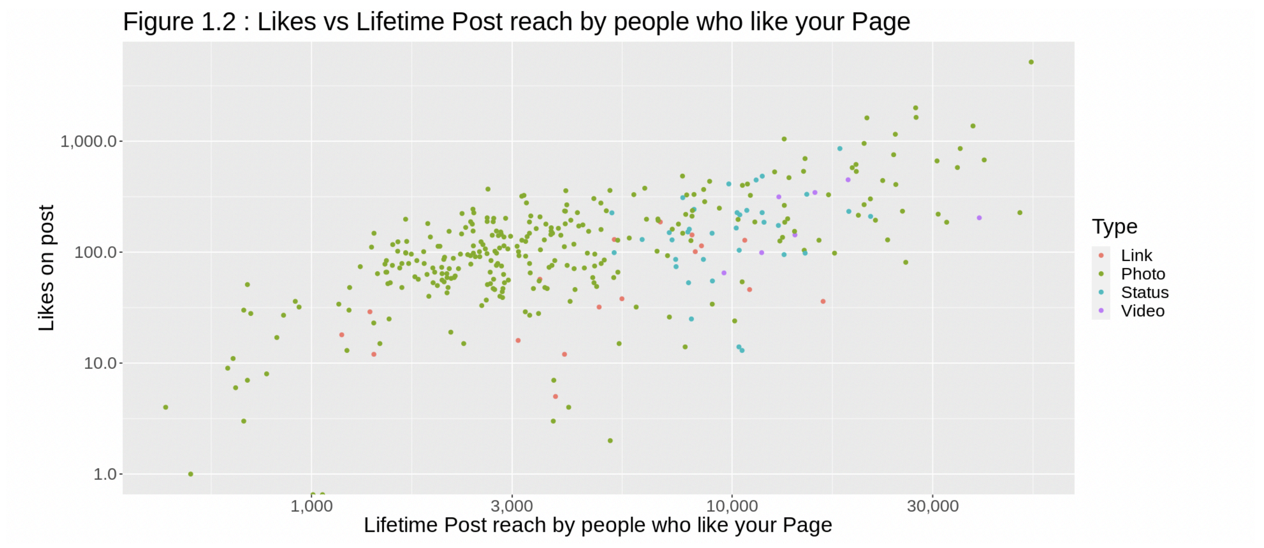
The plots show the relationship between the independent variables: Lifetime Engaged Users, Lifetime Post reach by people who like your Page, Comments on post, Amount of shares on a post, and Total Interactions and the dependent variable: likes on post. The data points are colored based on the type of post it is. The x-axis and y-axis have been scaled in order to provide a better visualization.

```
Lifetime.Engaged.Users_plot <- ggplot(facebook_train, aes(x = Lifetime.Engaged.Users, y
= like)) +
    geom_point(aes(colour=Type)) +
    labs(x = "Lifetime Engaged Users", y = "Likes on post") +
    scale_x_log10(labels = scales::comma) +
    scale_y_log10(labels = scales::comma) +
   theme(text = element_text(size = 15))+
   ggtitle("Figure 1.1 : Likes vs Lifetime Engaged Users")
Lifetime.Engaged.Users_plot
```



