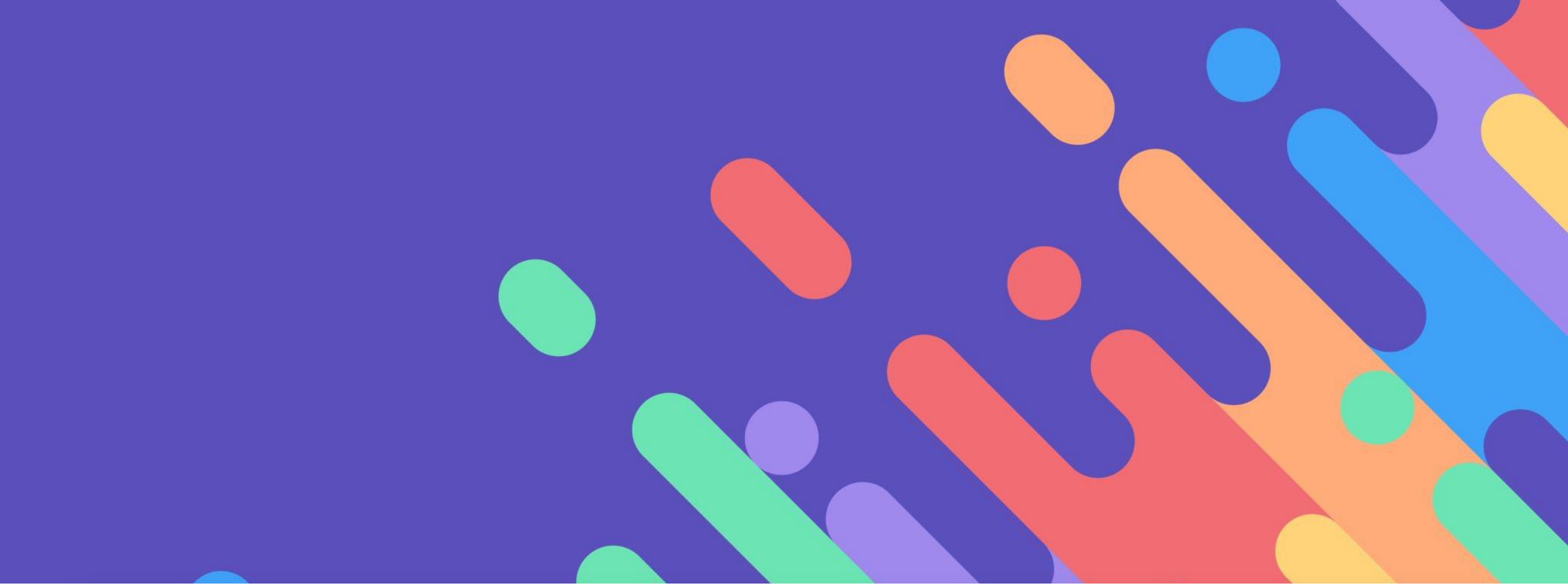


Data Mining Lab 2

NTU B10209001

Prediction of emotion of Tweets

Data preparation

1

load data

- json format

2

merge

- identification
- emotion

3

split data

- train_data
- test_data

4

save file

- pkl format

Data Observation

Tweets labeled by emotion

emotions

joy

anger

fear

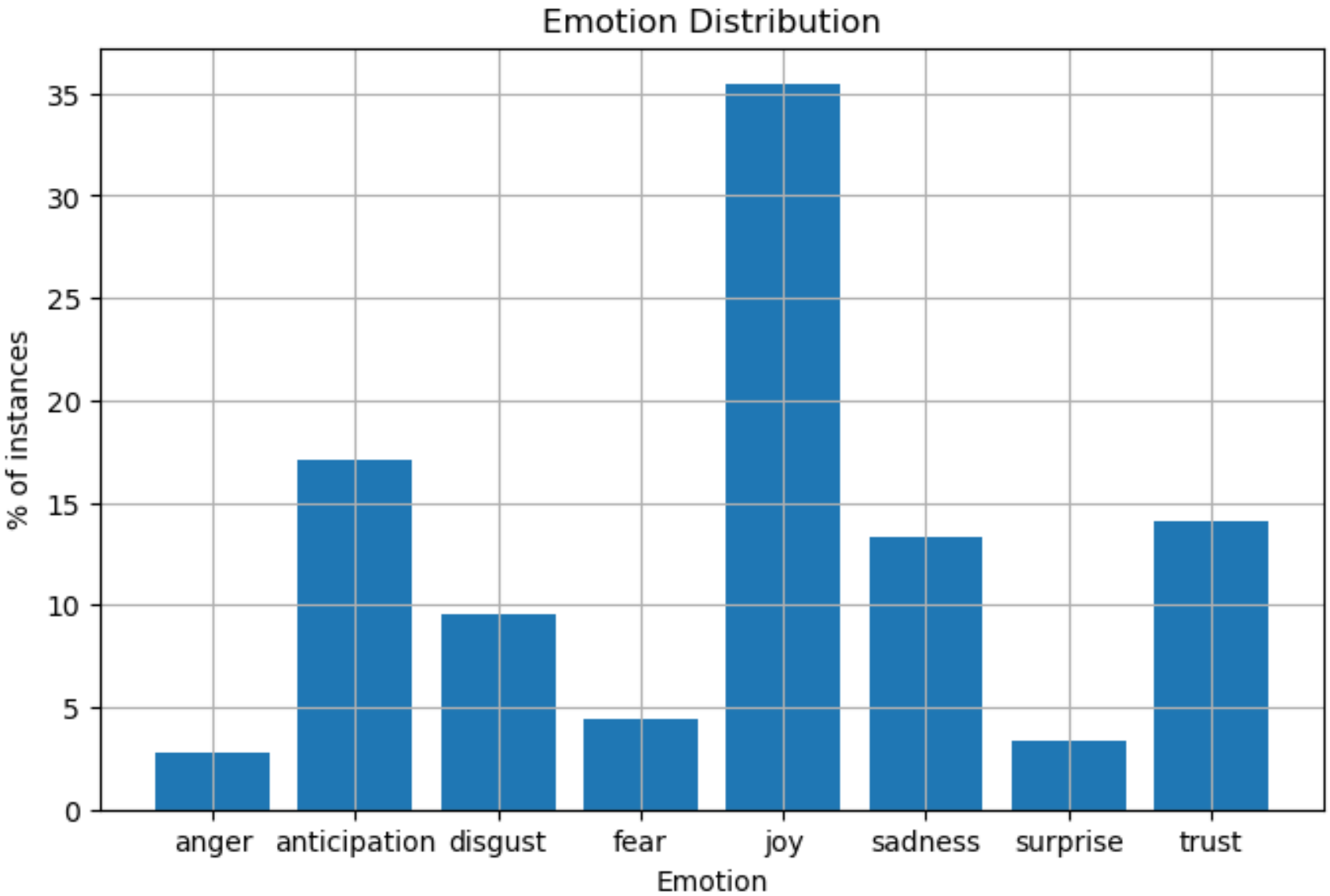
