Анализ эмоциональной окраски текста

на примере решения двух задач:

1) "Hate Speech and Offensive Language Dataset

2) "Russian Troll Tweets

Николай Павлов
группа DS-27

Содержание

Постановка задач

1

Постановка задач

Применение методов обработки текста для анализа тональности твитов в качестве задачи классификации, на примере двух задач:

1

"Hate Speech and Offensive Language Dataset - research hate-speech detection" (перевод с англ. яз. «Набор данных о высказываниях с ненавистью и оскорбительном языке — исследование обнаружения оскорбительных высказываний») - *в дальнейшем сокращённо **HATE***

2

"Russian Troll Tweets - 3 million tweets from accounts associated with the 'Internet Research Agency'" («Твиты русских троллей — 3 миллиона твитов с аккаунтов, связанных с «Агентством интернет-исследований») - *в дальнейшем сокращённо **TROLL***

Ход решения

2

Ход решения

1. **Анализ данных:**
   - ➔ Выявление особенностей
   - ➔ Обработка

2. **Пре-моделирование:**
   - ➔ Алгоритмы векторизации
   - ➔ Алгоритмы моделирования
   - ➔ Метрики качества

3. **Моделирование:**
   - ➔ Обучение
   - ➔ Настройка гиперпараметров

4. **Результаты:**
   - ➔ Отчёт классификации
   - ➔ Возвращение к п.2-3
   - ➔ Советы
   - ➔ Чтение дополнительных источников

5. **Итог:**
   - ➔ Рабочая модель

Особенности каждой задачи

3

Особенности каждой задачи

Датасеты

Задача 1 - HATE

	count	hate_speech	offensive_language	neither	class	tweet
0	3	0	0	3	2	!!! RT @mayasolovely: As a woman you shouldn't...
1	3	0	3	0	1	!!!!! RT @mleew17: boy dats cold...tyga dwn ba...
2	3	0	3	0	1	!!!!!!! RT @UrKindOfBrand Dawg!!!! RT @80sbaby...
3	3	0	2	1	1	!!!!!!!!! RT @C_G_Anderson: @viva_based she lo...
4	6	0	6	0	1	!!!!!!!!!!!!! RT @ShenikaRoberts: The shit you...

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 24783 entries, 0 to 25296
Data columns (total 6 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   count               24783 non-null  int64
 1   hate_speech         24783 non-null  int64
 2   offensive_language  24783 non-null  int64
 3   neither             24783 non-null  int64
 4   class               24783 non-null  int64
 5   tweet               24783 non-null  object
dtypes: int64(5), object(1)
memory usage: 1.3+ MB
```

Задача 2 - TROLL

	external_author_id	author	content	region	language	publish_date	harvested_date	fol
0	2.385425e+09	MARRINABEREZKA	Обама принял решение по санкциям против Ирана ...	United States	Russian	11/11/2015 6:33	11/11/2015 6:34	26
1	2.534361e+09	ANETTANOVGOROD	Встреча Лаврова и Керри стартовала в Нью-Йорке...	Azerbaijan	Russian	9/27/2015 15:11	9/27/2015 15:11	16
2	1.612107e+09	LILJORDAMN	#IndieAdvancement Slim The Phenom @therealslim...	United States	English	12/3/2016 22:36	12/3/2016 22:36	60
3	3.254274e+09	FINDDIET	'@ozzycaceres ozzy @laurengodfreyx1 Lauren @dj...	United States	English	8/5/2015 17:39	8/5/2015 17:39	3
4	1.647457e+09	COLINSNEVERLAND	This, BTW is why I don't instantly dismiss the...	United States	English	1/6/2016 18:02	1/6/2016 18:02	36

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2973371 entries, 0 to 2973370
Data columns (total 15 columns):
 #   Column               Dtype
---  ------               -----
 0   external_author_id   float64
 1   author               object
 2   content              object
 3   region               object
 4   language             object
 5   publish_date         object
 6   harvested_date       object
 7   following            int64
 8   followers            int64
 9   updates              int64
 10  post_type            object
 11  account_type         object
 12  new_june_2018        int64
 13  retweet              int64
 14  account_category     object
dtypes: float64(1), int64(5), object(9)
memory usage: 340.3+ MB
```

harvested_date	following	followers	updates	post_type	account_type	new_june_2018	retweet	account_category
11/11/2015 6:34	266	314	4160	RETWEET	Russian	1	1	NonEnglish
9/27/2015 15:11	166	153	1900	RETWEET	Russian	1	1	NonEnglish
12/3/2016 22:36	602	706	2531	RETWEET	left	0	1	LeftTroll
8/5/2015 17:39	3	200	21960	NaN	Commercial	1	0	Commercial
1/6/2016 18:02	364	202	127	RETWEET	Right	0	1	RightTroll

