Relationship Advice Classification

Relationship Advice vs Breakups
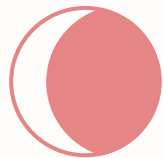
CONTENT

Overview

- Make up or Break up
- Creating classification model
- Identify key features of both subreddits

Problem Statement

Create a classification model to determine the text being relationship advice or breakup.

Data Collection

Code for Webscrapping

Setting Thread to webscrap

```python
1  # creating tags for scrapping
2  thread_1 = 'relationship_advice'
3  thread_2 = 'BreakUps'
4
5  url_1 = ('https://www.reddit.com/r/' + thread_1 + '.json')
6  url_2 = ('https://www.reddit.com/r/' + thread_2 + '.json')
7  url_1, url_2
```

```
('https://www.reddit.com/r/relationship_advice.json',
 'https://www.reddit.com/r/BreakUps.json')
```

```python
1   # looping through first thread
2   thread_1_posts = []
3   after = None
4
5   for a in range(40): #looping 40 times, 25 post each
6       if after == None:
7           current_url = url_1
8       else:
9           current_url = url_1 + '?after=' + after
10      print(current_url)
11      res = requests.get(current_url, headers={'User-agent': 'Pony Inc 1.0'})
12
13      if res.status_code != 200:
14          print('Status error', res.status_code)
15          break
16
17      current_dict = res.json()
18      current_posts = [p['data'] for p in current_dict['data']['children']]
19      thread_1_posts.extend(current_posts)
20      after = current_dict['data']['after']
21
22      # generate a random sleep duration to look more 'natural'
23      sleep_duration = random.randint(2,60)
24      print('Sleep duration: ' + str(sleep_duration))
25      print('Number of posts: ' + str(len(thread_1_posts)))
26      time.sleep(sleep_duration)
```

Check and remove duplicates

```python
1  print(f'No.of duplciated post in Thread {thread_1}: {df_t1.selftext.duplicated().sum()}')
2  print(f'No.of duplciated post in Thread {thread_2}: {df_t2.selftext.duplicated().sum()}')
```

```
No.of duplciated post in Thread relationship_advice: 29
No.of duplciated post in Thread BreakUps: 63
```

Amount of text scrapped

```python
1  print(f'No. of rows in Thread {thread_1}: {df_t1.shape[0]}')
2  print(f'No. of rows in Thread {thread_2}: {df_t2.shape[0]}')
```

```
No. of rows in Thread relationship_advice: 964
No. of rows in Thread BreakUps: 935
```

