Preprocessing 0 - Data frame preparation

- Aggregate the dataset with different formats (Json, csv) or source into single data frame for further manipulation.
- Remove the unnecessary data

	_score	hashtags	tweet_id	text	emotion
0	391	[Snapchat]	0x376b20	People who post "add me on #Snapchat" must be dehydrated. Cuz man.... that's <LH>	anticipation
1	433	[freepress, TrumpLegacy, CNN]	0x2d5350	@brianklaas As we see, Trump is dangerous to #freepress around the world. What a <LH> <LH> #TrumpLegacy. #CNN	sadness
2	232	[bibleverse]	0x28b412	Confident of your obedience, I write to you, knowing that you will do even more than I ask. (Philemon 1:21) 3/4 #bibleverse <LH> <LH>	test
3	376	[]	0x1cd5b0	Now ISSA is stalking Tasha 😂 😂 😂 <LH>	fear
4	989	[]	0x2de201	"Trust is not the same as faith. A friend is someone you trust. Putting faith in anyone is a mistake." ~ Christopher Hitchens <LH> <LH>	test
5	120	[authentic, LaughOutLoud]	0x1d755c	@RISKshow @TheKevinAllison Thx for the BEST TIME tonight. What stories! Heartbreakingly <LH> #authentic #LaughOutLoud good!!	joy
6	1021	[]	0x2c91a8	Still waiting on those supplies Liscus. <LH>	anticipation
7	481	[]	0x368e95	Love knows no gender. 🥺 😭 <LH>	joy

Preprocessing 1 - Missing Values & Duplicates

- Examine the missing values and duplicates records which focus on text column.

```python
missingValues = tweets_df.isnull()
total_missing = missingValues.sum()
print(total_missing)
```
✓ 0.2s

```
_score      0
hashtags    0
tweet_id    0
text        0
emotion     0
dtype: int64
```

- Drop the duplicated data

```
total records: 1867535
number of duplicated tweets: 3920
ratio of duplicated tweets: 0.0020990235792100282
number of duplicates in test data: 0
total tweets after dropping duplicates: 1863615
```

Preprocessing 2 – Clean data

1. Convert text to lowercase

2. Remove URLs from text (remove https://xxx)

3. Extract hashtags and clean text (remove the # but keep the consecutive word in text)

4. remove placeholders (remove <HL>)

5. remove repeated punctuation marks: (!!! -> !)

Data exploration an analysis

Analyzing the _score distribution in each emotion:

Emotion	Count	Mean	Std Dev	Min	25%	50%	75%	Max
Anger	39,711.0	512.031578	296.443840	1.0	255.0	510.0	769.0	1024.0
Anticipation	248,687.0	511.625248	295.255362	1.0	256.0	511.0	767.0	1024.0
Disgust	138,993.0	512.737721	295.360972	1.0	257.0	512.0	769.0	1024.0
Fear	63,819.0	512.853523	295.955314	1.0	255.0	514.0	769.0	1024.0
Joy	514,224.0	512.820341	295.808975	1.0	256.0	513.0	769.0	1024.0
Sadness	193,195.0	511.613448	295.653472	1.0	256.0	510.0	768.0	1024.0
Surprise	48,203.0	511.964567	295.797291	1.0	253.0	513.0	767.0	1024.0
Trust	204,878.0	512.566708	295.624719	1.0	256.0	514.0	768.0	1024.0
Test	411,905.0	512.536051	295.718023	1.0	256.0	513.0	768.0	1024.0

The distribution is quite similar so I guess this score_lable is generated by machine/specific model or it may be the result of normalization.

Training data selection and sampling

```
# select the data with _score being top 25% of the dataset as they are emotional-iconic data.
tweets_train = tweets_train[tweets_df['_score'] > 769]

# sampling the data to reduce the running time
tweets_train = tweets_train.sample(frac=0.05, random_state=42)
```

Because this time I want to test the training time of the model, I set frac=0.05 which is very small. It should have better performance if I sample larger training data size.

Data exploration after sampling

Emotion	Count	Mean	Std Dev	Min	25%	50%	75%	Max
Anger	501.0	897.267465	72.256517	770.0	835.0	895.0	959.00	1024.0
Anticipation	3121.0	893.826658	73.268131	770.0	830.0	892.0	956.00	1024.0
Disgust	1729.0	895.616541	74.522255	770.0	831.0	894.0	962.00	1024.0
Fear	784.0	898.970663	73.160300	770.0	834.0	902.5	961.00	1024.0
Joy	6400.0	896.777031	74.229857	770.0	832.0	895.5	962.00	1024.0
Sadness	2396.0	896.914441	73.183344	770.0	834.0	897.0	960.25	1024.0
Surprise	599.0	901.893155	73.722330	770.0	837.5	906.0	965.00	1024.0
Trust	2535.0	896.908481	73.117406	770.0	833.0	896.0	960.00	1024.0

Check I have enough training data to train the model for each emotion.

You can see the scores of the selected data shift to the right of about 400 points.

Model

I chose **Roberta** as it's transformer-based model that can capture the semantic meaning of text which is useful for emotional classification tasks.

```
# prepare model
model = RobertaForSequenceClassification.from_pretrained('roberta-base', num_labels=len(label_map))

# setting training arguments
training_args = TrainingArguments(
    output_dir='./results',
    evaluation_strategy="no",  # Disables evaluation during training because we don't have data to do the validation
    learning_rate=2e-5,
    per_device_train_batch_size=16,
    num_train_epochs=3,
    weight_decay=0.01,
    logging_dir='./logs',
    logging_steps=10,
)
```

Result

1. Roberta For Sequence Classification
    a. Training time with (GPU): 60 min
    b. Mean F1 Score: 0.427
2. Decision Tree
    a. Training time with (CPU): 20 min
    b. Mean F1 Score: 0.287`

Comparison

The decision tree cannot perform well when the data have correlation among themselves because every feature/node for splitting can only be used once. In this case, the `semantic correlation` is important, the model which can capture this kind of correlation is a good choice to this task.

Decision tree will have difficulty to well-split the data with correlation. In this case, emoji and hashtags may have strong correlation and semantic correlations even if their expressions are quite different (ex: both of "I got a new car!" and "I have a good news" are **joyful emotion**).