Особенности каждой задачи

Анализ

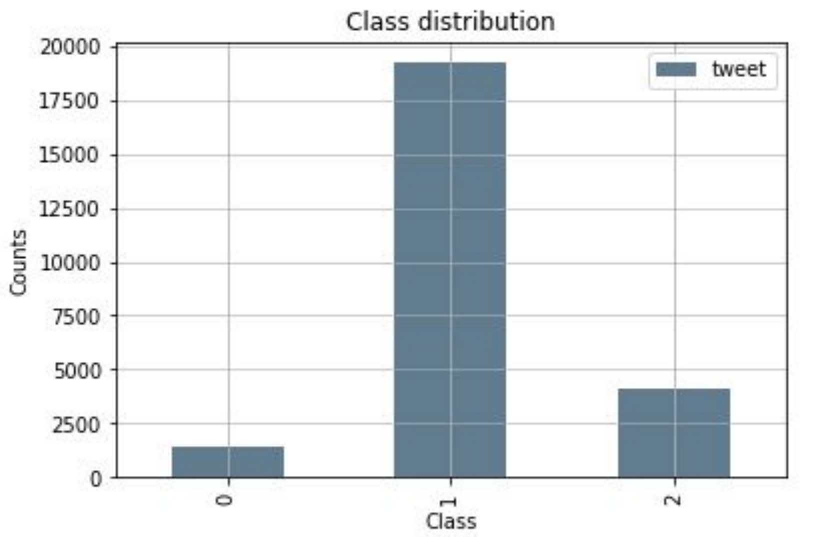
Задача 1 - HATE

```
df_1.tweet.iloc[0]
```

```
"!!! RT @mayasolovely: As a woman you shouldn't complain about cleaning up your house. &amp; as a man you should always take th
e trash out..."
```

```
df_1[['class', 'tweet']].groupby('class').count()
```

	tweet
class	
0	1430
1	19190
2	4163



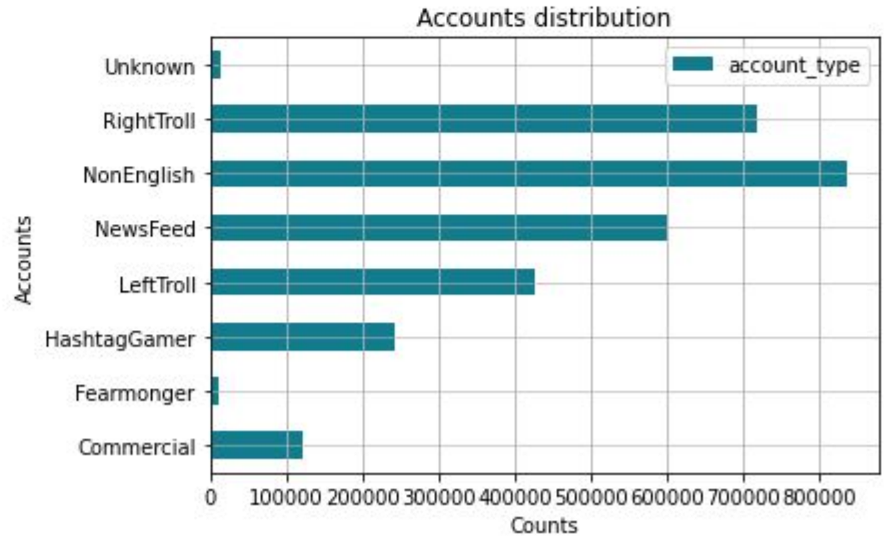
Задача 2 - TROLL

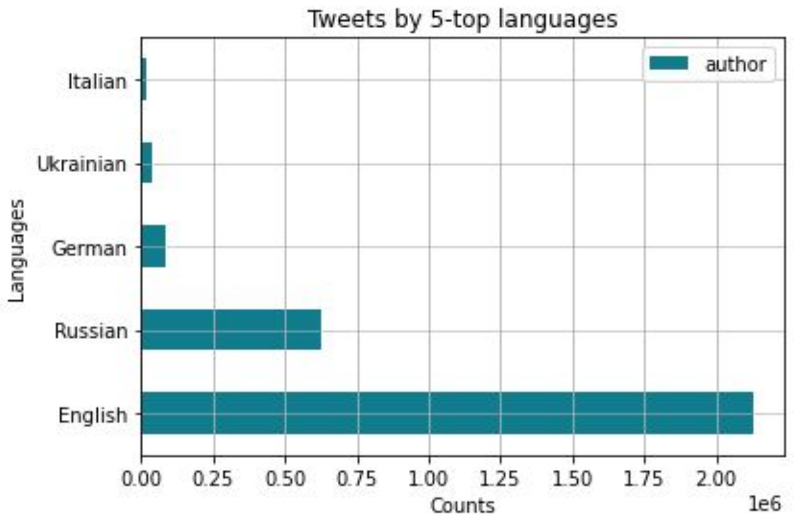
```
data_total[data_total['account_category'] == 'Commercial']['content'].iloc[0]
```

```
"'@ozzycaceres ozzy @laurengodfreyx1 Lauren @djhawes Danners @karleighwoelmer Karleigh h
ttp://t.co/QeKnVmkfxw https://t.co/ae3ItmkKin'"
```

```
data_total.groupby('account_category').count()[['author']]
```

	author
account_category	
Commercial	122582
Fearmonger	11140
HashtagGamer	241827
LeftTroll	427811
NewsFeed	599294
NonEnglish	837725
RightTroll	719087
Unknown	13905