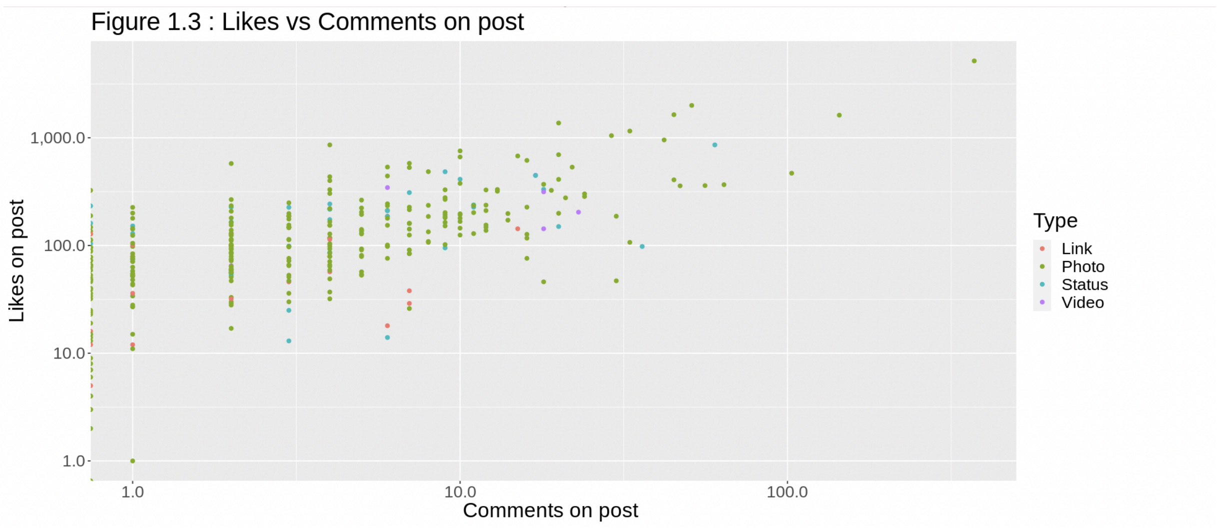
There seems to be a medium to slightly strong relationship between lifetime engaged users and the likes on the post. We can also see that on average, the post types that garner the most likes are videos and status updates. Links receive lower likes than their counterparts, while photos show a large variation.

```
Lifetime.Post.reach.by.people.who.like.your.Page_plot <- ggplot(facebook_train, aes(x =
Lifetime.Post.reach.by.people.who.like.your.Page, y = like)) +
    geom_point(aes(colour=Type)) +
    labs(x = "Lifetime Post reach by people who like your Page", y = "Likes on post") +
    scale_x_log10(labels = scales::comma) +
    scale_y_log10(labels = scales::comma) +
   theme(text = element_text(size = 15))+
ggtitle("Figure 1.2 : Likes vs Lifetime Post reach by people who like your Page ")
Lifetime.Post.reach.by.people.who.like.your.Page_plot
```



There seems to be a medium to slightly strong, positive relationship between lifetime post reach by people who like your page and the likes on the post. The points seem to be stuck together towards the centre of the graph, and show more variation in likes towards both ends of the x-axis.

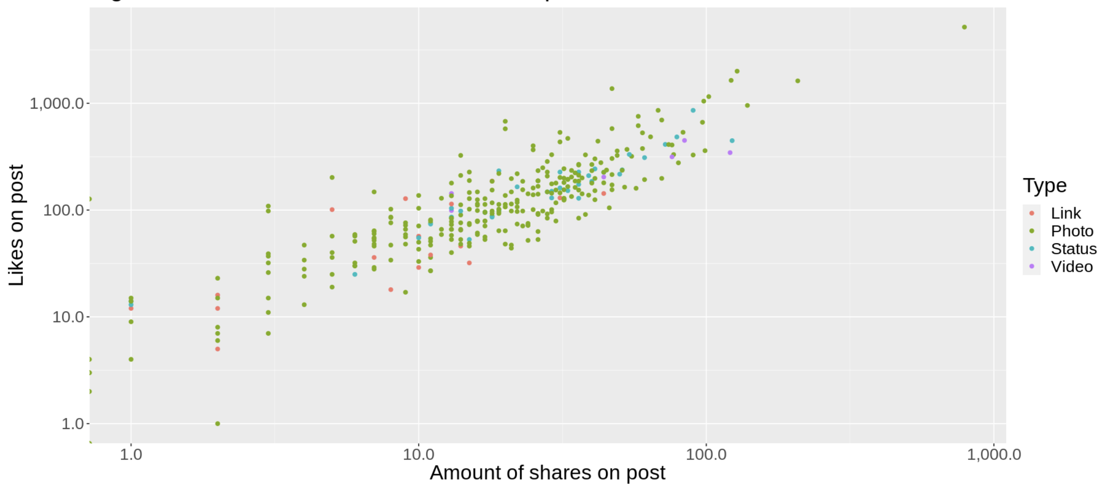
```
comment_plot <- ggplot(facebook_train, aes(x = comment, y = like)) +
    geom_point(aes(colour=Type)) +
    labs(x = "Comments on post ", y = "Likes on post") +
    scale_x_log10(labels = scales::comma) +
    scale_y_log10(labels = scales::comma) +
   theme(text = element_text(size = 15))+
ggtitle("Figure 1.3 : Likes vs Comments on post")
comment_plot
```



