Data Preprocessing

Firstly, we check for NULL values in the dataset. The result is shown in Figure 1.

```
[ ]  df = pd.read_csv("/content/drive/My Drive/Colab Notebooks/Facebook_metrics/dataset_Facebook.csv",delimiter=';')
     df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 19 columns):
Page total likes                                                  500 non-null int64
Type                                                              500 non-null object
Category                                                          500 non-null int64
Post Month                                                        500 non-null int64
Post Weekday                                                      500 non-null int64
Post Hour                                                         500 non-null int64
Paid                                                              499 non-null float64
Lifetime Post Total Reach                                         500 non-null int64
Lifetime Post Total Impressions                                   500 non-null int64
Lifetime Engaged Users                                            500 non-null int64
Lifetime Post Consumers                                           500 non-null int64
Lifetime Post Consumptions                                        500 non-null int64
Lifetime Post Impressions by people who have liked your Page      500 non-null int64
Lifetime Post reach by people who like your Page                  500 non-null int64
Lifetime People who have liked your Page and engaged with your post  500 non-null int64
comment                                                           500 non-null int64
like                                                              499 non-null float64
share                                                             496 non-null float64
Total Interactions                                                500 non-null int64
dtypes: float64(3), int64(15), object(1)
memory usage: 74.3+ KB
```

Figure 1. Dataset Information

The result shows that there are missing value in `Paid`, `like`, and `share`. We manage to fill all missing values by 0 and rename some attributes for briefly referring as shown in Figure 2.

```
[ ]  df['like'].fillna(0,inplace=True)
     df['share'].fillna(0,inplace=True)
     df['Paid'].fillna(0,inplace=True)
```

```
[ ]  df.rename(columns={
         'Lifetime Post Total Reach':'LPTR',
         'Lifetime Post Total Impressions':'LPTI',
         'Lifetime Engaged Users' : 'LEU',
         'Lifetime Post Consumers' : 'LPCM',
         'Lifetime Post Consumptions' : 'LPCP',
         'Lifetime Post Impressions by people who have liked your Page' : 'LPI',
         'Lifetime Post reach by people who like your Page' : 'LPR',
         'Lifetime People who have liked your Page and engaged with your post' : 'LPW'
          },
               inplace=True)
```

Figure 2. Fill all Missing Values

Data Visualization

We plot the correlation matrix to see the correlation between each attribute. The output correlation matrix is shown in Figure 3.