1. Векторизация
2. Регистр
3. Ники пользователей
4. Нехорошие ссылки
5. Слова с цифрами
6. Пунктуация
7. Стоп-слова
8. Лемматизация
9. Множественные пробелы

Используемые библиотеки, алгоритмы, метрики

4

Используемые библиотеки, алгоритмы, метрики
Библиотеки

Задача 1 - HATE

Загрузка данных
- os

Анализ
- numpy
- pandas
- matplotlib.pyplot
- tqdm

Предобработка
- re
- string.punctuation
- nltk.corpus.stopwords
- spacy
- sklearn.feature_extraction.text.CountVectorizer
- sklearn.feature_extraction.text.TfidfVectorizer
- nltk.word_tokenize
- imblearn.over_sampling.SMOTE
- sklearn.preprocessing.LabelEncoder

Метрики
- sklearn.metrics.classification_report

Задача 2 - TROLL

Моделирование
- sklearn.model_selection.train_test_split
- sklearn.pipeline.Pipeline
- xgboost.XGBClassifier
- sklearn.linear_model.LogisticRegression
- catboost.CatBoostClassifier
- sklearn.svm.SVC
- transformers
- torch
- tensorflow.keras.models.Model
- tensorflow.keras.layers (LSTM, Activation, Dense, Dropout, Input, Embedding, SpatialDropout1D, Flatten)
- tensorflow.keras.optimizers.Adam
- tensorflow.keras.preprocessing.text.Tokenizer
- tensorflow.keras.preprocessing.sequence
- tensorflow.keras.utils.to_categorical
- tensorflow.keras.callbacks.EarlyStopping
- tensorflow.keras.models.Sequential
- tensorflow.keras.callbacks.EarlyStopping, ModelCheckpoint

Используемые библиотеки, алгоритмы, метрики

Функции

Задача 1 - HATE

```python
def clean_text(text, lemma, noise_words):
    '''функция на вход получает текст, на выходе выдаёт очищенный текст'''
    text = str(text).lower() # первый шаг - все тексты приводим к нижнему регистру

    text = re.sub("@[\w'._+-:]+", '', text) # второй шаг - убираем ники пользователей твитера, т.к. обычно не несут никакой окраски
    text = re.sub('https?://\S+|www\.\S+', '', text) # третий шаг - убираем ссылки в твитах, т.к. названия ссылок  обычно не влияю
на тональность
    text = re.sub('\w*\d\w*', '', text) # четвёртый шаг - убираем "слова", внутри которых есть цифры

    text = re.sub('[^\w\s^.]','', text) # пятый шаг - убираем знаки пунктуации
    text = re.sub('[_\.]+',' ', text)

    text = " ".join([word for word in text.split(' ') if word not in noise_words]) # шестой шаг - отбираем только НЕстоп-слова

    text = " ".join([word.lemma_ for word in lemma(text)]) # седьмой шаг - лемматизация при помощи spacy
    text = re.sub('[\s]+', ' ', text) # восьмой шаг - заменяем любой пробельный символ(табуляция, конец строки и т.п.) на пробел

    return text
```

```python
def clean_text_for_BERT(text):
    '''функция на вход получает текст, на выходе выдаёт очищенный текст для BERT'''

    text = re.sub("@[\w'._+-:]+", 'USER_NAME_TAG', text) # второй шаг - убираем ники пользователей твитера, т.к. обычно не несут н
икакой окраски
    text = re.sub('https?://\S+|www\.\S+', 'URL_TAG', text) # третий шаг - убираем ссылки в твитах, т.к. названия ссылок  обычно н
е влияют на тональность

    return text
```

Задача 2 - TROLL

```python
m = pymorphy2.MorphAnalyzer()
def lemmatize(text, mystem=m):
    try:
        return ' '.join((m.parse(t)[0].normal_form for t in text.split(' ')))
    except:
        return " "
```

```python
def clean_text_2(text, lemmatize, noise_wrods_2):
    '''функция на вход получает текст, на выходе выдаёт очищенный текст'''
    text = str(text).lower() # первый шаг - все тексты приводим к нижнему регистру

    text = re.sub("@[\w'._+-:]+", '', text) # второй шаг - убираем ники пользователей твитера,
т.к. обычно не несут никакой окраски
    text = re.sub('https?://\S+|www\.\S+', '', text) # третий шаг - убираем ссылки в твитах,
т.к. названия ссылок  обычно не влияют на тональность
    text = re.sub('\w*\d\w*', '', text) # четвёртый шаг - убираем "слова", внутри которых есть
цифры

    text = re.sub('[^\w\s^.]','', text) # пятый шаг - убираем знаки пунктуации
    text = re.sub('[_.]+',' ', text)

  #стоп-слова  не всех языков
    text = " ".join([word for word in text.split(' ') if word not in noise_wrods_2]) # шесто
й шаг - отбираем только НЕстоп-слова

    text = lemmatize(text) # седьмой шаг - лемматизация при помощи spacy
    text = re.sub('[\s]+', ' ', text) # восьмой шаг - заменяем любой пробельный символ(табуляци
я, конец строки и т.п.) на пробел

    return text
```