There seems to be a strong positive relationship between comments and the likes on the post. The points are closer together in the beginning part of the graph that displays a lower number of comments. Which reflects the reality that posts rarely receive over 10 comments unless the account that posted it is an extremely large account. However, it seems to be possible to receive a large number of likes on a post with very few comments.

```
share_plot <- ggplot(facebook_train, aes(x = share, y = like)) +
   geom_point(aes(colour=Type)) +
   labs(x = "Amount of shares on post", y = "Likes on post") +
   scale_x_log10(labels = scales::comma) +
   scale_y_log10(labels = scales::comma) +
  theme(text = element_text(size = 15))+
ggtitle("Figure 1.4 : Likes vs Amount of shares on post")
Share_plot
```

Figure 1.4 : Likes vs Amount of shares on post

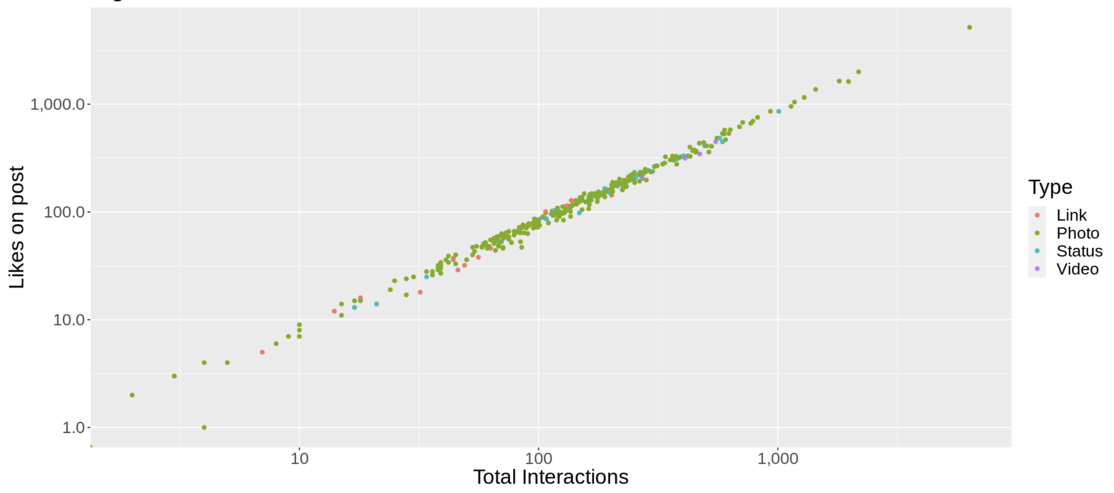


There seems to be a very strong positive relationship between shares and the likes on the post. The points are close together for the majority of the graph, only showing large variation towards the lower end (where the number of shares is extremely low), and only a few outliers towards the higher end (where the number of shares is extremely high). This reflects the idea that the more shares a post receives, and the more feeds it reaches, the more likely it is to gain a higher number of likes as it is rare for a "bad" post to receive a lot of shares. The upper end of the graph (the outliers) can be explained as the accounts producing the posts could be celebrity accounts, which rely more on the following than the individual shares to receive likes.

```
Total.Interactions_plot <- ggplot(facebook_train, aes(x = Total.Interactions, y = like)) +
    geom_point(aes(colour=Type)) +
    labs(x = "Total Interactions", y = "Likes on post") +
    scale_x_log10(labels = scales::comma) +
    scale_y_log10(labels = scales::comma) +
   theme(text = element_text(size = 15))+
ggtitle("Figure 1.5 : Likes vs Total Interactions")
```

Total.Interactions_plot

Figure 1.5 : Likes vs Total Interactions



There seems to be a very strong positive relationship between interactions and the likes on the post. The points are close together and show almost no outliers. This predictor seems to be "too" good. This can be explained as the number of interactions a post receives is the sum of likes, shares, comments and reactions. Since the likes are included in the algorithm for interactions, then the points will only be shifted from their number of likes depending on the values of the aforementioned variables that make it up. Therefore, it seems like it would not be a useful predictor for the model, as it doesn't offer any tangible ways to increase the likes. Increasing interactions cannot be done manually or artificially, so it would be a useless predictor to base the model on.

