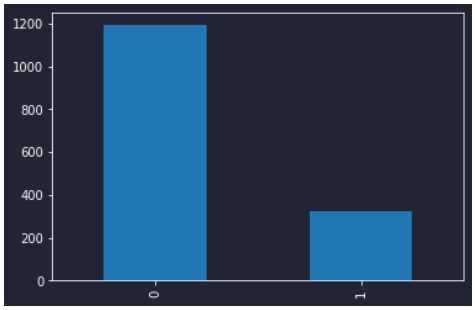
# Ristek Datathon 2022


## Eduardus Tjitrahardja

**NPM**: 2106653602
**Team Name**: edutjie


## Label Distribution

Bisa dilihat disini bahwa data label/target prediksi sangat tidak *balanced* antara 0 dan 1. 0 (1192) jauh lebih banyak dibanding 1 (326). Hal ini mungkin harus ditangani jika diperlukan untuk meningkatkan performa model.

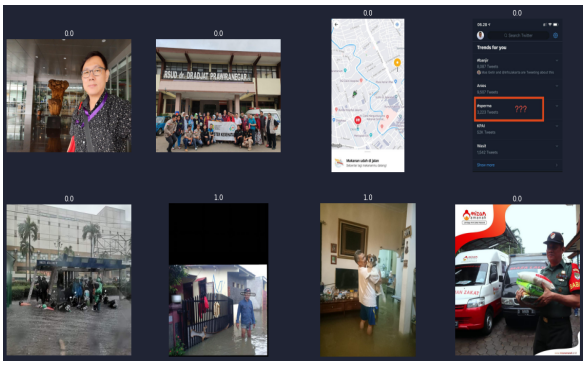



## Data Preprocessing

1. **Membuat function yang berguna untuk membersihkan text**
   Pertama-tama, semua text diubah menjadi lower case, lalu dihilangkan semua tanda baca, angka, dan whitespace. Saya menggunakan library Sastrawan untuk melakukan stemming dan stop words removing pada data text. Setelah itu, saya melihat masih ada kata-kata yang tidak diperlukan yaitu http, yg, dan di jadi saya melakukan penghapusan kata-kata tersebut.

2. **Word cloud kata-kata yang telah dibersihkan:**
   Dapat dilihat pada word cloud dan grafik diatas, bahwa kata-kata yang banyak terdapat pada dataset merupakan banjir, jakarta,air, bantu, dst. Dimana hal itu masuk akal karena dataset ini merupakan dataset tentang banjir.


3. **Images preview:**
   Saat akan melakukan ekstraksi fitur image, image akan diresize menjadi 256 x 256 dan akan dipreprocess dengan function preprocessing bawaan dari pretrained model yang akan saya pakai.



# Feature Extraction

1. **Ekstraksi fitur gambar**

   Membuat fungsi yang akan me-resize dan me-preprocess matriks gambar dan mengekstrak fitur gambar dari model yang dipilih. Model yang saya gunakan adalah VGG19 dan DenseNet121. Dari percobaan yang saya lakukan, ternyata fitur yang diekstrak dari DenseNet121 lah yang meraih score akurasi dan f1 lebih tinggi dari VGG19, oleh karena itu saya memilih DenseNet121. Hasilnya merupakan dataframe dengan 256 kolom.

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0.002412 | 0.255406 | 0.073089 | 0.020818 | 0.021223 | 0.001798 | 0.008892 | 0.010197 | 0.070118 | 0.018849 | 0.054602 | 0.003118 | 0.007767 | 0.003931 | 0.005033 | 0.030503 | 0.003325 | 0.013837 | 0.002505 | 0.00102 |
| 1 | 0.003262 | 0.209445 | 0.076465 | 0.020601 | 0.013247 | 0.001746 | 0.016001 | 0.010862 | 0.054241 | 0.016908 | 0.127318 | 0.002016 | 0.008529 | 0.003211 | 0.009340 | 0.016651 | 0.002608 | 0.021475 | 0.006082 | 0.00156 |
| 2 | 0.001464 | 0.083154 | 0.018514 | 0.024711 | 0.028696 | 0.002615 | 0.012801 | 0.007457 | 0.061622 | 0.019426 | 0.531163 | 0.002015 | 0.022792 | 0.002975 | 0.012297 | 0.030024 | 0.001120 | 0.037323 | 0.003109 | 0.00099 |
| 3 | 0.001842 | 0.070153 | 0.027252 | 0.012801 | 0.121898 | 0.001826 | 0.005749 | 0.007936 | 0.151572 | 0.014932 | 1.261574 | 0.001957 | 0.006310 | 0.002818 | 0.005722 | 0.030206 | 0.000464 | 0.002036 | 0.006015 | 0.00100 |
| 4 | 0.003214 | 0.142908 | 0.028323 | 0.022380 | 0.038958 | 0.001857 | 0.010106 | 0.006874 | 0.074566 | 0.024453 | 0.093079 | 0.002279 | 0.008727 | 0.004549 | 0.010948 | 0.022070 | 0.003543 | 0.034039 | 0.003839 | 0.00083 |

5 rows × 256 columns

2. **Extraksi fitur textual**

   a. Pertama tokenize dulu text yang sudah dibersihkan menggunakan NLTK.
   b. Melakukan word embedding menggunakan pretrained model berbahasa Indonesia dari FastText yang bernama cc.id.300. Hasilnya adalah dataframe dengan 300 kolom.

```
embedded_words = [[ft_model.wv[word] for word in token] for token in tqdm(word_tokens)]
embedded_words_encoding = [np.mean(embedded_word, axis=0) for embedded_word in tqdm(embedded_words)]
df_embedding = pd.DataFrame(embedded_words_encoding)
df_embedding.head()
```

