Ristek Datathon 2022

Eduardus Tjitrahardja

NPM: 2106653602
Team Name: edutjie

Overview

Pada kompetisi ini, peserta diminta untuk menggunakan data cuitan tentang banjir beserta dengan gambarnya yang di-*scrape* dari Twitter. Data tersebut digunakan untuk memprediksi apakah cuitan tersebut dapat digunakan untuk *actionable intelligence* yang berguna untuk menanggulangi bencana.

Loading Data

Pertama saya load dulu data yang diberikan dari Kaggle, lalu saya inspect datanya.

1. **Data preview**

	created_at	id	user_id	user_name	url	text	media	label
0	2020-02-23 04:06:39+00:00	1231430140824973313	133931409	r0b1 sur1a (黃玉春)	https://twitter.com/R0b1Sur1a/status/123143014...	Akhirnya sampai juga setelah menerobos banjir....	ERbrKgFU4AAnADb.jpg	0
1	2020-01-05 01:46:46+00:00	1213637932411580417	253063316	Beradaptasi di Era Pandemi 🟠	https://twitter.com/MarikaRahman_/status/12136...	Kemekes RI, IDI Banten, IBI Banten dan PPNI Ba...	ENe1HUVUEAANLFT.jpg	0
2	2020-01-18 06:22:09+00:00	1218418277946396673	64318803	rywyu	https://twitter.com/rywyu/status/1218418277946...	Dikarenakan cikini rada2 banjir tdi pagi hingg...	EOiw80RU0AcJ4CI.jpg	0
3	2020-02-22 23:38:00+00:00	1231362534717837313	17383917	ICALIZERS	https://twitter.com/icalizers/status/123136253...	#sperma TT dikala warga jakarta sedang prihati...	ERatrt9UYAAtLMI.jpg	0
4	2019-12-17 10:54:31+00:00	1206890412574568449	3102973556	AN	https://twitter.com/lokbin103/status/120689041...	KUIS!\n\nJakarta banjir parah hari ini, pertan...	EL-8Z-vUcAIiIlV.jpg	0

2. **Check null value**

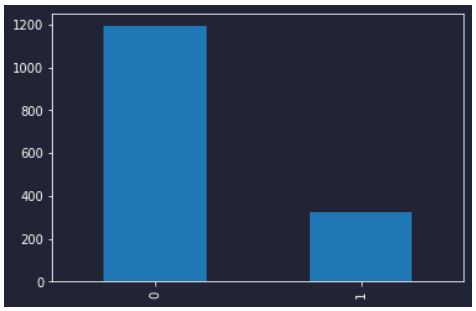
   Dataset ini sudah bersih dari null value, bisa kita lihat pada gambar di kiri bahwa tidak ada *missing data*.

```
train_df.isnull().sum()
created_at    0
id            0
user_id       0
user_name     0
url           0
text          0
media         0
label         0
dtype: int64
```

```
test_df.isnull().sum()
created_at    0
id            0
user_id       0
user_name     0
url           0
text          0
media         0
dtype: int64
```

3. **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.