EDA

Distribution of post in hours/minutes for both threads

Blue = Breakup
Orange = Relationship advice
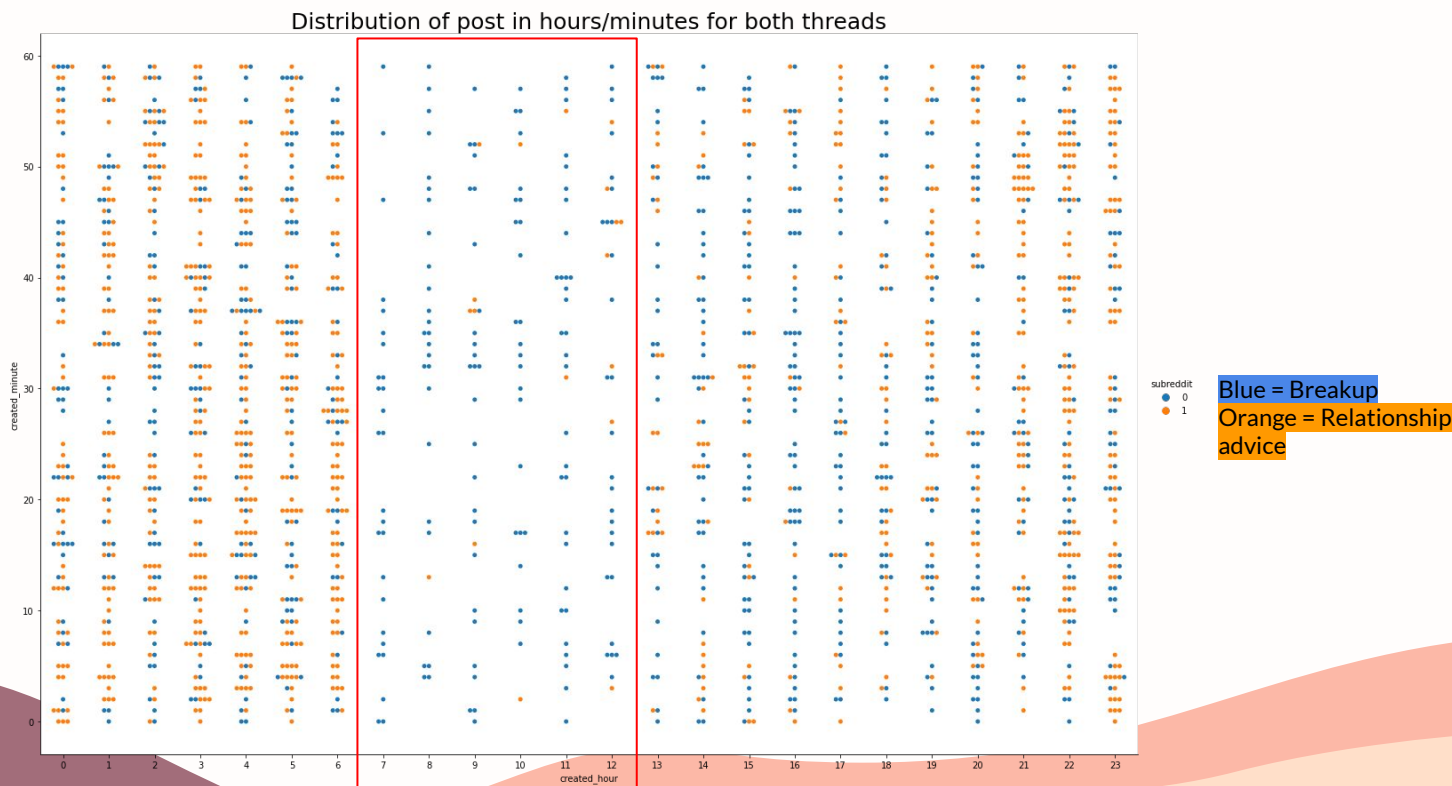
Lesser Relationship advice post

Cleaning of Data

1. **Removing HTML Features**
2. Removing numerical values
3. Lowercase text
4. Removing stopwords
5. Stemming Text

Cleaning of Data

1. Removing HTML Features
2. **Removing numerical values**
3. Lowercase text
4. Removing stopwords
5. Stemming Text

4.1.3 Removing numerical values

```
1   letters_only = re.sub('[^a-zA-Z]',
2                         " ",
3                         example1.get_text())
4
5   letters_only[:50]
```

'So basically I ve been dating an amazing person f'

Cleaning of Data

1. Removing HTML Features
2. Removing numerical values
3. **Lowercase text**
4. Removing stopwords
5. Stemming Text

4.1.4 Lowercase text

```
1   #lowercase text
2   lower_case = letters_only.lower()
3   #splitting text to strings
4   words = lower_case.split()
5   words[:10]
```

```
['so',
 'basically',
 'i',
 've',
 'been',
 'dating',
 'an',
 'amazing',
 'person',
 'for']
```

Cleaning of Data

1. Removing HTML Features
2. Removing numerical values
3. Lowercase text
4. **Removing stopwords**
5. Stemming Text

4.1.5 Removing stopwords

```python
stop_words = [stopwords.words('english') + ['relationship',
                                            'relation',
                                            'advice',
                                            'breakup',
                                            'break',
                                            'broke']]
words = [w for w in words if w not in stop_words]
words[:10]
```