Используемые библиотеки, алгоритмы, метрики

Алгоритмы

Задача 1 - HATE

Задача 2 - TROLL

Векторизация

- ➢ CountVectorizer
- ➢ TfidfVectorizer

Моделирование

- LogisticRegression
- SVC
- XGBClassifier
- CatBoostClassifier
- tensorflow.keras.models.Model(LSTM)
- transformers (Bert) + LogisticRegression
- transformers (Bert) + SVC

Используемые библиотеки, алгоритмы, метрики
Метрики

Задача 1 - HATE

Задача 2 - TROLL

Метрики

➢ sklearn.metrics.classification_report

1. **Recall**
2. **Precision**
3. **F1-score**
4. **Accuracy**

Итоги решений

5

Итоги решений

Задача 1 - HATE

CountVectorizer + LogisticRegression

```
vec = CountVectorizer(ngram_range=(1, 1))
vec.fit(df_2['tweet'])
bow = vec.transform(X_train)
```

```
clf = LogisticRegression(random_state=42, solver='liblinear', class_weight = 'balanced')
clf.fit(bow, y_train)
pred = clf.predict(vec.transform(X_test))
print(classification_report(pred, y_test))
```

```
              precision    recall  f1-score   support

           0       0.60      0.68      0.64       247
           1       0.87      0.88      0.88       816
           2       0.96      0.90      0.92       856

    accuracy                           0.86      1919
   macro avg       0.81      0.82      0.81      1919
weighted avg       0.87      0.86      0.87      1919
```

CountVectorizer + LogisticRegression

```
pipe3 = Pipeline([
                ('CountVectChar', CountVectorizer(analyzer='char', ngram_range=(1, 7))),
                ('LogReg', LogisticRegression(random_state=42, solver='liblinear',
                                               class_weight = 'balanced'))
                ])
pipe3.fit(X_train, y_train)
y_pred3 = pipe3.predict(X_test)
print(classification_report(y_pred3, y_test))
```

```
              precision    recall  f1-score   support

           0       0.62      0.64      0.63       265
           1       0.82      0.87      0.85       743
           2       0.96      0.90      0.93       911

    accuracy                           0.85      1919
   macro avg       0.80      0.80      0.80      1919
weighted avg       0.86      0.85      0.85      1919
```

TfidfVectorizer + LogisticRegression

```
pipe = Pipeline([
                ('tf-idf', TfidfVectorizer()),
                ('LogReg', LogisticRegression(random_state=42,
                                              solver='liblinear',
                                              class_weight = 'balanced'))
                ])
pipe.fit(X_train, y_train)
y_pred = pipe.predict(X_test)
print(classification_report(y_pred, y_test))
```

```
              precision    recall  f1-score   support

           0       0.63      0.68      0.65       253
           1       0.83      0.90      0.87       724
           2       0.97      0.88      0.92       942

    accuracy                           0.86      1919
   macro avg       0.81      0.82      0.81      1919
weighted avg       0.87      0.86      0.86      1919
```

CountVectorizer + LogisticRegression

```
pipe4 = Pipeline([
                ('CountVectChar', CountVectorizer(analyzer='char', ngram_range=(1, 9))),
                ('LogReg', LogisticRegression(random_state=42, solver='liblinear',
                                               class_weight = 'balanced'))
                ])
pipe4.fit(X_train, y_train)
y_pred4 = pipe4.predict(X_test)
print(classification_report(y_pred4, y_test))
```

```
              precision    recall  f1-score   support

           0       0.61      0.66      0.63       256
           1       0.83      0.87      0.85       750
           2       0.96      0.90      0.93       913

    accuracy                           0.86      1919
   macro avg       0.80      0.81      0.80      1919
weighted avg       0.86      0.86      0.86      1919
```

Итоги решений

Задача 1 - HATE

CountVectorizer + XGBClassifier

```
pipe5 = Pipeline([
            ('CountVectChar', CountVectorizer(tokenizer=word_tokenize,
                                              ngram_range=(1, 1))),
            ('XGB', XGBClassifier(objective = 'multi:softprob' ,
                                  use_label_encoder=False,
                                  eval_metric='mlogloss'))
            ])
pipe5.fit(X_train, y_train)
y_pred5 = pipe5.predict(X_test)
print(classification_report(y_pred5, y_test))
```

```
              precision    recall  f1-score   support

           0       0.57      0.74      0.64       210
           1       0.87      0.88      0.87       785
           2       0.97      0.90      0.93       924

    accuracy                           0.87      1919
   macro avg       0.80      0.84      0.82      1919
weighted avg       0.89      0.87      0.88      1919
```