surprise

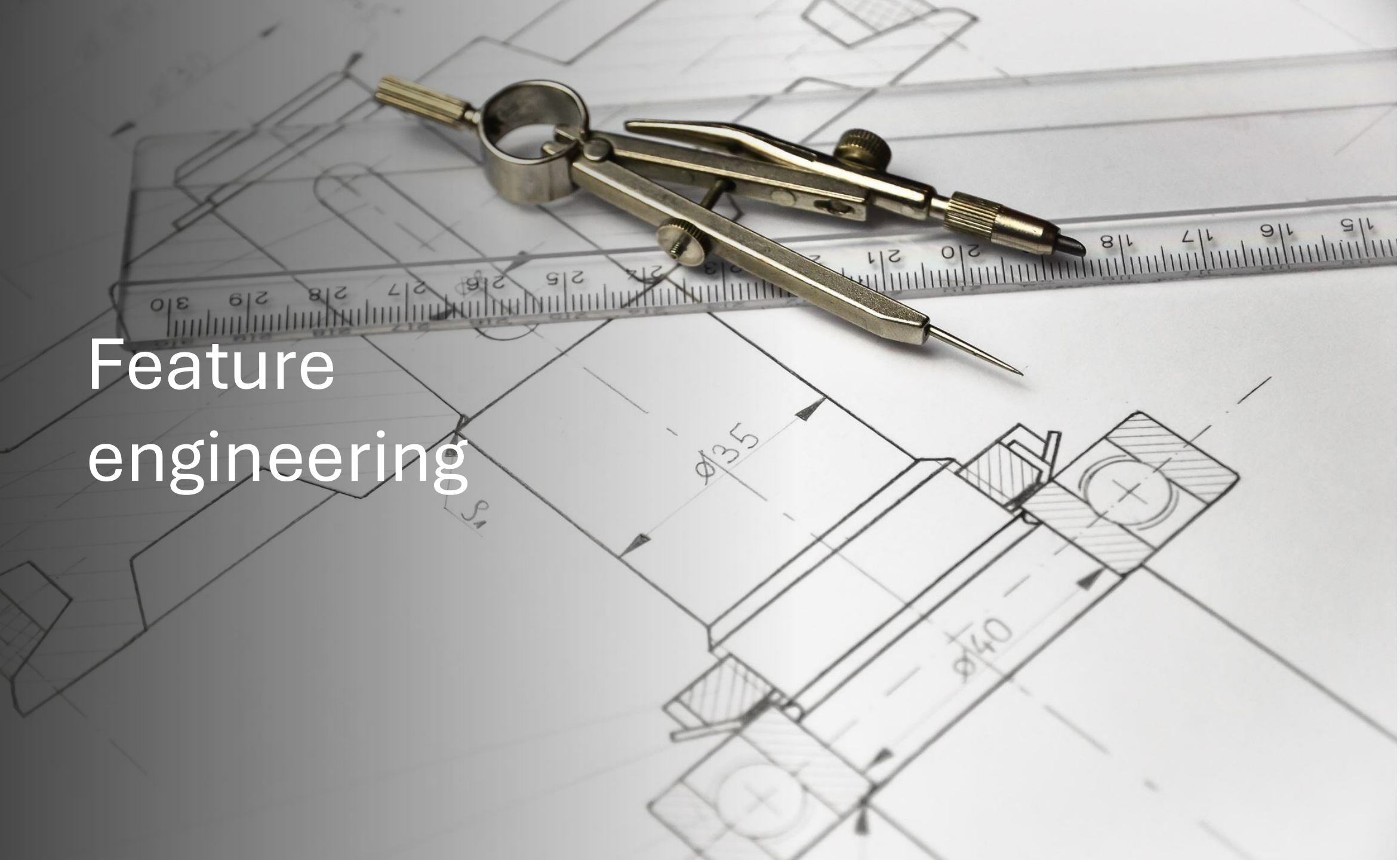
trust

sadness

anticipation

disgust


Emotion Distribution
% of instances
0
5
10
15
20
25
30
35
anger
anticipation
disgust
fear
joy
sadness
surprise
trust
Emotion



Feature engineering

Vectorize word in text

向量化文本資料，嘗試使用不同的向量化工具優化預測結果。

Bag Of Word與Tf-IDF放到不同模型訓練的差異不大，所以後來嘗試Word2Vec，預期可以更好的提供模型訓練上下文關係。

Bag Of Word

```python
import nltk
# build analyzers (bag-of-words)
BOW_500 = CountVectorizer(max_features=500, tokenizer=nltk.word_tokenize)

# apply analyzer to training data
BOW_500.fit(train_data['text'])

train_data_BOW_features_500 = BOW_500.transform(train_data['text'])

## check dimension
train_data_BOW_features_500.shape

## adjust its type for visualize
train_data_BOW_features_500.toarray()
```