Cleaning of Data

1. Removing HTML Features
2. Removing numerical values
3. Lowercase text
4. Removing stopwords
5. **Stemming Text**

4.1.6 Stemming text ¶

```python
1  p_stemmer = PorterStemmer()
2  words_pstem = [p_stemmer.stem(i) for i in words]
```

```python
1  # Print only those stemmed tokens that are different.
2  for i in range(len(words)):
3      if words[i] != words_pstem[i]:
4          print((words[i], words_pstem[i]))
```

```
('basically', 'basic')
('dating', 'date')
('amazing', 'amaz')
('years', 'year')
('issues', 'issu')
('caring', 'care')
('everything', 'everyth')
('exactly', 'exactli')
('exciting', 'excit')
('feels', 'feel')
('planning', 'plan')
('sticking', 'stick')
('amazing', 'amaz')
('always', 'alway')
('lacked', 'lack')
('excitement', 'excit')
('expected', 'expect')
('someone', 'someon')
```
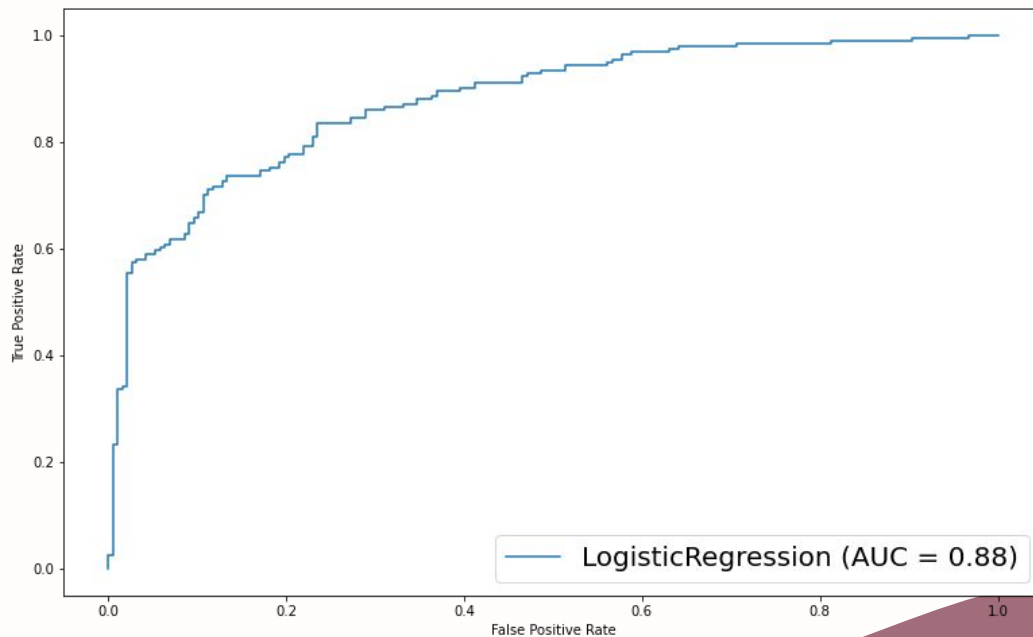
Baseline Model (Logistic Regression)

