Содержание

Постановка задачи

Имеется набор текстовых данных, состоящий из кликбейтных и некликбейтных заголовков статей [1].

Цель курсовой работы – решить задачу бинарной классификации заголовков статей.

Поставленные задачи:

1. Анализ датасета,
2. Подбор подходящей модели обучения,
3. Проверка полученной модели.

Для работы использвался Jupyter Notebook, Python версии 3.11.11. Используемые библиотеки:

- wordcloud matplotlib, seaborn,
- pandas, numpy, re
- nltk, sklearn, tensorflow, keras

Предобработка и анализ данных

При чтении заголовков из файлов сразу удалим пустые строки. Содержание датасета:

Кликбейтные статьи	15999
Некликбейтные статьи	16001

Данные сбалансированы.

Для предобработки используем функцию:

```
def preprocess(text):
    result = text.replace('/', '')
    result = re.sub(r'(\w)(\1{2,})', r'\1', result) # letters repetition
    result = ''.join(t for t in result if t not in punctuation)
    result = re.sub(r' +', ' ', result).lower().strip() # spaces, lowercase
    return result
```

Визуализируем данные: построим графики часто встречающихся слов для каждой категории (рис. 1 – рис. 2).

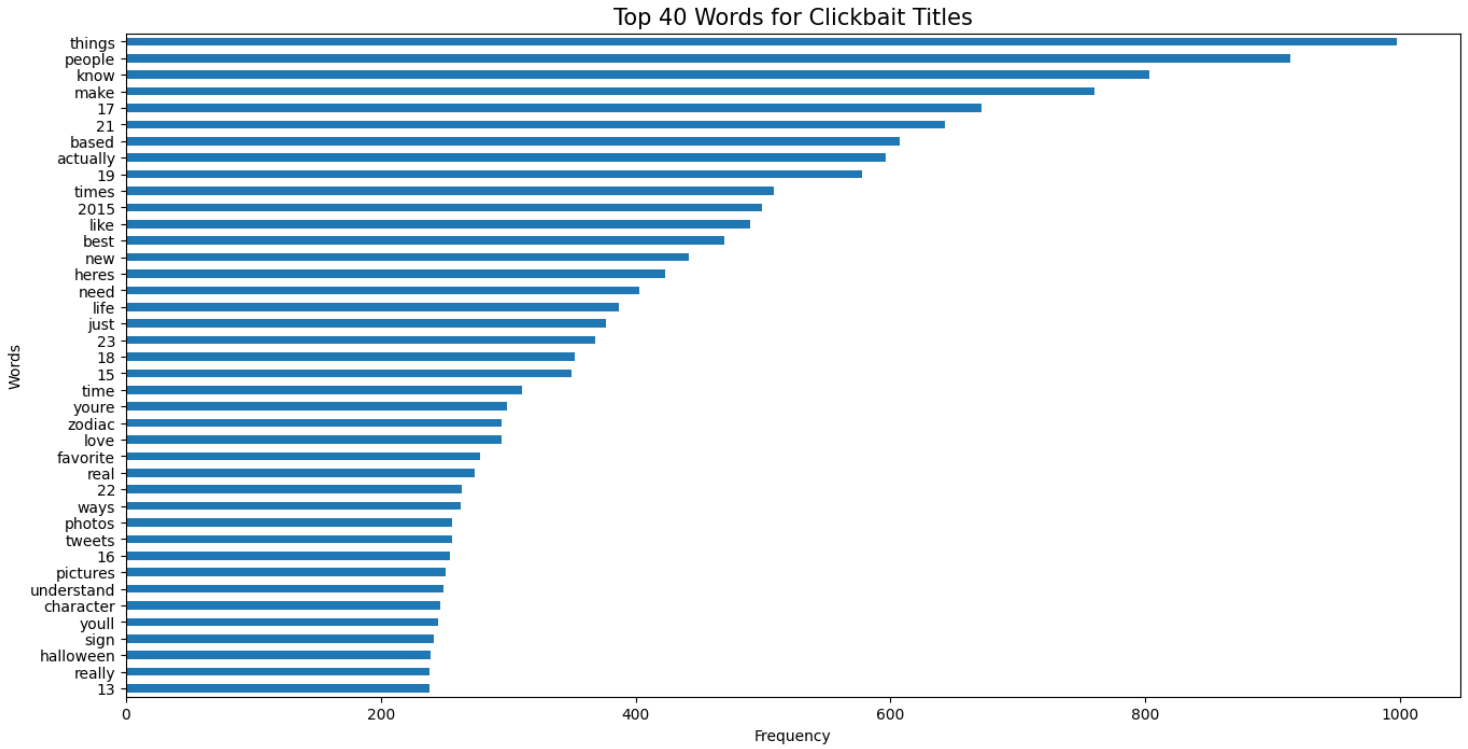

Рисунок 1. Часто встречающиеся слова в кликбейтных заголовках

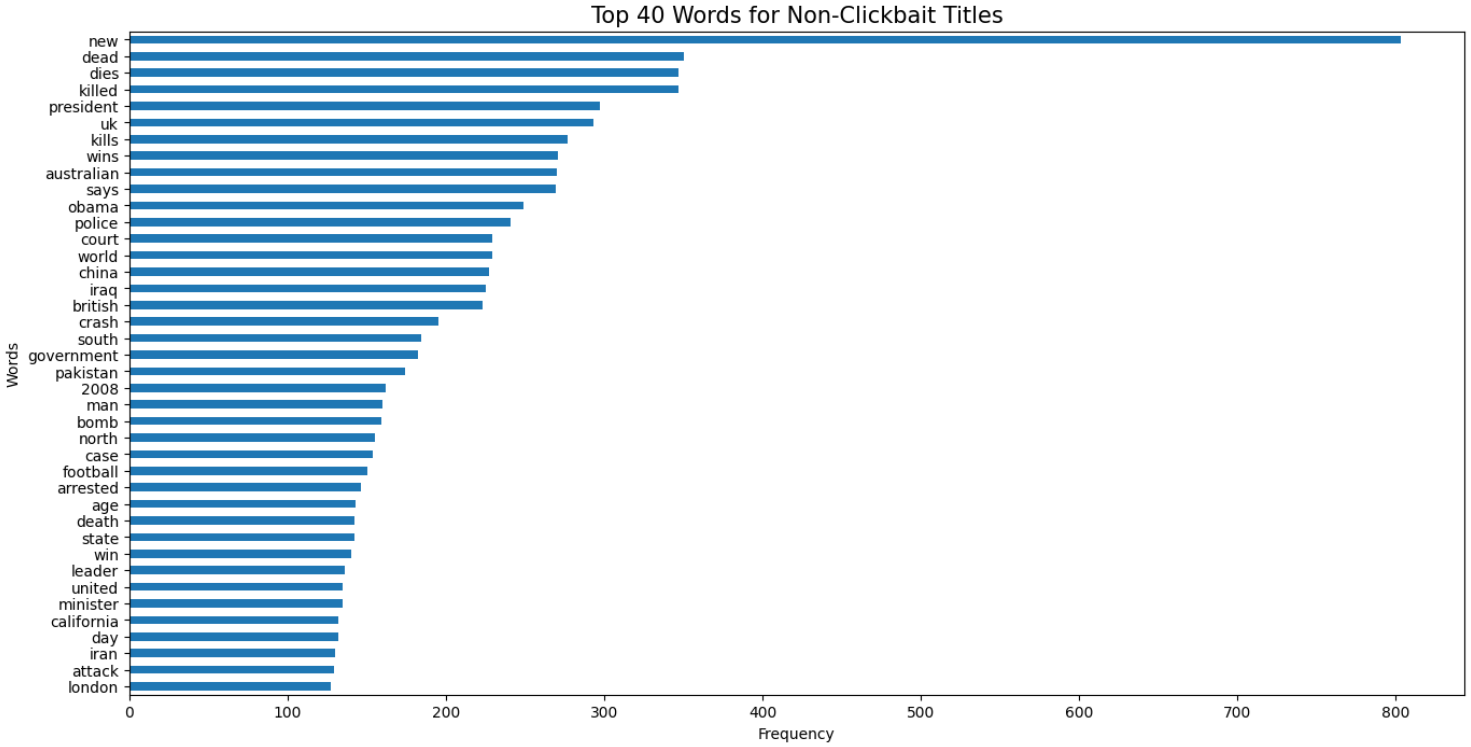

Рисунок 2. Часто встречающиеся слова в некликбейтных заголовках

Видим, что часто включаемые в некликбейтные заголовки слова в основном относятся к каким-либо политическим, мировым и новостям о происшествиях. В кликбейтных заголовках, наоборот, отражены более общие темы, часто используются цифры.

Посмотрим также на словосочетания (рис. 3).