For each of the plots above, we can see that on average, the post types that garner the most likes are videos and status updates. Links receive lower likes than their counterparts, While photos show a large variation.

9. Preprocessing data and Performing Cross Validation

#creating a recipe with the predictor 'share' and the variable we are predicting 'like'
#centering and scaling the data in recipe
#it is important to scale the data because K-NN regression uses euclidean distance to compute predictions, scaling is also just good practice. Furthermore, there is a large

deviation in the means of the variables, so it is important to standardise the data for a more accurate analysis.

```
facebook_recipe <- recipe(like ~ share, data = facebook_train) %>%
step_scale(all_predictors()) %>%
step_center(all_predictors())
```

#creating the specification for the KNN regression model

We use tuning in order to split the data into multiple ways, then train and evaluate the classifier for each split. This will allow us to narrow down the value of K that produces the most accurate model.

```
facebook_spec <- nearest_neighbor(weight_func = "rectangular", neighbors = tune())
%>%
  set_engine("kknn") %>%
  set_mode("regression") # indicating a regression problem
```

Cross validation is used here as a compliment to the tuning process. We will split our data into multiple folds (in this case 5, as it is the optimal number of folds that keeps the process reasonably fast).

There is an important detail to mention about the process of tuning: we can, if we want to, split our overall training data up in multiple different ways, train and evaluate a classifier for each split, and then choose the parameter based on *all* of the different results. If we just split our overall training data *once*, our best parameter choice will depend strongly on whatever data was lucky enough to end up in the validation set. Perhaps using multiple different train / validation splits, we'll get a better estimate of accuracy, which will lead to a better choice of the number of neighbours

K

K

for the overall set of training data.

Note: you might be wondering why we can't we use the multiple splits to test our final classifier after tuning is done. This is simply because at the end of the day, we will produce a single

classifier using our overall training data. If we do multiple train / test splits, we will end up with multiple classifiers, each with their own accuracy evaluated on different test data.

Let's investigate this idea in R! In particular, we will use different seed values in the `set.seed` function to generate five different train / validation splits of our overall training data, train five different K-nearest neighbour models, and evaluate their accuracy.

```
#perform cross validation by using 5 folds
facebook_vfold <- vfold_cv(facebook_train, v = 5, strata = like)

#finalizing the system using a workflow
facebook_wkflw <- workflow() %>%
  add_recipe(facebook_recipe) %>%
  add_model(facebook_spec)
```

10. Find the K value corresponding to lowest RMSPE

```
# creating a tibble with gridvalues from 1 to 200, to be used in the tuning process of K
gridvals <- tibble(neighbors = seq(1, 200))


# Finding the accuracy of the model based on all values of K
facebook_results <- facebook_wkflw %>%
  tune_grid(resamples = facebook_vfold, grid = gridvals) %>%
  collect_metrics()
```

The value of K with the lowest rmse will provide us with the most accurate prediction model, as it will indicate that the difference between the observed variable value and the predicted variable value is as small as possible.

```
# Finding the K value with the minimum rmse
facebook_min <- facebook_results %>%
filter(.metric == "rmse") %>%
filter(mean == min(mean))

facebook_min
```

11. Retraining the model using the appropriate value of k (k = 4)

```
set.seed(1234)

kmin <- facebook_min %>% pull(neighbors)  #extract smallest K
facebook_spec <- nearest_neighbor(weight_func = "rectangular", neighbors = kmin) %>%
  set_engine("kknn") %>%
  set_mode("regression")

facebook_fit <- workflow() %>%
  add_recipe(facebook_recipe) %>%
  add_model(facebook_spec) %>%
  fit(data = facebook_train)
```

We evaluate the finalised model using the best value of K, in order to test its accuracy when predicting the testing data. This is the data analysis step.

12. Evaluating the model on the test data

To determine whether or not our K-NN regression model does well on predicting data it has not seen before.

```
facebook_summary <- facebook_fit %>%
  predict(facebook_test) %>%
  bind_cols(facebook_test) %>%
  metrics(truth = like, estimate = .pred)

facebook_summary

set.seed(1234)
facebook_preds <- facebook_fit %>%
  predict(facebook_train) %>%
  bind_cols(facebook_train)
```

Since the RMSPE from our testing set is not worse than the cross validation RMSPE, our model is doing a good job predicting on data it has not seen before.