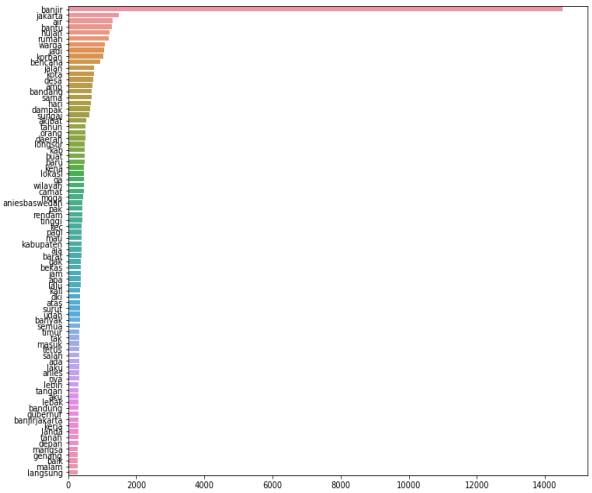
Data Preprocessing

1. **Ini merupakan function yang berguna untuk membersihkan text:**

```python
def text_precessor(text):
    # lowercase
    text = text.lower()
    # remove punctuation
    text = text.translate(str.maketrans('', '', string.punctuation))
    # remove numbers
    text = text.translate(str.maketrans('', '', string.digits))
    # remove whitespaces
    text = text.strip()

    # stemming
    factory = StemmerFactory()
    stemmer = factory.create_stemmer()
    text = stemmer.stem(text)

    # remove stopwords
    factory = StopWordRemoverFactory()
    stopword = factory.create_stop_word_remover()
    stop = stopword.remove(text)

    # tokenization
    tokens = nltk.tokenize.word_tokenize(stop)
    return ' '.join(tokens)
```

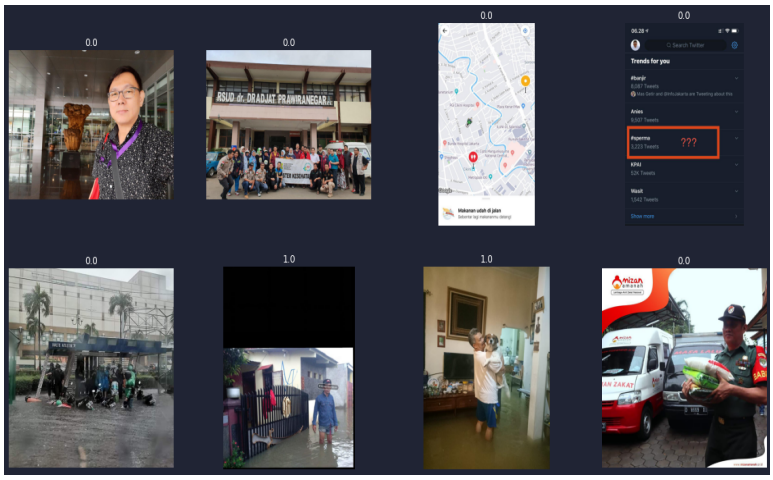
Pertama-tama, semua text diubah menjadi lower case, lalu dihilangkan semua tanda baca, angka, dan whitespace. Saya menggunakan library Sastrawan untuk melakukan stemming dan stop words removing pada data text. Setelah itu, saya melihat masih ada kata-kata yang tidak diperlukan yaitu http, yg, dan di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

```python
def extract_features(df, model, preprocess_input):
    img_size = 256
    batch_size = 16
    n_batches = df.shape[0] // batch_size + 1
    features = {}
    for b in notebook.tqdm(range(n_batches)):
        start = b * batch_size
        end = (b + 1) * batch_size
        batch = df[start:end]
        images = []
        for image_path in batch["media"]:
            img = image.load_img(f"dataset/media/Image/{image_path}", target_size=(img_size, img_size))
            x = image.img_to_array(img)
            x = np.expand_dims(x, axis=0)
            x = preprocess_input(x)
            images.append(x)
        images = np.vstack(images)
        batch_features = model.predict(images)
        for i, index in enumerate(batch.index):
            features[index] = batch_features[i]
    return pd.DataFrame.from_dict(features, orient="index")
```