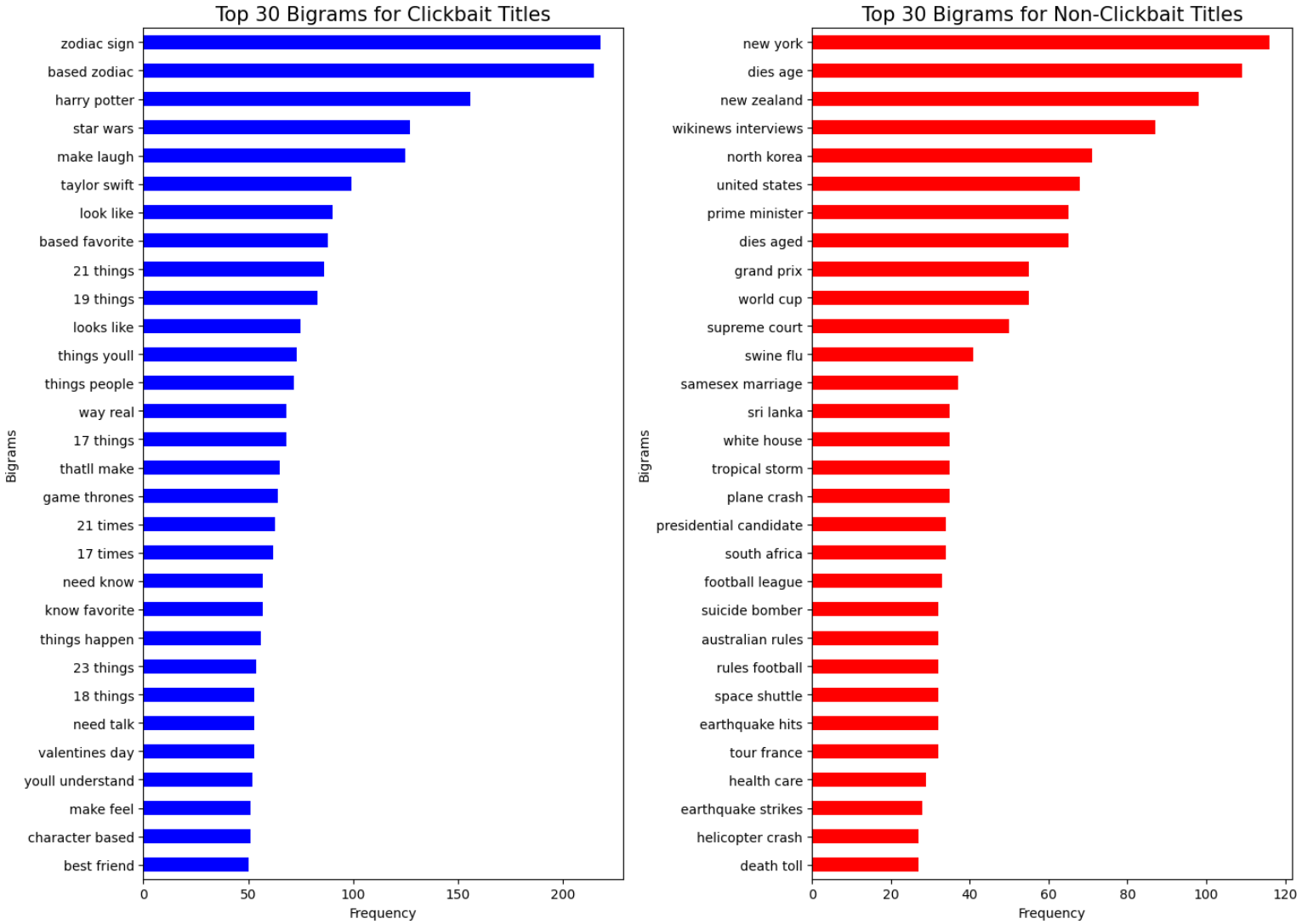

Рисунок 3. Часто встречающиеся в заголовках словосочетания

Попробуем выявить некоторые дополнительные признаки, по которым можно классифицировать заголовки.

Пометим заголовки, начинающиеся с цифры:

```
def starts_with_num(headline):
    if headline.startswith(('1','2','3','4','5','6','7','8','9')):
        return 1
    else:
        return 0

headlines['num_start'] = headlines['headline'].map(starts_with_num)
```

По графику (рис. 4) видим, что практически все заголовки, начинающиеся на цифру – кликбейт.