13. K-NN model predictions visualization

This plot will visualize our predictions for likes for all possible numbers of shares. We can see that this plot neither overfits nor under fits the data, therefore our choice of K was made well.

```
facebook_plot_final <- ggplot(facebook_preds, aes(x = share, y = like)) +
  geom_point(alpha = 0.4) +
  xlab("Shares on post") +
  ylab("Likes on post") +
  scale_x_log10(labels = scales::comma) +
  scale_y_log10(labels = scales::comma) +
  geom_line(data = facebook_preds, aes(x = share, y = .pred), color = "blue") +
  ggtitle(paste("K = ", kmin)) +
  theme(text = element_text(size = 20))

facebook_plot_final
```

refers to the total number of people who have liked the page who saw the post and clicked on it.

Discussion

Discuss what impact could such findings have?

- Increase brand image through social media
- Influencers and businesses can aim for higher shares by creating posts that encourage individuals to share, which in turn can help them gain more likes and grow their brand.
- We also found out that there is a positive relationship between comments and Lifetime Post reach by people who like your Page with likes. So this could enable social media influencers/businesses to design their posts in a way that helps them get more comments. Moreover, it is important for these pages to make more people like their page itself in order for them to gain higher likes on their posts. To do that, they could focus on creating content that allows them to build strong and long-term relationships with their consumer/fan base.

Discuss what future questions could this lead to?

Would there also be a strong positive correlation between reactions on a Facebook post and the number of shares it has?
Would the relationships that we saw in this project for Facebook be similar to the ones in other social media platforms (such as Instagram, Twitter, etc.)?

Sérgio, M. Paulo, R. & Bernardo, V. (2016). Predicting social media performance metrics and evaluation of the impact on brand building: A data mining approach. Science Direct, 69(9), 3341-3351.
https://www.sciencedirect.com/science/article/abs/pii/S0148296316000813?via%3Dihub

Alfred, L. (N/A). Post Less, Boost Top Posts, and More: 14 Ways to Increase Your Facebook Page Engagement, Buffer.
https://buffer.com/library/increase-facebook-page-engagement/

Christina, N. (2018). 17 Simple Ways to Increase Facebook Engagement (Free Calculator), Hootsuite. https://blog.hootsuite.com/increase-facebook-engagement/

Sérgio, M. Paulo, R. & Bernardo, V. (2016). Facebook Metrics Data Set. Machine Learning Repository. https://archive.ics.uci.edu/ml/datasets/Facebook+metrics

Facebook is one of the largest social media platforms on the internet, where users can interact with each other and produce content. Content can come in the form of text posts, images, videos, or other forms of media.

As Social Media continues to grow exponentially, so does the need for businesses and influencers to take advantage of it to grow their brand. A large reach on social media yields a large fanbase, which in turn increases the brand influence. Brands and businesses can achieve this by finding ways to improve their reach through increasing engagements.

The best way to gauge the popularity of a post is by looking at the number of 'likes' it receives. A brand or business that finds a systematic method to improve the number of likes per post, will be able to increase its reach, popularity, and value much more than a brand that doesn't take advantage of the system.

Social media experts have identified a multitude of variables that can be manipulated to improve engagements. Therefore, it should be possible for a business to formulate an algorithm that predicts the number of engagements or likes that a post receives based on these variables. We will attempt to create such a model, as we believe that a prediction algorithm is an excellent way to increase the popularity of an account, and consequently, the reach of the brand/business that runs it.

Would a KNN regression model be able to accurately predict the number of likes of a given facebook post based on any of shares, lifetime engagements, lifetime reach, comments, and interactions?

The prediction model will use KNN-regression, as all of the variables suggested are quantitative. We will use a mixture of pre-posting variables as well as variables determined after the post is produced, to predict the number of likes that it receives.

To produce this model, we will use a dataset obtained from the machine learning database https://archive.ics.uci.edu/ml/datasets/Facebook+metrics. The dataset contains a random selection of Facebook posts from a multitude of accounts. The latter vary in lifetime likes and visibility. Furthermore, the posts are a selection of images, status updates, videos, and links.