Accuracy score for Train data set: 0.994
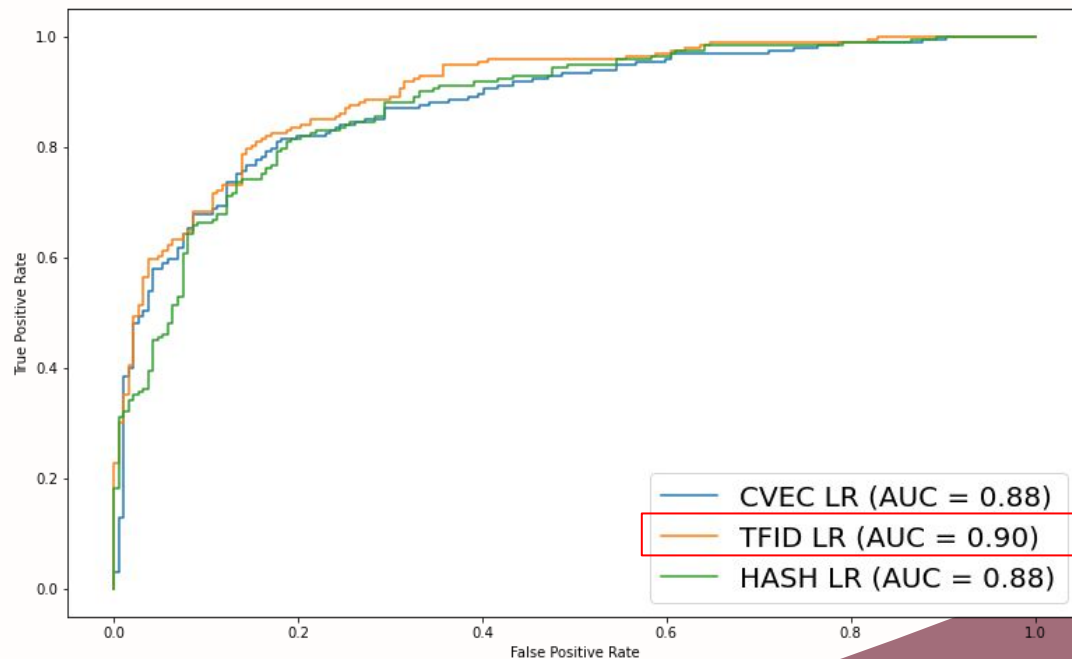Accuracy score for Test data set: 0.781

Specificity: 0.775
Sensitivity: 0.787

Alternative Model (part 1)

model	vectorizer	train_score	test_score	time(s)
LogisticRegression	CountVectorizer	0.815963	0.810526	35
LogisticRegression	TFIDVectorizer	0.852886	0.807895	25
LogisticRegression	HashingVectorizer	0.821880	0.786842	3

Alternative Model (part 2)