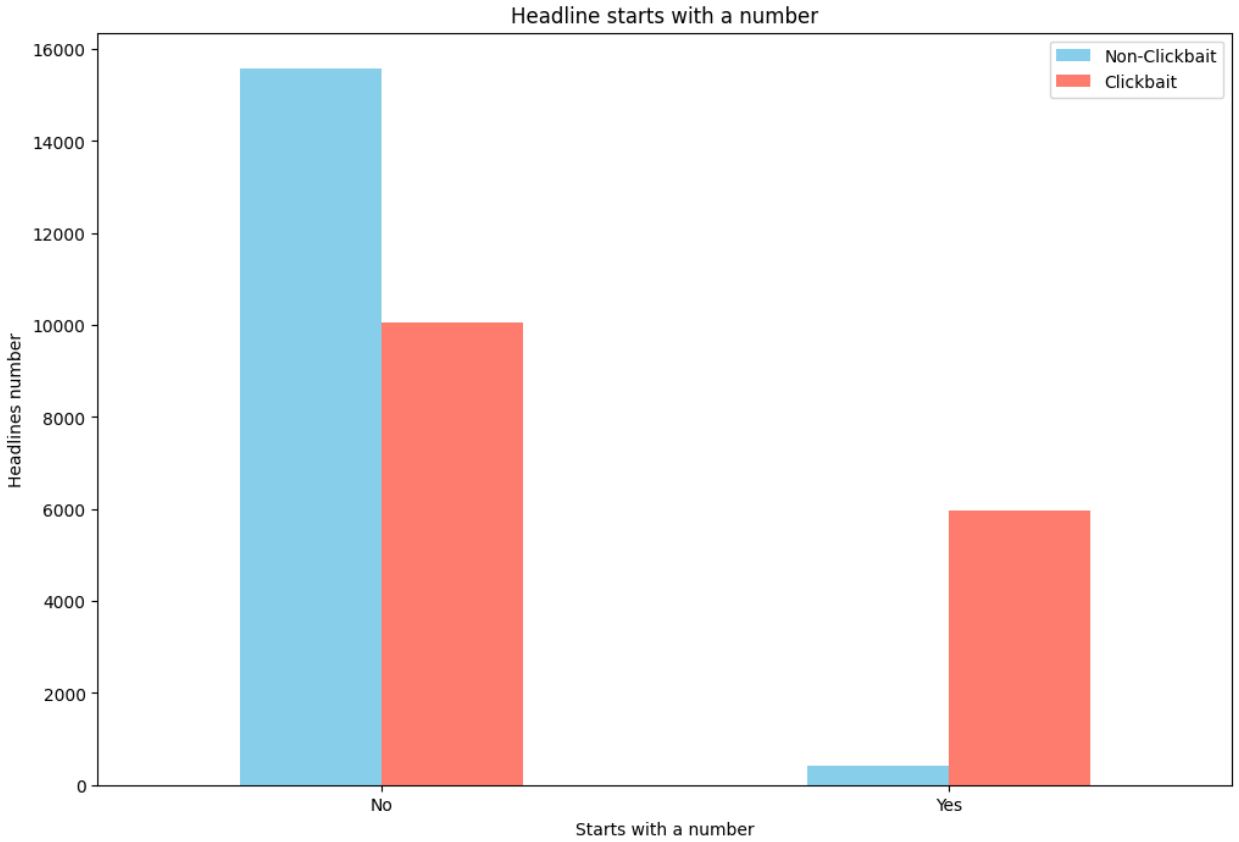

Рисунок 4. Признак "Начинается с цифры"

Вычислим длины заголовков.

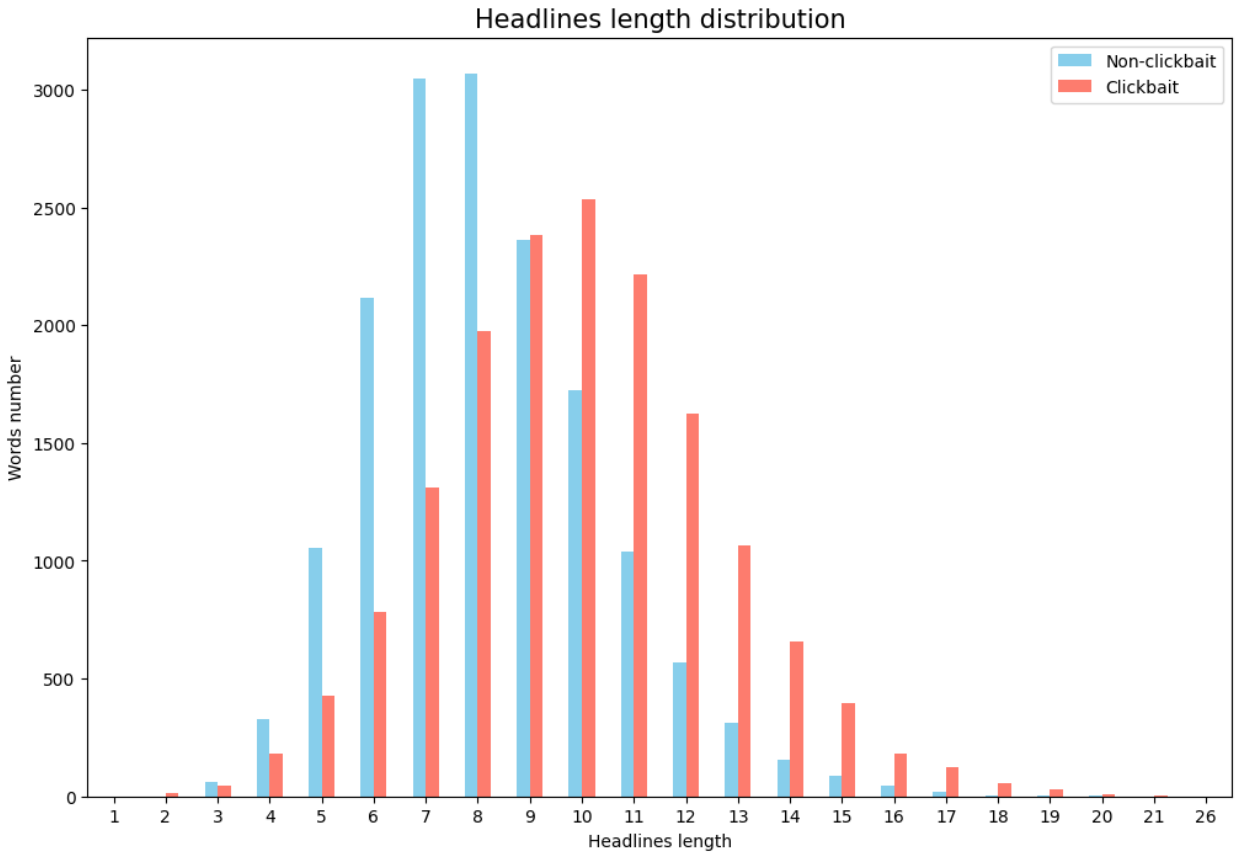

Рисунок 5. Распределение длин заголовков, признак "Количество слов"

Из графика (рис. 5) видно, что распределения длин заголовков разделимы и близки к нормальным.