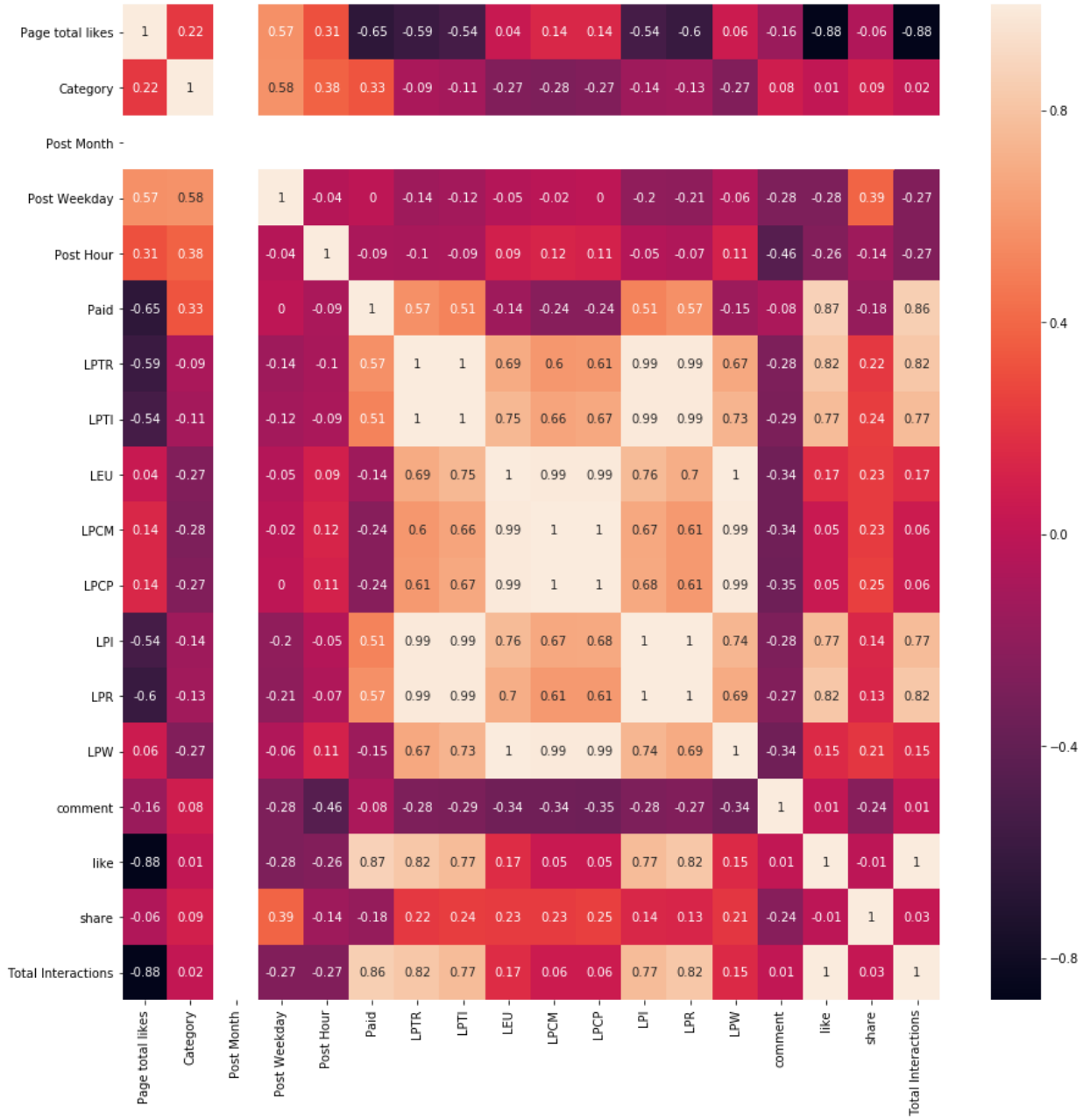

Figure 3. Correlation Matrix

We found that Lifetime Post Consumer (LPCM) has strong positive correlation between three variables, namely Lifetime Post Consumptions (LPCP), Lifetime Engaged Users (LEU), and Lifetime People who have liked your Page and engaged with your post (LPW).

Pair plotting is used to visualize the relationship between all three variables. Figure 4 illustrates the pair plotting result.

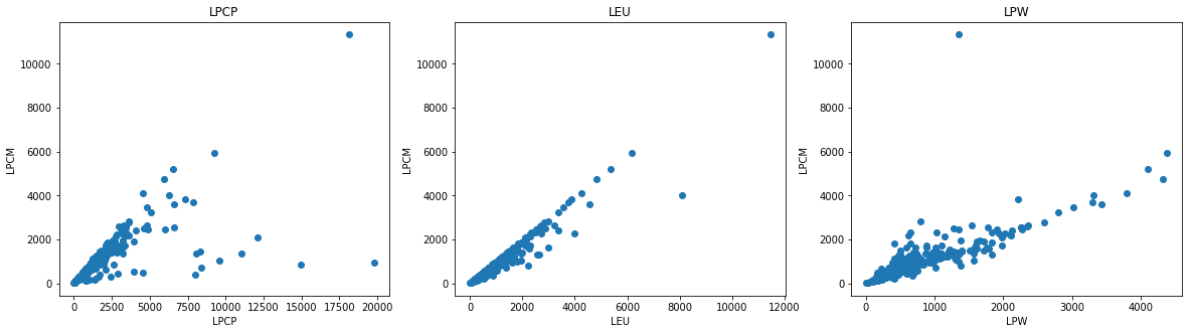

Figure 4. Scatter Pair Plotting

Scatter plots of these three variables show that LPCP have positive correlation to LPCM, LEU, and LPW.