Read the file into R. Clean up the data by removing the NA values.
The function used to read the file is csv2, as the data is using ‘;' as the delimiter.

Create a correlation matrix for the Facebook dataset, this will give us each variable's correlation coefficient.

First, we remove non-numerical variables such as 'type' and 'category' because the cor() function only takes in data with numerical variables.

We use the 'pearson' correlation coefficient, this is the default method that computes the linear dependency between two variables.

From the correlation matrix, extract rows 1-13 because they are the variables we would like to calculate the correlation coefficients for.

Next, we extract the columns 11-12 because they include the variable we are predicting (likes).

Create a tibble with row names of possible predictors as "variable", this way we have a neat and organized table of correlation coefficients.

Next, we take columns 11-12 from the matrix. The matrix contains a column we don't want, so we select the 2 we WANT to look at (variable and like).

Finally, arrange the data in descending order.

#### Top 5 predictors are:
1- Total.Interactions

2 - share

3 - comment

4 - Lifetime.Post.reach.by.people.who.like.your.Page

5 - Lifetime.Engaged.Users

These are the best predictors because their correlation coefficients are near 1. Meaning, that they have the strongest relationship with ‘like’, the variable that we are predicting.

We split the data up into 70% training and 30% testing. We build the classifier using only the training set and evaluate the accuracy of the classifier using the testing data. This practice will indicate the classifier’s accuracy.
We chose this ratio because it is the perfect balance that allows the model to not be over-reliant on the training data, while preserving a higher level of accuracy when evaluating the model using the testing set.

Since our model is a KNN regression model, it is best to remove the columns with categorical variables, binary variables and discrete variables with a small finite range. This is because they will be undesirable predictors for a quantitative variable using a regression model, as the line of best fit will be impossible to make or of poor quality. The following variables fit this criteria: Category, Post.Month, Post.Weekday, Post.Hour, Paid.

The only categorical variable that will not be removed is "Type", as according to our sources, the type of the post can have an impact on how many likes it receives. We will test this theory by colour coding our visualizations based on Type, and therefore it is important to keep the variable.

Here, we will use ggpairs to view visualizations, as well as correlation coefficients of all predictor variables in one place.

We can see that there is noticeable multicollinearity between total interactions, shares and likes. This was not noticeable in the
initial analysis involving the correlation coefficients. This is because we only examined the relationships of likes and the remaining variables, but did not examine the relationships within those variables. We predict that it might be necessary later to discount one of the collinear variables.

There seems to be a positive, medium to slightly strong relationship between lifetime engaged users and the likes on the post.
The data points are close together towards the centre of the graph, with outliers equally likely towards the highest and lowest values of lifetime engaged users.

There is a medium to slightly strong, positive relationship between the variables above. The points seem to be stuck together towards the centre of the graph, showing more variation in likes towards both ends of the x-axis.

There is a strong positive relationship between the variables above. The points are closer together in the beginning part of the graph that displays a lower number of comments. We hypothesize that this reflects the reality that posts rarely receive over 10 comments unless the account that posted it is extremely popular. However, it seems to be possible to receive a large number of likes on a post with very few comments.

There is a very strong positive relationship between the variables above. The points are close together for the majority of the graph, only showing large variation towards the lower end (where the number of shares is extremely low), and only a few outliers towards the higher end (where the number of shares is extremely high). We believe that this reflects the idea that the more shares a post receives, and the more feeds it reaches, the more likely it is to gain a higher number of likes as it is rare for a “bad” post to receive a lot of shares. We hypothesize that the upper end of the graph (the outliers) can be explained as the accounts producing the posts could be celebrity accounts, which rely more on the following than the individual shares to receive likes.

There is a very strong positive relationship between the variables above. The points are close together and show almost no outliers. This predictor seems to be “too” good. Since the likes are included in the algorithm for interactions, then the points will only be shifted from their number of likes depending on the values of the aforementioned variables that make it up. Therefore, it seems like it would not be a useful predictor for the model, as it doesn’t offer any tangible ways to increase the likes. Increasing interactions cannot be done manually or artificially, so it would be a useless predictor to base the model on.