Вычислим также количество встречающихся стоп-слов.

```
stopwords_list = stopwords.words('english')

def stopwords_num(text):
    split_text = text.split()
    return len([word for word in split_text if word in stopwords_list])
```

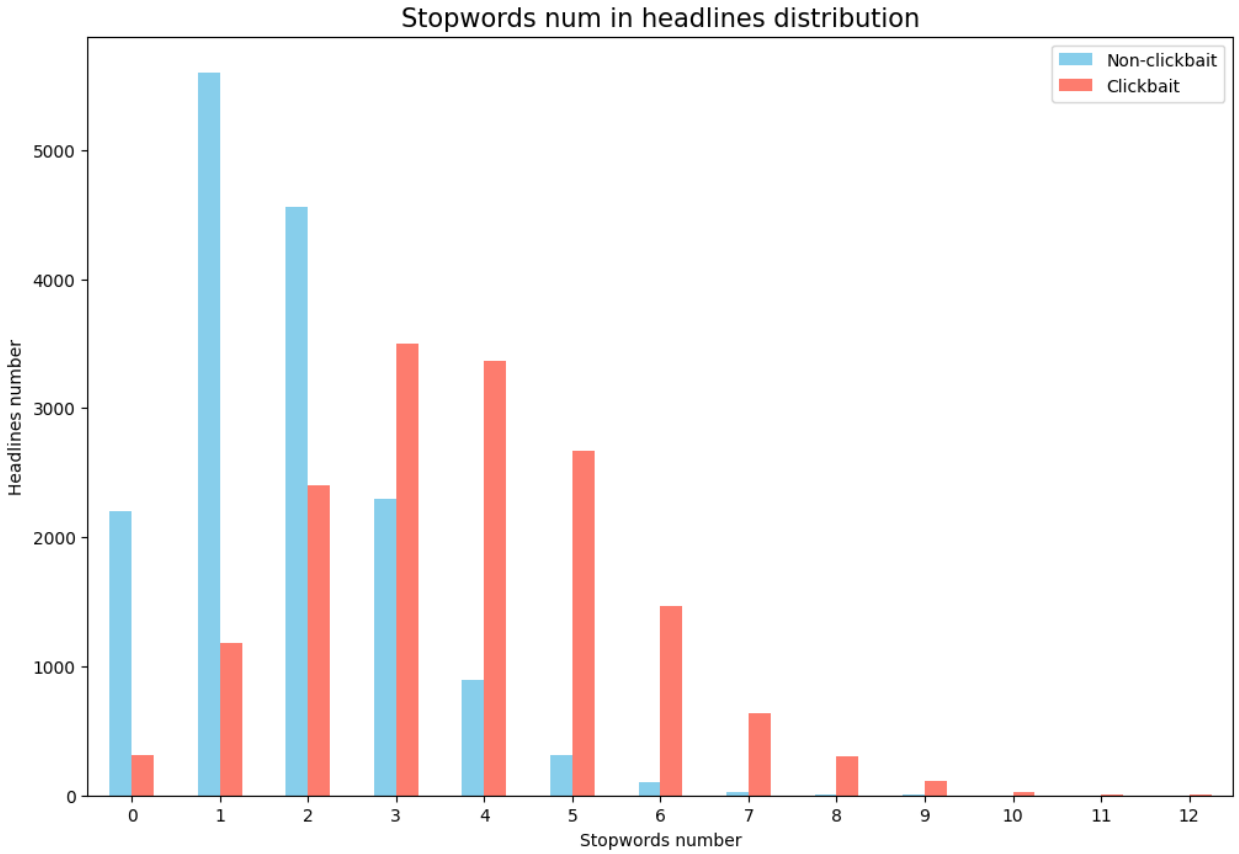

Рисунок 6. Распределение числа стоп-слов, признак "Количество стоп-слов"

Распределения (рис. 6) также разделимы и близки к нормальным. Заметим, что встречаемость стоп слов очень велика, но часто смысловой нагрузки они несут мало, поэтому попробуем удалить стоп-слова из данных.

Посмотрим на корреляцию добавленных численных признаков (рис. 7). Заметна достаточно высокая корреляция между числом стоп-слов, числа в начале заголовка и тем, является ли заголовок кликбейтом или нет.