CountVectorizer + CatBoostClassifier

```
pipe7 = Pipeline([
            ('CountVectChar', CountVectorizer(ngram_range=(1, 1))),
            ('CBC', CatBoostClassifier(learning_rate=0.6, depth=4,
                                       loss_function='MultiClass'))
            ])
pipe7.fit(X_train, y_train)
y_pred7 = pipe7.predict(X_test)
print(classification_report(y_pred7, y_test))
```

```
              precision    recall  f1-score   support

           0       0.55      0.75      0.64       210
           1       0.89      0.89      0.89       809
           2       0.98      0.89      0.93       900

    accuracy                           0.88      1919
   macro avg       0.81      0.84      0.82      1919
weighted avg       0.89      0.88      0.88      1919
```

TfidfVectorizer + XGBClassifier

```
pipe6 = Pipeline([
            ('tf-idf', TfidfVectorizer()),
            ('XGB', XGBClassifier(booster='gbtree', objective = 'multi:softprob' ,
                                  use_label_encoder=False,eval_metric='mlogloss'))
            ])
pipe6.fit(X_train, y_train)
y_pred6 = pipe6.predict(X_test)
print(classification_report(y_pred6, y_test))
```

```
              precision    recall  f1-score   support

           0       0.57      0.71      0.63       222
           1       0.87      0.88      0.87       781
           2       0.96      0.90      0.93       916

    accuracy                           0.87      1919
   macro avg       0.80      0.83      0.81      1919
weighted avg       0.88      0.87      0.87      1919
```

TfidfVectorizer + CatBoostClassifier

```
pipe8 = Pipeline([
            ('tf-idf', TfidfVectorizer(tokenizer=word_tokenize)),
            ('CBC', CatBoostClassifier(learning_rate=0.6, depth=4,
                                       loss_function='MultiClass'))
            ])
pipe8.fit(X_train, y_train)
y_pred8 = pipe8.predict(X_test)
print(classification_report(y_pred8, y_test))
```

```
              precision    recall  f1-score   support

           0       0.52      0.74      0.61       197
           1       0.88      0.87      0.88       824
           2       0.97      0.89      0.93       898

    accuracy                           0.87      1919
   macro avg       0.79      0.83      0.80      1919
weighted avg       0.89      0.87      0.87      1919
```

Итоги решений

Задача 1 - HATE

TfidfVectorizer + LGBMClassifier

```
pipe0 = Pipeline([
            ('tf-idf', TfidfVectorizer(tokenizer=word_tokenize)),
            ('LGMClass', LGBMClassifier())
            ])
pipe0.fit(X_train, y_train)
y_pred0 = pipe0.predict(X_test)
print(classification_report(y_pred0, y_test))

              precision    recall  f1-score   support

           0       0.49      0.72      0.58       191
           1       0.90      0.87      0.88       868
           2       0.95      0.89      0.92       860

    accuracy                           0.86      1919
   macro avg       0.78      0.83      0.80      1919
weighted avg       0.88      0.86      0.87      1919
```

TfidfVectorizer + LightGBM + SMOTE

```
vec_10_1 = TfidfVectorizer()
vec_10_1.fit(df_2['tweet'])
bow_10_1 = vec_10_1.transform(X_train)

sm_1 = SMOTE (#sampling_strategy = 0.9,
        random_state=0,
        k_neighbors=25)
X_train_res_1, y_train_res_1 = sm_1.fit_resample(bow_10_1, y_train)

pipe0_1 = LGBMClassifier()

pipe0_1.fit(X_train_res_1, y_train_res_1)
y_pred0_1 = pipe0_1.predict(vec_10_1.transform(X_test))
print(classification_report(y_pred0_1, y_test))

              precision    recall  f1-score   support

           0       0.60      0.66      0.63       276
           1       0.84      0.87      0.85       758
           2       0.96      0.91      0.94       885

    accuracy                           0.86      1919
   macro avg       0.80      0.81      0.81      1919
weighted avg       0.86      0.86      0.86      1919
```

tf.keras + Tokenizer + LSTM

```
max_words = 50000
max_len = 300
tokenizer = Tokenizer(num_words=max_words)
tokenizer.fit_on_texts(X_train)
sequences = tokenizer.texts_to_sequences(X_train)
sequences_matrix = sequence.pad_sequences(sequences,maxlen=max_len)

model_9 = Sequential()
model_9.add(Embedding(max_words, 100, input_length=max_len))
model_9.add(SpatialDropout1D(0.2))
model_9.add(LSTM(100, dropout=0.2, recurrent_dropout=0.2))
model_9.add(Dense(1, activation='relu'))
model_9.add(Dense(1, activation='sigmoid'))
model_9.summary()
model_9.compile(loss='categorical_crossentropy', optimizer=Adam(), metrics=['accuracy'])
stop = EarlyStopping(
    monitor='val_accuracy',
    mode='max',
    patience=3
)

checkpoint = ModelCheckpoint(
    filepath='./',
    save_weights_only=True,
    monitor='val_accuracy',
    mode='max',
    save_best_only=True)
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 300, 100)          5000000
_________________________________________________________________
spatial_dropout1d (SpatialDr (None, 300, 100)          0
_________________________________________________________________
lstm (LSTM)                  (None, 100)               80400
_________________________________________________________________
dense (Dense)                (None, 1)                 101
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 2
=================================================================
Total params: 5,080,503
Trainable params: 5,080,503
Non-trainable params: 0
_________________________________________________________________
```

```
history = model_9.fit(sequences_matrix, y_train,batch_size=1024, epochs=10,
          validation_split=0.2, callbacks=[stop, checkpoint])

Epoch 1/10
16/16 [==============================] - 25s 1s/step - loss: 0.0000e+00 - accuracy: 0.7346 - val_los
s: 0.0000e+00 - val_accuracy: 0.7736
Epoch 2/10
16/16 [==============================] - 22s 1s/step - loss: 0.0000e+00 - accuracy: 0.7754 - val_los
s: 0.0000e+00 - val_accuracy: 0.7736
Epoch 3/10
16/16 [==============================] - 22s 1s/step - loss: 0.0000e+00 - accuracy: 0.7754 - val_los
s: 0.0000e+00 - val_accuracy: 0.7736
Epoch 4/10
16/16 [==============================] - 22s 1s/step - loss: 0.0000e+00 - accuracy: 0.7754 - val_los
s: 0.0000e+00 - val_accuracy: 0.7736
```