Python

TF-IDF

Using scikit-learn `TfidfVectorizer` perform word frequency and use these as features to train a model.
使用不同的向量化工具

```python
from sklearn.feature_extraction.text import TfidfVectorizer

# 初始化 TfidfVectorizer，并限制特征数为 1000
tfidf_vect = TfidfVectorizer(max_features=1000)

# 使用 TF-IDF 向量化器进行训练集和测试集的向量化
train_tfidf = tfidf_vect.fit_transform(train_data["text"])
test_tfidf = tfidf_vect.transform(test_data["text"])

# 获取特征名称列表
feature_names_tf = tfidf_vect.get_feature_names_out()
```

Python

Word2Vec

```python
# 訓練 Word2Vec 模型
word2vec_model = Word2Vec(
    sentences=processed_texts,
    vector_size=100,  # 詞向量維度
    window=5,         # 上下文窗口大小
    min_count=1,      # 忽略出現頻率小於 min_count 的詞
    workers=4,        # 使用的 CPU 核數
    sg=0,             # CBOW 模型 (sg=1 為 Skip-Gram 模型)
    epochs=10         # 訓練輪數
)

import os
# 保存模型
model_folder = "word2vec_model/"
os.makedirs(model_folder, exist_ok=True)
model_path = os.path.join(model_folder, "word2vec_trained.model")
word2vec_model.save(model_path)
print(f"Word2Vec 模型已保存到: {model_path}")
```

Python

model

主要使用資料庫內現有的模型，自己建立簡單的layer預測結果都不如預期。我應該花更多時間研究建立基礎神經網路的訓練模型。

本來想用Softmax嘗試預測，但因為時間不夠就沒做了。

Decision Tree

```
## build DecisionTree model
DT_model = DecisionTreeClassifier(random_state=1)

## training!
DT_model = DT_model.fit(X_train, y_train)

## predict!
y_train_pred = DT_model.predict(X_train)


from sklearn.metrics import accuracy_score

# 計算訓練集的準確度(Decision Tree)
acc_train = accuracy_score(y_true=y_train, y_pred=y_train_pred)

# 輸出準確度，將結果四捨五入保留兩位小數
print('training accuracy: {}'.format(round(acc_train, 2)))
DTA = round(acc_train, 2)
```