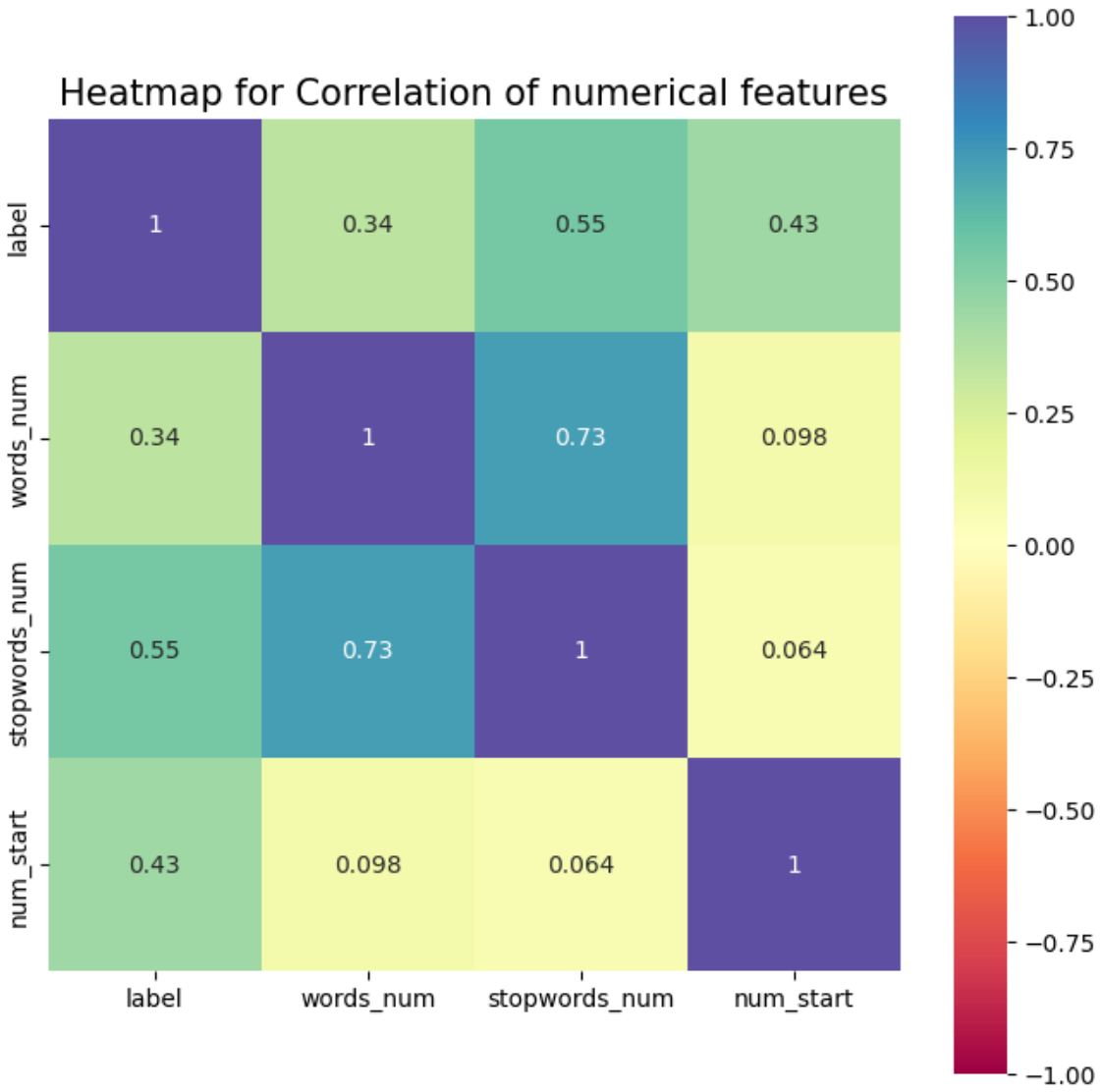

Рисунок 7. Корреляция численных признаков

Выбор модели и обучение

Многие слова встречаются как в кликбейтных, так и в некликбейтных заголовках. Это может быть проблемой при использовании классических методов машинного обучения (логистическая регрессия, наивный байесовский классификатор, SVC).

Для решения задачи выбрана модель LSTM (длинная цепь элементов краткосрочной памяти) — разновидность архитектуры рекуррентных нейронных сетей, способная изучать долгосрочные корреляции. LSTM нейросети очень хорошо работают над широким классом проблем и в настоящее время широко используются в задачах классификации текста.

Модель строится как последовательность слоев:

```
model = Sequential()
model.add(Input(shape=(MAX_LENGTH,)))
model.add(Embedding(VOCAB_SIZE, 32))
model.add(LSTM(32, return_sequences=True))
model.add(GlobalMaxPooling1D())
model.add(Dropout(0.2))
model.add(Dense(1, activation='sigmoid'))
model.summary()
```

Layer (type)	Output Shape	Param #
embedding_2 (Embedding)	(None, 26, 32)	396,352
lstm_2 (LSTM)	(None, 26, 32)	8,320
global_max_pooling1d_2 (GlobalMaxPooling1D)	(None, 32)	0
dropout_2 (Dropout)	(None, 32)	0
dense_2 (Dense)	(None, 1)	33

Total params: 404,705 (1.54 MB)

Trainable params: 404,705 (1.54 MB)

Non-trainable params: 0 (0.00 B)

Обучение производится со следующими параметрами:

```
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
history = model.fit(X_train, y_train, batch_size=512, validation_data=(X_test, y_test), epochs=20, callbacks=callbacks)
```

Объем тестовой выборки составляет 20% от всех данных.

Подготовка данных для обучения

Предварительно необходимо выполнить векторизацию текстовых данных – преобразовать каждый заголовок в последовательности целых чисел одинаковой длинны (равной максимальной длине заголовка).

Для этого используется класс токенизации текста Tokenizer:

```
tokenizer = Tokenizer(num_words=VOCAB_SIZE)
tokenizer.fit_on_texts(X.tolist())
total_unique_words = len(tokenizer.word_counts)
VOCAB_SIZE = round(total_unique_words * 0.5) # 40%
```

В результате удаления стоп-слов из словаря было удалено 131 слово. Будем использовать 40% словаря.