| model	vectorizer	train_score	test_score	time(s)
LogisticRegression	TFIDVectorizer	0.848940	0.823684	138
KNN	TFIDVectorizer	0.830465	0.818421	72
MultinomialNB	TFIDVectorizer	0.849603	0.847368	44
SVC	TFIDVectorizer	0.851555	0.805263	73
CART	TFIDVectorizer	0.704468	0.671053	78
BaggingClassifier	TFIDVectorizer	0.773736	0.744737	164
ADABoost	TFIDVectorizer	0.777725	0.784211	384
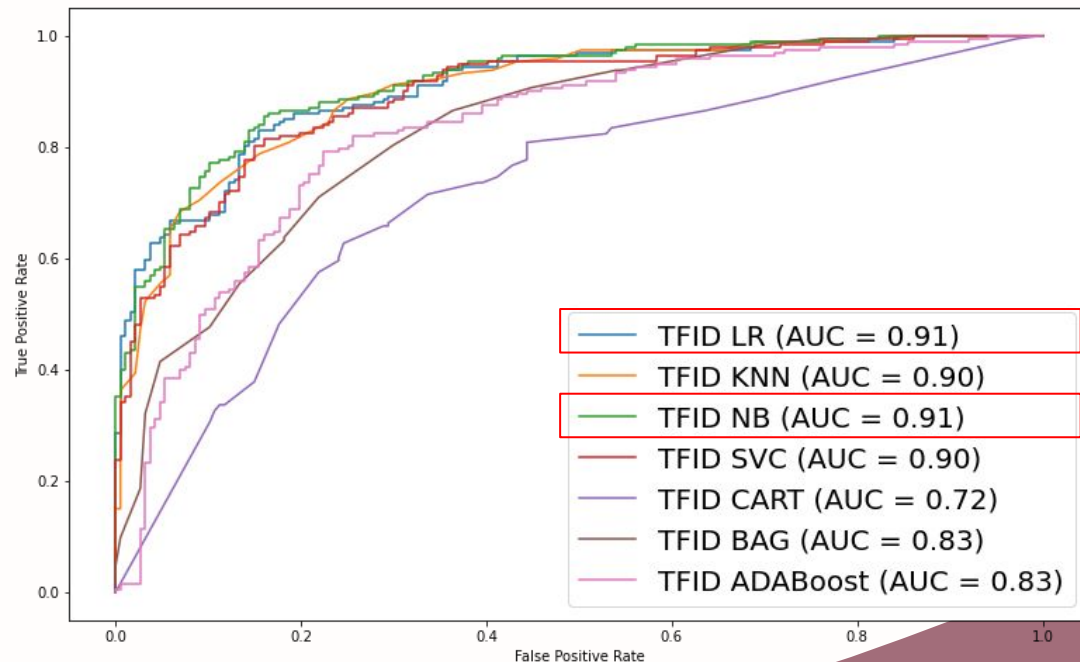
Final Model (TFID Logistic Regression)
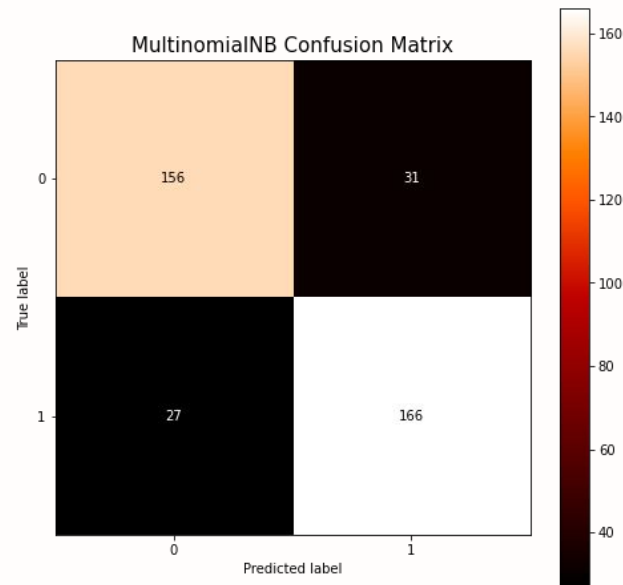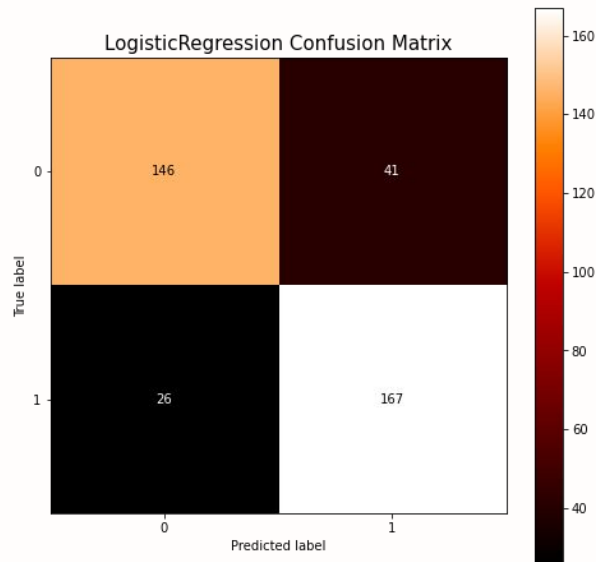
Key findings:
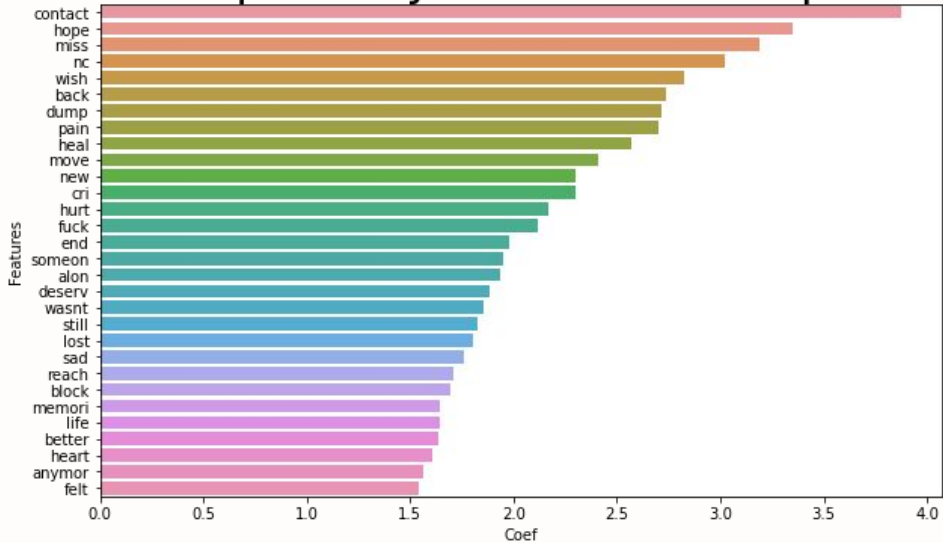# Relationship advice