Naïve Bayes

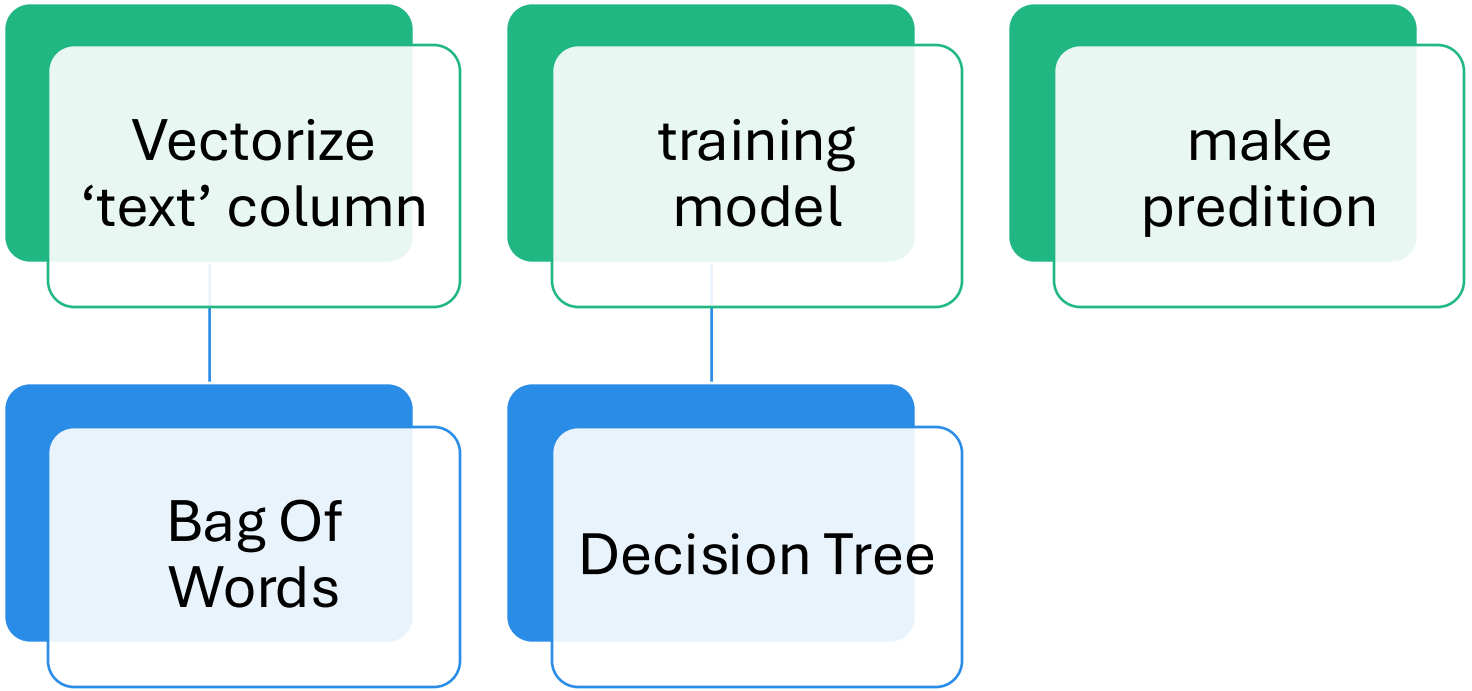
```python
# Step 2: 建立和訓練 Naive Bayes 模型
nb_model = MultinomialNB()
nb_model.fit(X_train, y_train)

# Step 3: 模型預測
y_train_pred = nb_model.predict(X_train)

# 計算訓練集的準確度(Naive Bayes)
acc_train = accuracy_score(y_true=y_train, y_pred=y_train_pred)

# 輸出準確度，將結果四捨五入保留兩位小數
print('training accuracy: {}'.format(round(acc_train, 2)))
NBA = round(acc_train, 2)
```