Обучение

Рассмотрим 4 сценария:

1. Стоп-слова не убраны, дополнительные признаки не используются.
2. Стоп слова не убраны, используются дополнительные признаки.
3. Стоп слова убраны, дополнительные признаки не используются.
4. Стоп слова убраны, используются дополнительные признаки.

Результаты

Результаты работы модели на тестовой выборке представлены матрицами ошибок, а также показателями Recall, Precision (рис. 8 – рис. 11).

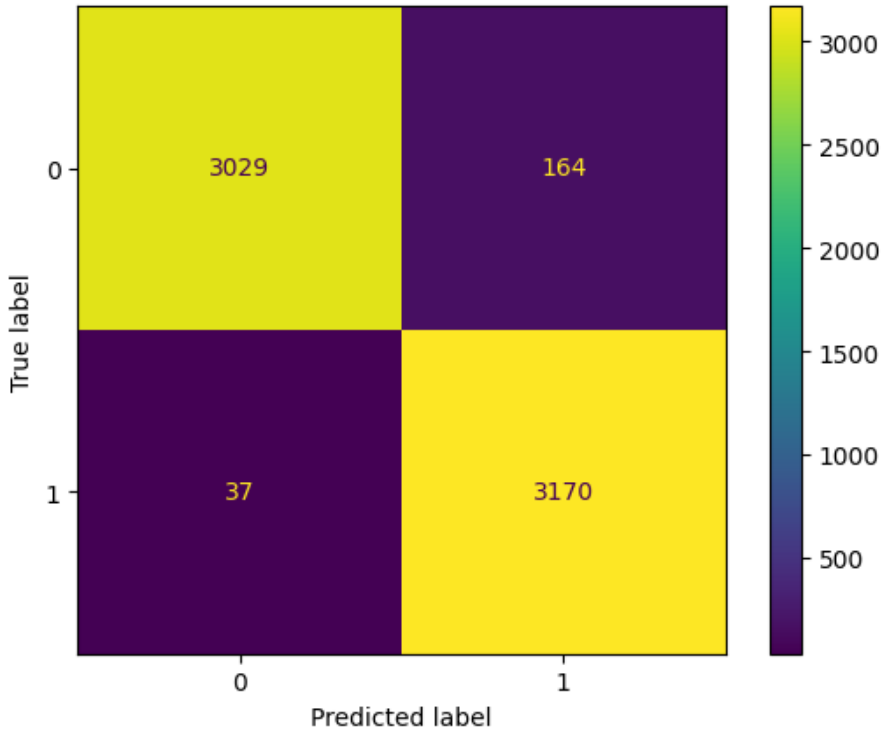

Рисунок 8 Матрица ошибок для первого сценария

Recall of the model is 0.99

Precision of the model is 0.95

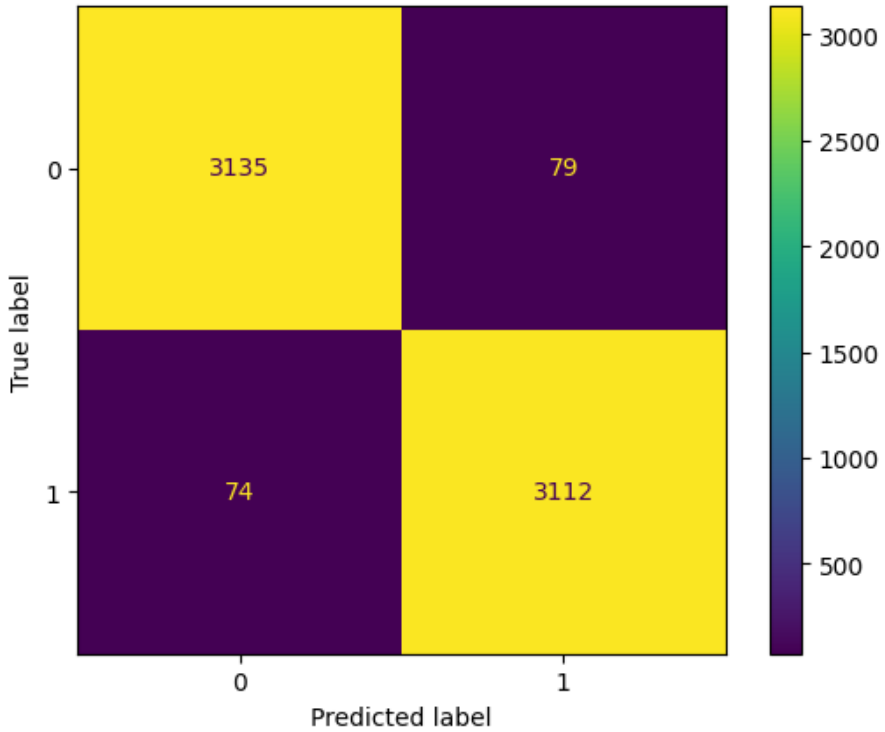

Рисунок 9. Матрица ошибок для второго сценария

Recall of the model is 0.98

Precision of the model is 0.97

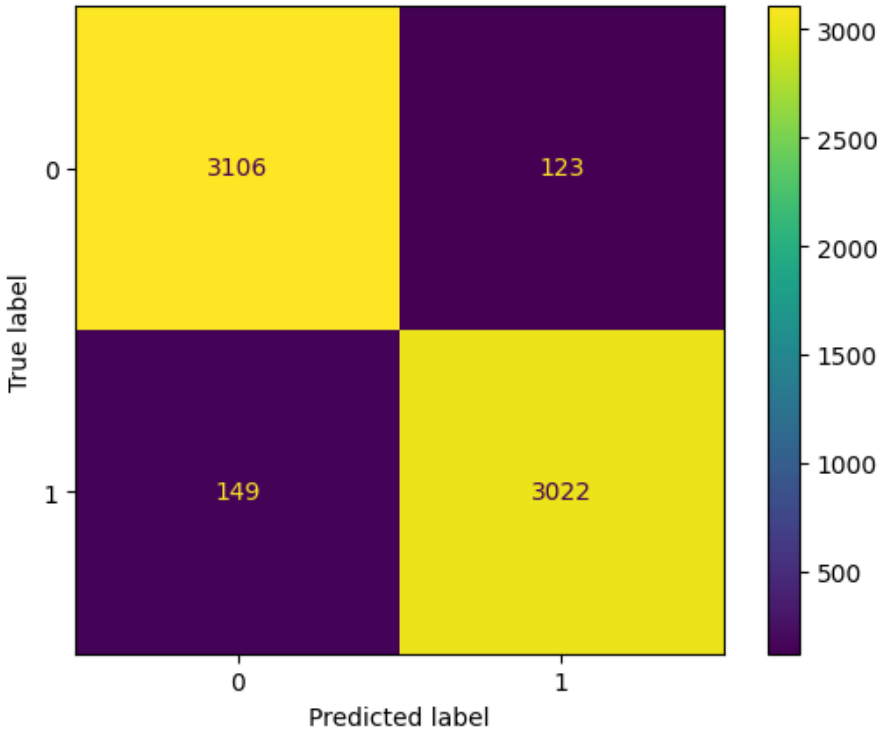

Рисунок 10. Матрица ошибок для третьего сценария

Recall of the model is 0.95

Precision of the model is 0.96