Fungsi ini 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 datafram dengan 300 kolom.

```python
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

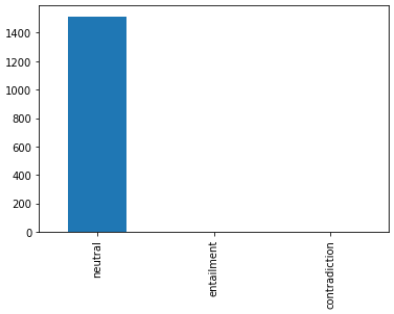
1. **Rename kolomnya lalu combine fitur gambar dan textual.**

	image_0	image_1	image_2	image_3	image_4	image_5	image_6	image_7	image_8	image_9	...	text_290	text_291	text_292	text_293	text_294	text_295	text_296	text_297	text_29
0	0.002412	0.255406	0.073089	0.020818	0.021223	0.001798	0.008892	0.010197	0.070118	0.018849	...	0.007333	0.013171	0.015966	-0.002016	0.032844	-0.006217	0.022417	-0.057426	0.01750
1	0.003262	0.209445	0.076465	0.020601	0.013247	0.001746	0.016001	0.010862	0.054241	0.016908	...	-0.040443	-0.033196	-0.006029	0.014043	-0.015839	0.013577	0.005713	0.001407	-0.00354
2	0.001464	0.083154	0.018514	0.024711	0.028696	0.002615	0.012801	0.007457	0.061622	0.019426	...	-0.032494	-0.030817	-0.034458	0.015121	-0.031749	0.027591	-0.008889	-0.035597	-0.00887
3	0.001842	0.070153	0.027252	0.012801	0.121898	0.001826	0.005749	0.007936	0.151572	0.014932	...	-0.010753	0.018024	-0.011522	-0.014646	0.006166	0.028416	0.047759	0.010205	-0.00738
4	0.003214	0.142908	0.028323	0.022380	0.038958	0.001857	0.010106	0.006874	0.074566	0.024453	...	-0.014846	-0.011152	-0.055743	0.017050	-0.014798	0.048946	0.032517	-0.016963	-0.01231

5 rows × 556 columns

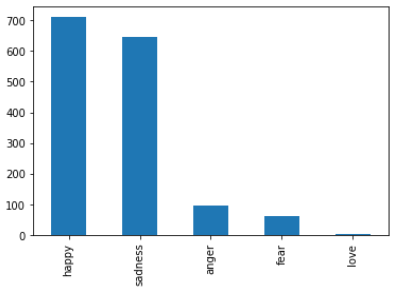
2. **Fitur Sentiment**

   Menggunakan pretrained model dari “w11wo/indonesian-roberta-base-indonli” untuk menganalisis sentiment dari text data. Hasilnya hampir semua text sentimennya netral, jadi fitur ini tidak akan berdampak pada model dan tidak akan saya pakai.



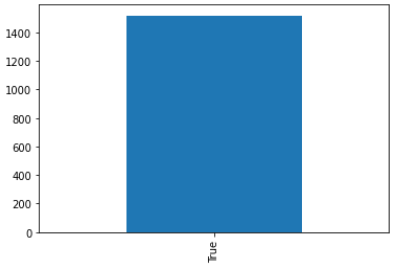
3. **Fitur Emotion**

   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).



4. **Fitur Is Banjir**

   Saya awalnya berpikir untuk menambahkan fitur flag apakah banjir adalah di text. Namun setelah di analisa, semua text pada dataset mengandung kata “banjir”. Jadi fitur ini tidak akan saya gunakan karena tidak berguna.



Prepare Data For Training

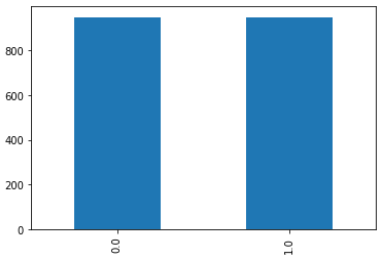
1. **Split training dan validation data**

```
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=42)
```

2. **SMOTE**

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

```
smt = SMOTE()
print("Size of X_train before SMOTE: ", X_train.shape)
X_train, y_train = smt.fit_resample(X_train, y_train)
print("Size of X_train after SMOTE: ", X_train.shape)

Size of X_train before SMOTE:  (1214, 561)
Size of X_train after SMOTE:  (1896, 561)
```



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):

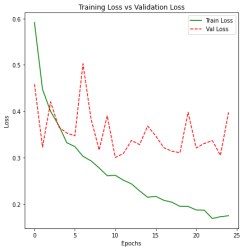
	precision	recall	f1-score	support
0	0.92	0.96	0.94	244
1	0.80	0.65	0.72	60
accuracy			0.90	304
macro avg	0.86	0.80	0.83	304
weighted avg	0.89	0.90	0.89	304

	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