Python

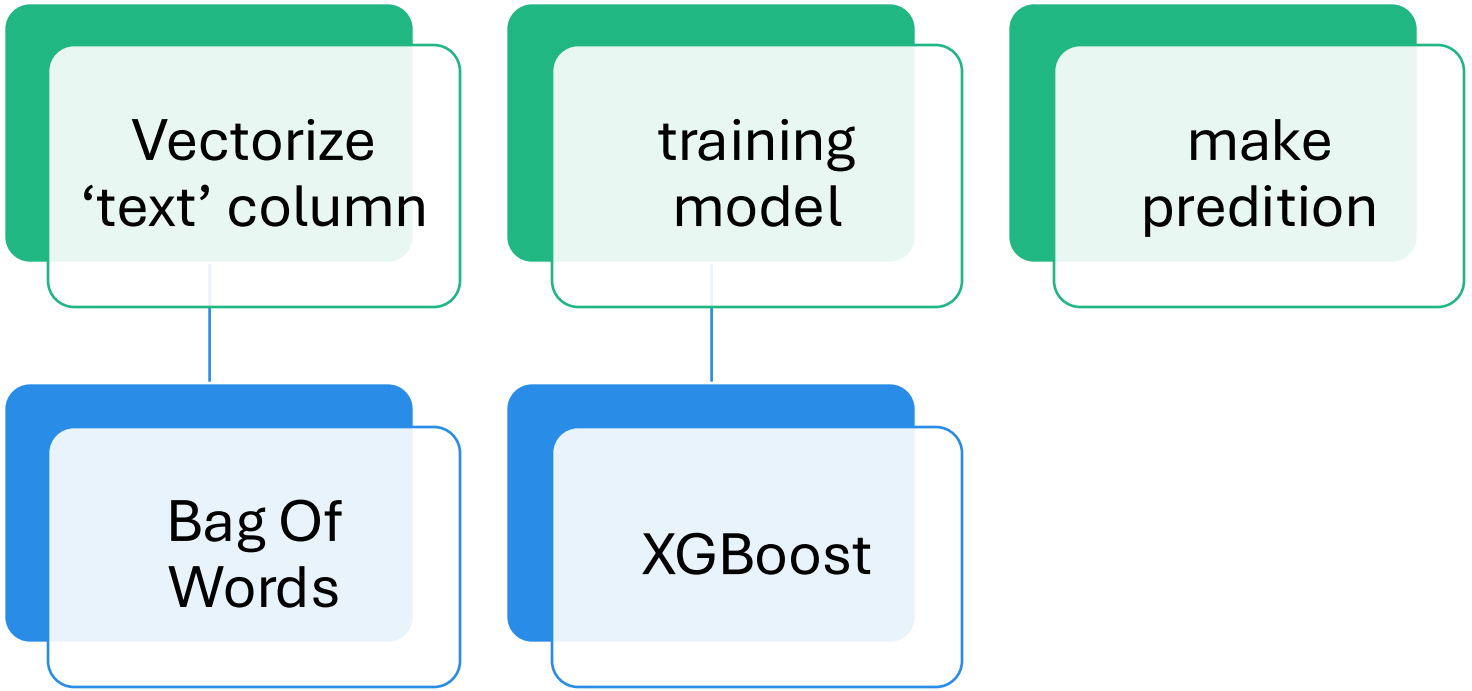
Flow Chart

將所有嘗試的流程圖畫出

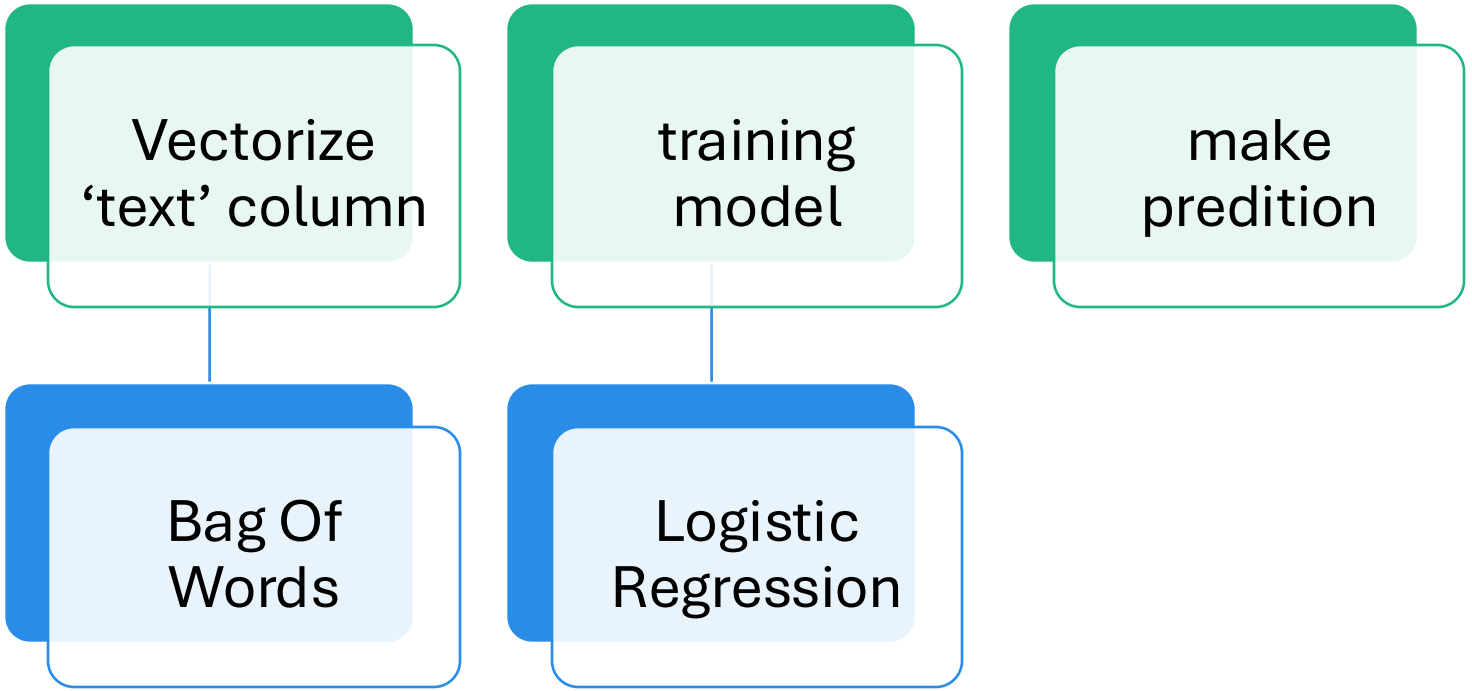
Flow chart of submission 0.28376



Flow chart of submission 0.20148

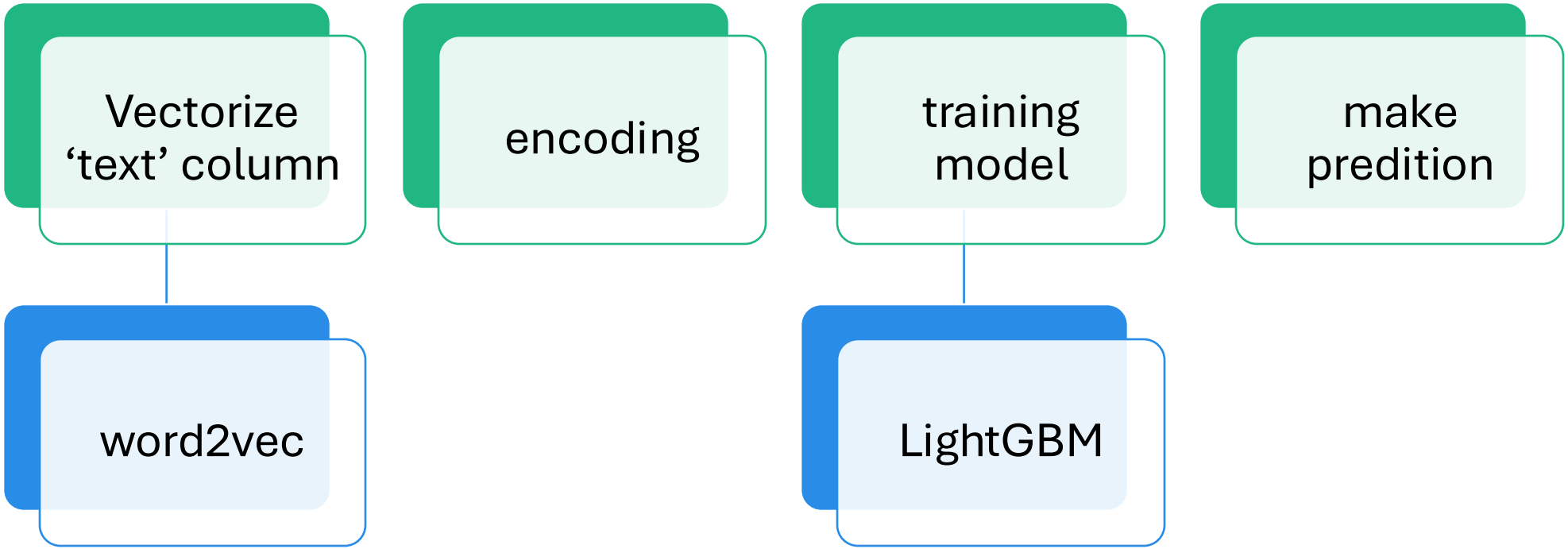


Flow chart of submission 0.33557