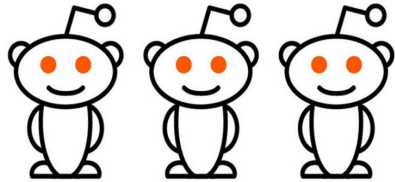

Top 30 keywords to Relationship Advice
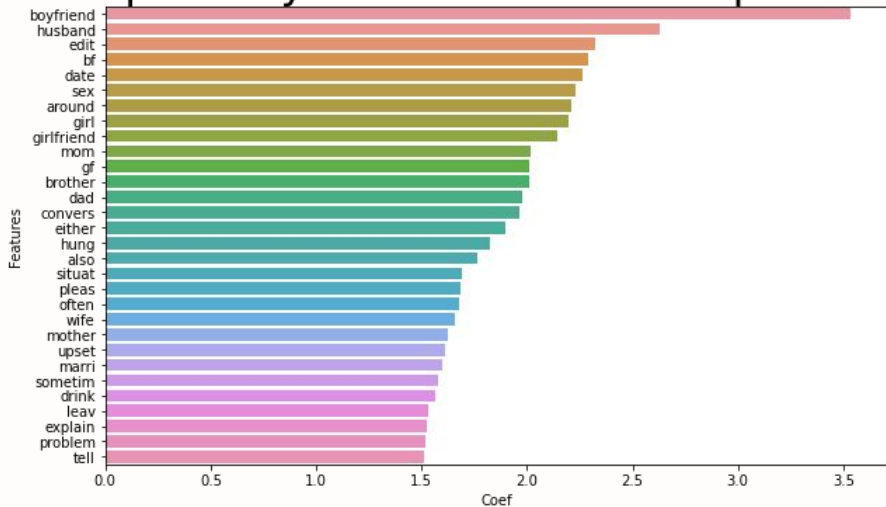
Key findings: Breakup

Top 30 Keywords to Breakups

Limitation and further research

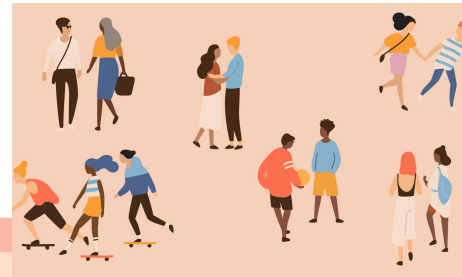Misclassification of user posted Subreddit

Limitation of hardware
Computational power

Webscraping from more platforms

More classification groups

Conlusion

- Webscrapping Subreddit
  - Relationship_advice
  - BreakUps

- Data Cleaning/ Exploration

- Natural Language Exploration
  - Stopword, Stemming, HTML, numerical values

- Modelling
  - Classification models

- Evaluation
  - Final model