Итоги решений

Задача 1 - HATE

SMOTE + CountVectorizer + SVC

```
sm = SMOTE (#sampling_strategy = 0.9,
        random_state=0,
        k_neighbors=4)
X_train_res, y_train_res = sm.fit_resample(bow_10, y_train_10)
```

```
print('\t\tДО балансировки \tПОСЛЕ балансировки ')
print('класс 2 : \t{}\t\t\t{}'.format(sum(y_train_10==2), sum(y_train_res==2)))
print('класс 1 : \t{}\t\t\t{}'.format(sum(y_train_10==1), sum(y_train_res==1)))
print('класс 0 : \t{}\t\t\t{}'.format(sum(y_train_10==0), sum(y_train_res==0)))
print('y :\t\t{}\t\t\t{}'.format(y_train_10.shape, y_train_res.shape))
print('X :\t\t{}\t\t{}'.format(bow_10.shape, X_train_res.shape))
```

```
                ДО балансировки         ПОСЛЕ балансировки
класс 2 :       3368                    3368
класс 1 :       3159                    3368
класс 0 :       1147                    3368
y :             (7674,)                 (10104,)
X :             (7674, 11677)           (10104, 11677)
```

```
model_10 = SVC()
model_10.fit(X_train_res, y_train_res)
pred_10 = model_10.predict(vec_10.transform(X_test_10))
print(classification_report(pred_10, y_test_10))
```

```
              precision    recall  f1-score   support

           0       0.51      0.62      0.56       240
           1       0.82      0.89      0.85       716
           2       0.95      0.84      0.89       963

    accuracy                           0.83      1919
   macro avg       0.76      0.78      0.77      1919
weighted avg       0.85      0.83      0.84      1919
```

SMOTE + CountVectorizer + LogReg - 15000 и 5000

```
clf_00 = LogisticRegression(random_state=42, solver='liblinear')
clf_00.fit(X_train_res, y_train_res)
pred_00 = clf_00.predict(vec_10.transform(X_test_10))
print(classification_report(pred_00, y_test_10))
```

```
              precision    recall  f1-score   support

           0       0.47      0.32      0.38       434
           1       0.89      0.94      0.92      3644
           2       0.87      0.82      0.84       879

    accuracy                           0.86      4957
   macro avg       0.74      0.69      0.71      4957
weighted avg       0.85      0.86      0.86      4957
```

```
clf_00 = LogisticRegression(random_state=42, solver='liblinear')
clf_00.fit(X_train_res, y_train_res)
pred_00 = clf_00.predict(vec_10.transform(X_test_10))
print(classification_report(pred_00, y_test_10))
```

```
              precision    recall  f1-score   support

           0       0.62      0.57      0.59       318
           1       0.83      0.89      0.86       731
           2       0.91      0.90      0.91       870

    accuracy                           0.84      1919
   macro avg       0.79      0.78      0.79      1919
weighted avg       0.84      0.84      0.84      1919
```

Итоги решений

Задача 1 - HATE

BERT + LogReg - 3000

```
lr_clf_11 = LogisticRegression(class_weight = 'balanced')
lr_clf_11.fit(train_features, train_labels)
print(classification_report(lr_clf_11.predict(test_features), test_labels))

              precision    recall  f1-score   support

           0       0.53      0.24      0.33       171
           1       0.76      0.94      0.84       604
           2       0.78      0.60      0.68       225

    accuracy                           0.74      1000
   macro avg       0.69      0.59      0.61      1000
weighted avg       0.72      0.74      0.71      1000
```

BERT + LogReg - 1000 - 0.2 + SMOTE

```
lr_clf_11 = LogisticRegression(class_weight = 'balanced')
lr_clf_11.fit(X_train_res, y_train_res)
print(classification_report(lr_clf_11.predict(test_features), test_labels))

              precision    recall  f1-score   support

           0       0.39      0.39      0.39        18
           1       0.86      0.88      0.87       137
           2       0.72      0.69      0.70        45

    accuracy                           0.79       200
   macro avg       0.66      0.65      0.65       200
weighted avg       0.79      0.79      0.79       200
```