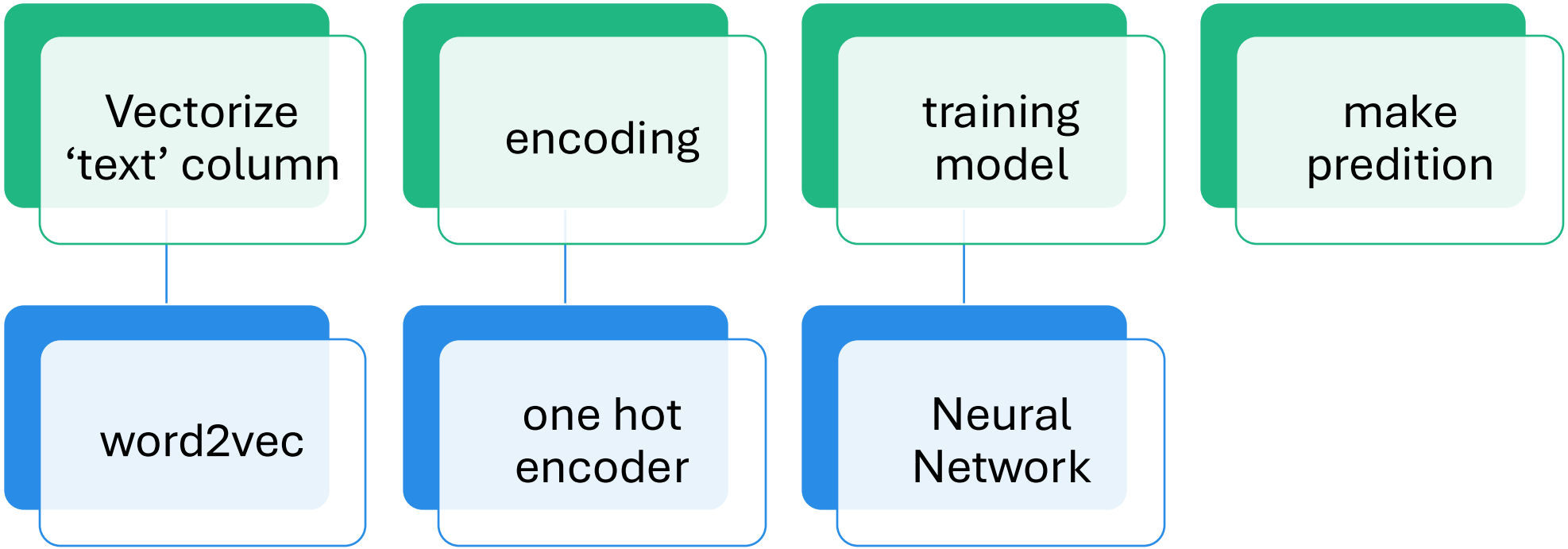


改變了向量化工具，自行建立模型

Flow chart (0.28818)



Flow chart (undone)



訓練數據類別分佈:

joy	516017
anticipation	248935
trust	205478
sadness	193437
disgust	139101
fear	63999
surprise	48729
anger	39867

Name: count, dtype: int64

預測結果類別分佈:

joy	403580
anticipation	4170
sadness	3669
fear	263
trust	151
surprise	89
disgust	42
anger	8

Name: count, dtype: int64

Training LightGBM model...

待處理問題

推測是因為訓練集中的特定情緒類別過多發生了過擬合現象。做了取樣本訓練結果仍相同。

總結

雖然延長了作業繳交期限，但因為其他課程安排仍沒有更多時間可以排來繼續做Lab2。但是感覺有逐漸抓到有哪些可以調整。

過擬合現象挺容易發生的，但是沒有在繳交前特別打開預測文件也不一定能發現，結果應該要印出來特別檢查過。