Preparing Data

We concatenate the LPCP, LEU and LPW columns. Then, the data is split into training and test sets as shown in Figure 5.

```
[ ]  X = pd.DataFrame(np.c_[df['LPCP'], df['LEU'], df['LPW']], columns = ['LPCP','LEU','LPW'])
     Y = df['LPCM']
```

```
[ ]  from sklearn.model_selection import train_test_split

     X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2, random_state=5)
     print(X_train.shape)
     print(X_test.shape)
     print(Y_train.shape)
     print(Y_test.shape)
```

Figure 5. Preparing Data

Create Model and Training

Linear Regression and 10-fold cross validation are applied to train the data as shown in Figure 6.

```
[ ]  from sklearn.linear_model import LinearRegression
     from sklearn.metrics import mean_squared_error,mean_absolute_error

     from sklearn.model_selection import KFold

     r2scores = []
     rmse = []
     mae = []
     lin_model = LinearRegression()

     XX = np.array(X_train)
     YY = np.array(Y_train)

     cv = KFold(n_splits=10, random_state=42, shuffle=False)

     for train_index, test_index in cv.split(XX):
         XX_train, XX_test, yy_train, yy_test = XX[train_index], XX[test_index], YY[train_index], YY[test_index]
         lin_model.fit(XX_train, yy_train)
         y_train_predict = lin_model.predict(XX_test)
         rmse.append(np.sqrt(mean_squared_error(yy_test, y_train_predict)))
         mae.append(mean_absolute_error(yy_test, y_train_predict))
         r2scores.append(lin_model.score(XX_test, yy_test))
```

Figure 6. Create Model and Training

Evaluate using RMSE, MAE, and R2

Figure 7 shows the code and the evaluating result.

```
y_test_predict = lin_model.predict(X_test)
rmse = (np.sqrt(mean_squared_error(Y_test, y_test_predict)))
mae = mean_absolute_error(Y_test, y_test_predict)
r2 = lin_model.score(X_test, y_test_predict)

print("The model performance for testing set")
print("--------------------------------------")
print('RMSE is {}'.format(rmse))
print('MAE is {}'.format(mae))
print('R2 score is {}'.format(r2))
```

```
The model performance for testing set
--------------------------------------
RMSE is 179.31351887616682
MAE is 101.86023631014952
R2 score is 1.0
```

Figure 7. The Performance for Testing Set

Our model gets RMSE 179.31, MAE 101.86, and R^2 score 1.0. The scatter plot between the actual value and predicted value is shown in Figure 8.

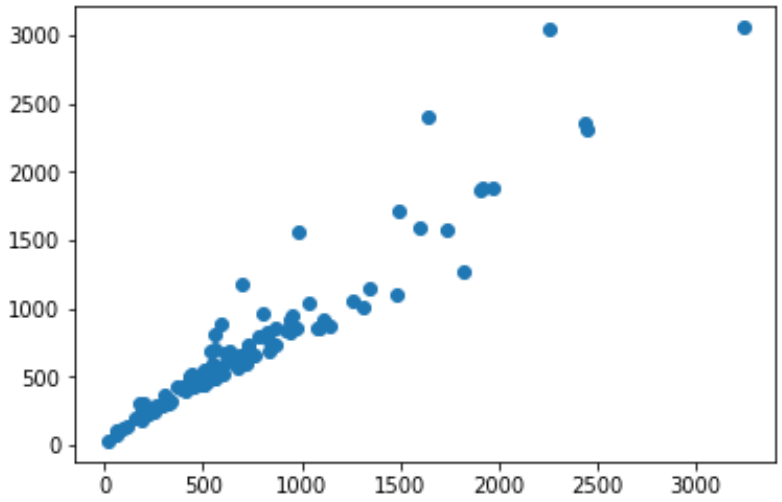

Figure 8. Scatter Plot between Predicted Values and Actual Values