BERT + LogReg - 1000

```
lr_clf_11 = LogisticRegression(class_weight = 'balanced')
lr_clf_11.fit(train_features, train_labels)
print(classification_report(lr_clf_11.predict(test_features), test_labels))

              precision    recall  f1-score   support

           0       0.45      0.36      0.40        14
           1       0.86      0.93      0.89       145
           2       0.72      0.56      0.63        41

    accuracy                           0.81       200
   macro avg       0.68      0.62      0.64       200
weighted avg       0.80      0.81      0.81       200
```

BERT + LogReg - 1000 - 0.35 + SMOTE

```
lr_clf_11 = LogisticRegression(class_weight = 'balanced')
lr_clf_11.fit(X_train_res, y_train_res)
print(classification_report(lr_clf_11.predict(test_features), test_labels))

              precision    recall  f1-score   support

           0       0.33      0.40      0.36        20
           1       0.89      0.90      0.89       259
           2       0.69      0.61      0.65        71

    accuracy                           0.81       350
   macro avg       0.64      0.64      0.64       350
weighted avg       0.82      0.81      0.81       350
```

Итоги решений

Задача 1 - HATE

BERT + SVC- 3000

```
lr_clf_12 = SVC(class_weight = 'balanced')
lr_clf_12.fit(train_features, train_labels)
print(classification_report(lr_clf_12.predict(test_features), test_labels))

              precision    recall  f1-score   support

           0       0.55      0.23      0.32       189
           1       0.72      0.94      0.82       575
           2       0.79      0.58      0.67       236

    accuracy                           0.72      1000
   macro avg       0.69      0.58      0.60      1000
weighted avg       0.71      0.72      0.69      1000
```

BERT + SVC- 1000 - 0.2/0.35

```
lr_clf_12 = SVC(class_weight = 'balanced')
lr_clf_12.fit(train_features, train_labels)
print(classification_report(lr_clf_12.predict(test_features), test_labels))

              precision    recall  f1-score   support

           0       0.36      0.13      0.20        30
           1       0.76      0.94      0.84       126
           2       0.66      0.48      0.55        44

    accuracy                           0.72       200
   macro avg       0.59      0.52      0.53       200
weighted avg       0.68      0.72      0.68       200
```

```
lr_clf_12 = SVC(class_weight = 'balanced')
lr_clf_12.fit(train_features, train_labels)
print(classification_report(lr_clf_12.predict(test_features), test_labels))

              precision    recall  f1-score   support

           0       0.57      0.15      0.24        52
           1       0.77      0.96      0.85       221
           2       0.72      0.56      0.63        77

    accuracy                           0.75       350
   macro avg       0.69      0.56      0.57       350
weighted avg       0.73      0.75      0.71       350
```