Python

```
100%|██████████| 1518/1518 [00:00<00:00, 13412.56it/s]
100%|██████████| 1518/1518 [00:00<00:00, 41538.87it/s]
```

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | ... | 290 | 291 | 292 | 293 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0.009883 | -0.021286 | 0.025969 | 0.092511 | 0.028668 | -0.028770 | 0.016315 | -0.010510 | 0.008982 | -0.090800 | ... | 0.007333 | 0.013171 | 0.015966 | -0.002016 |
| 1 | 0.034512 | 0.009235 | 0.018049 | 0.045457 | 0.006226 | -0.075077 | 0.004485 | -0.026857 | -0.048104 | -0.050542 | ... | -0.040443 | -0.033196 | -0.006029 | 0.014043 |
| 2 | 0.037173 | -0.050290 | 0.011853 | 0.118136 | -0.018496 | -0.046303 | 0.029109 | 0.011791 | -0.015547 | -0.080154 | ... | -0.032494 | -0.030817 | -0.034458 | 0.015121 |
| 3 | 0.000976 | -0.008979 | 0.025069 | 0.051203 | 0.021625 | -0.046954 | 0.021365 | 0.008321 | -0.029315 | -0.015738 | ... | -0.010753 | 0.018024 | -0.011522 | -0.014646 |
| 4 | 0.029130 | 0.045310 | 0.027381 | 0.040282 | -0.047682 | 0.001377 | 0.017470 | -0.008542 | 0.011416 | -0.097448 | ... | -0.014846 | -0.011152 | -0.055743 | 0.017050 |

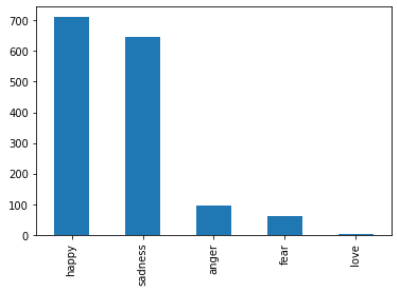
5 rows × 300 columns

# Feature Engineering

1. **Rename kolomnya lalu combine fitur gambar dan textual.**

2. **Fitur Emotion**

   Menggunakan pretrained model dari "StevenLimcorn/indonesian-roberta-base-emotion-classifier" untuk menganalisis emosi dari text. Hasilnya model ini dapat membedakan emosi senang dan sedih, jadi saya memutuskan untuk menggunakannya sebagai fitur. Hasil

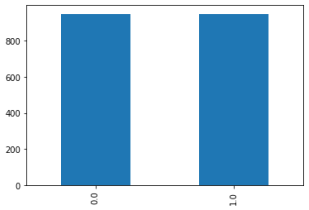


dataframe emosi yang dihasilkan model di-encode menjadi one hot encoder lalu di concat dengan X (features).

## Prepare Data For Training

1. **Split training dan validation data**

2. **SMOTE**
   Saya menggunakan SMOTE untuk melakukan oversampling supaya data dan label menjadi balanced.




## Modeling and Training

1. **CatBoost**
   Saya menggunakan CatBoostClassifier dengan max_depth 9. Dari sini saya mendapatkan classification report seperti ini (Tidak memakai SMOTE):

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0.0 | 0.88 | 0.98 | 0.93 | 244 |
| 1.0 | 0.87 | 0.45 | 0.59 | 60 |
| accuracy | | | 0.88 | 304 |
| macro avg | 0.88 | 0.72 | 0.76 | 304 |
| weighted avg | 0.88 | 0.88 | 0.86 | 304 |

2. **LightGBM**
   Saya mencoba juga menggunakan LightGBM(LGBM) dan memperoleh score f1 yang cukup baik dibandingkan dengan CatBoost. Classfication report (kiri tidak memakai SMOTE dan yang kanan menggunakan SMOTE):

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.92 | 0.96 | 0.94 | 244 |
| 1 | 0.80 | 0.65 | 0.72 | 60 |
| accuracy | | | 0.90 | 304 |
| macro avg | 0.86 | 0.80 | 0.83 | 304 |
| weighted avg | 0.89 | 0.90 | 0.89 | 304 |

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.91 | 0.91 | 0.91 | 244 |
| 1 | 0.63 | 0.65 | 0.64 | 60 |
| accuracy | | | 0.86 | 304 |
| macro avg | 0.77 | 0.78 | 0.77 | 304 |
| weighted avg | 0.86 | 0.86 | 0.86 | 304 |

Dengan model yang tanpa SMOTE, saya mendapatkan score f1 sekitar 0.78 di submisi publik dan yang denga SMOTE lebih rendah.

3. **Deep Learning (Final Model dengan SMOTE mendapatkan f1 score 0.8 di submisi publik)**
   Saya mencoba membuat model sederhana deep learning yang menggunakan 3 layer dense dan 2 layer drop out, lalu dicompile dengan optimizer adam, binary crossentropy loss, dan accuracy metric. Classfication report (kiri tidak memakai SMOTE dan yang kanan menggunakan SMOTE):

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.91 | 0.95 | 0.93 | 244 |
| 1 | 0.75 | 0.63 | 0.68 | 60 |
| accuracy | | | 0.88 | 304 |
| macro avg | 0.83 | 0.79 | 0.81 | 304 |
| weighted avg | 0.88 | 0.88 | 0.88 | 304 |

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.95 | 0.86 | 0.90 | 244 |
| 1 | 0.59 | 0.82 | 0.69 | 60 |
| accuracy | | | 0.85 | 304 |
| macro avg | 0.77 | 0.84 | 0.79 | 304 |
| weighted avg | 0.88 | 0.85 | 0.86 | 304 |