For the plots above, we can see that on average, the post types that garner the most likes are videos and status updates. Links receive lower likes than their counterparts, While photos show a large variation.

We decided to use "shares" as our predictor variable, as it has the second-highest correlation coefficient. It is the variable with the strongest relationship with "likes", excluding "Total Interactions".

We scaled the data because K-NN regression uses euclidean distance to compute predictions. Furthermore, there is a large deviation in the means of the variables, so it is important to standardize the data for a more accurate analysis.

We use tuning to split the data into multiple ways, then train and evaluate the classifier for each split. This will allow us to narrow down the value of K that produces the most accurate model.

Cross-validation is used here as a complement to the tuning process. We will split our training data into multiple folds (in this case 5, as it is the optimal number of folds that keeps the process reasonably fast).

Next, we create a tibble with grid values that will vary depending on the number of K values we would like to test for the tuning process. In this project, we will test from K = 1 up to K = 200.

Finally, we evaluate the accuracy of the model for every value of K. We do this to identify the ideal value of K for the prediction model.
The value of K with the lowest rmse will provide us with the most accurate prediction model, as it will indicate that the difference between the observed variable value and the predicted variable value is as small as possible.

As we can see, the ideal K to use for our KNN model is K = 4. This is because it has the lowest rmse out of all the values of K. Therefore, we will use it to retrain our model.

Here, we evaluate the model using the test data. Evaluating the accuracy of our model will let us know how effective it will be when predicting the likes of any new post from a new dataset, or any data it has not seen before.

Since the RMSPE from our testing set is not worse than the cross-validation RMSPE, our model is doing a good job predicting on data it has not seen.

This plot will visualize our predictions for likes for all possible numbers of shares. We can see that this plot neither overfits nor under fits the data, therefore our choice of K was made well.

## Discussion

### Summarize what you found:

It is possible to predict facebook likes using a KNN regression model. The best modifiable variable to do that with is the number of shares. We found that comments can also be used as a predictor, albeit less effectively as shares. We received a significantly lower rmse when evaluating our model on the testing set than on the training set. This shows that our model was remarkably accurate (somewhat unexpectedly) at predicting the likes for posts it has not seen.

### Discuss whether this is what you expected to find?

In theory, it should've been possible to predict the likes using shares. Observable when using various Social Media platforms, it tends to be very common for large accounts with large shares per post to receive plenty of likes on them. According to our sources, comments, shares and type of post are all strongly linked to the number of likes a post will receive. Therefore, our findings matched this thesis as the correlation coefficients for both comments and shares were very high. Additionally, the types of posts were indicative of a pattern where the videos and status updates seemed to attract the most likes. "Video" type posts were recommended by Social Media growth experts as the best kind of posts to attract likes, which was reinforced by our findings.

### Discuss what impact could such findings have?

By manipulating the aforementioned variables (shares, comments etc.), a brand can increase its social media presence, and in turn, improve its image.

Influencers and businesses can aim for higher shares by curating more "shareable" posts, which in turn can help them gain more likes and grow their following.

We observed a positive relationship between comments and Lifetime Post reach by people who like the user's page, with likes. This could enable social media influencers/businesses to design their posts in a way that helps them get more comments. Such as, asking a question, or creating a discussion or debate around a topic.

Moreover, these pages need to make more people like their page itself for them to gain higher likes on their posts. To do that, they could focus on creating content that allows them to build strong and long-term relationships with their consumers/fan base.

Therefore, it seems likely that a "recipe" or method can be developed to attract higher likes on social media posts, one that exploits the factors mentioned in our report (and by the following sources: [][]).

### Discuss what future questions could this lead to?

Would there also be a strong positive correlation between reactions (a new mechanism in Facebook, not included in our dataset) on a Facebook post and the number of shares it has?

Would the relationships we saw in this project for Facebook be similar to those in other social media platforms (such as Instagram, Twitter, etc.)?

We were not able to use "Hour at which the post was created" as one of our predictors, even though it was mentioned in the source [] as a relevant contributor to Facebook likes. The limitation here was that we could not work with a discrete variable that had such a small finite range. Perhaps, we could revisit this prediction model with the Post.hour variable and find a way to incorporate it.