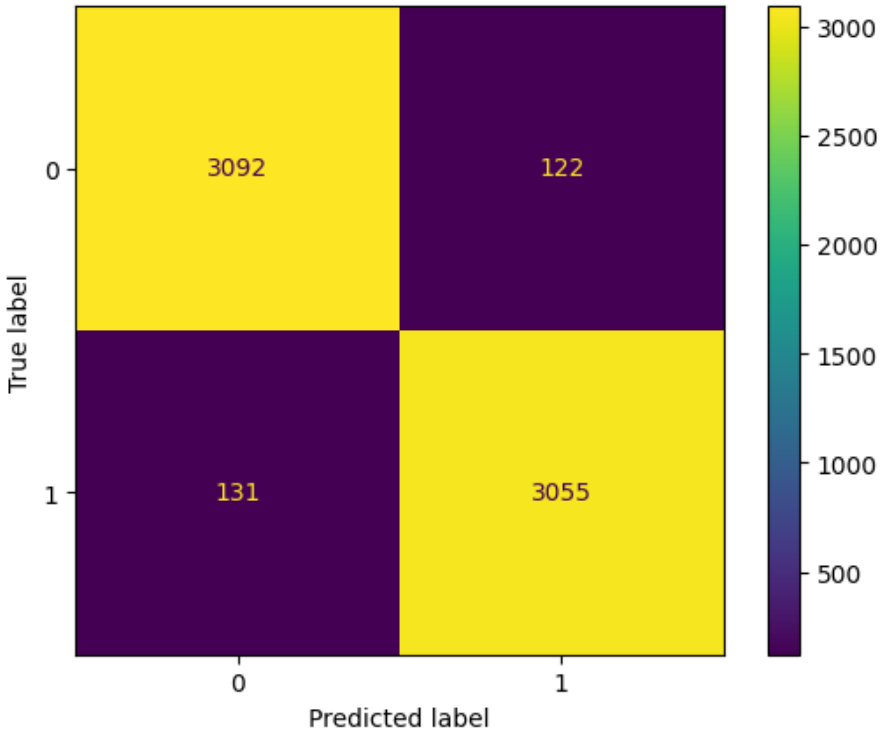

Рисунок 11. Матрица ошибок для четвертого сценария

Recall of the model is 0.96

Precision of the model is 0.96

В результате модель первого сценария имеет наилучшую способность распознавания кликбейтных заголовков (recall) – 0.99. Благодаря учету дополнительно введенных признаков наилучшей точностью (precision) обладает вторая модель – 0.97. Удаление стоп-слов не дало улучшения, так как число стоп-слов в заголовке коррелирует с меткой класса. Даже включение в рассмотрение признака о количестве стоп-слов не помогло.

Заключение

В ходе работы был проанализирован набор данных, получены модели LSTM для бинарной классификации заголовков статей. По результатам работы моделей на тестовых данных можно сказать, что они справляются с классификацией хорошо.

Список источников

1. https://github.com/bhargaviparanjape/clickbait/tree/master/dataset