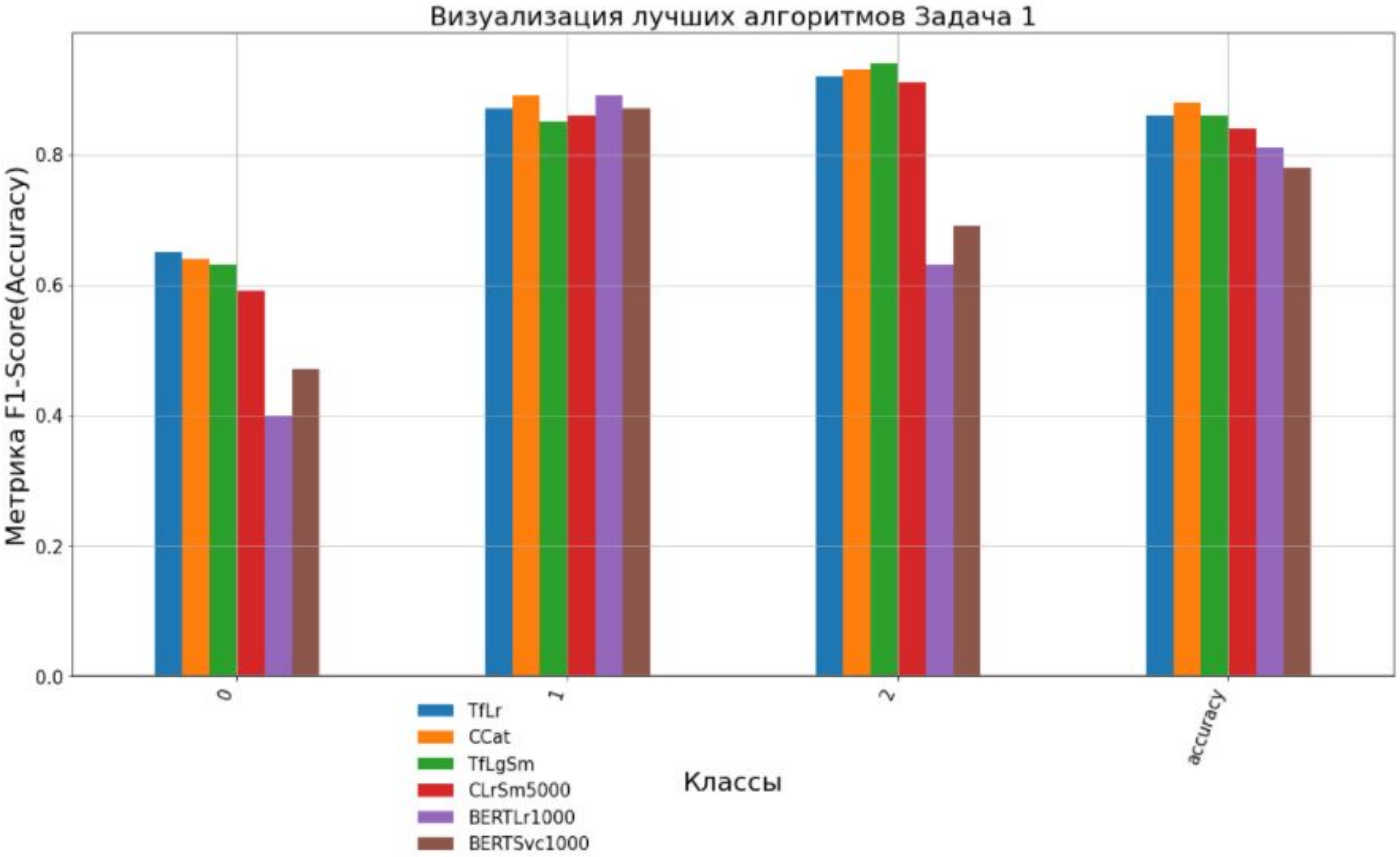
BERT + SVC- 1000 - 0.2/0.35 + SMOTE

```
lr_clf_12 = SVC(class_weight = 'balanced')
lr_clf_12.fit(X_train_res, y_train_res)
print(classification_report(lr_clf_12.predict(test_features), test_labels))

              precision    recall  f1-score   support

           0       0.61      0.38      0.47        29
           1       0.81      0.94      0.87       119
           2       0.77      0.63      0.69        52

    accuracy                           0.78       200
   macro avg       0.73      0.65      0.68       200
weighted avg       0.77      0.78      0.77       200
```

```
lr_clf_12 = SVC(class_weight = 'balanced')
lr_clf_12.fit(X_train_res, y_train_res)
print(classification_report(lr_clf_12.predict(test_features), test_labels))

              precision    recall  f1-score   support

           0       0.54      0.30      0.39        43
           1       0.80      0.94      0.86       223
           2       0.73      0.54      0.62        84

    accuracy                           0.77       350
   macro avg       0.69      0.59      0.62       350
weighted avg       0.75      0.77      0.75       350
```

Итоги решений

Задача 1 - HATE



	TfLr	CCat	TfLgSm	CLrSm5000	BERTLr1000	BERTSvc1000
0	0.65	0.64	0.63	0.59	0.40	0.47
1	0.87	0.89	0.85	0.86	0.89	0.87
2	0.92	0.93	0.94	0.91	0.63	0.69
accuracy	0.86	0.88	0.86	0.84	0.81	0.78

2 1 3 4 5 6

Итоги решений

Задача 2 - TROLL

CountVectorizer + LogisticRegression

```
vec_2 = CountVectorizer(ngram_range=(1, 1))
vec_2.fit(data_total_4_English['content'].values.astype('U'))
bow_2 = vec_2.transform(X_train_E_2)
```

```
clf_2_1 = LogisticRegression(random_state=42, solver='liblinear',
                             class_weight = 'balanced')
clf_2_1.fit(bow_2, y_train_E_2)
pred_2_1 = clf_2_1.predict(vec_2.transform(X_test_E_2))
print(classification_report(pred_2_1, y_test_E_2))
```

```
              precision    recall  f1-score   support

           0       0.95      0.94      0.94     10628
           1       0.68      0.28      0.40      2544
           2       0.83      0.80      0.81     22712
           3       0.67      0.69      0.68     38740
           4       0.88      0.86      0.87     57489
           5       0.59      0.29      0.39      5420
           6       0.72      0.83      0.77     58206
           7       0.25      0.04      0.07      4261

    accuracy                           0.78    200000
   macro avg       0.69      0.59      0.62    200000
weighted avg       0.77      0.78      0.77    200000
```

TfidfVectorizer + LogisticRegression

```
pipe_2_2 = Pipeline([
                ('tf-idf', TfidfVectorizer()),
                ('LogReg', LogisticRegression(random_state=42,
                                              solver='liblinear',
                                              class_weight = 'balanced'))
                ])
```

```
pipe_2_2.fit(X_train_E_2, y_train_E_2)
y_pred_2_2 = pipe_2_2.predict(X_test_E_2)
print(classification_report(y_pred_2_2, y_test_E_2))
```

```
              precision    recall  f1-score   support

           0       0.96      0.92      0.94     10927
           1       0.68      0.29      0.41      2464
           2       0.81      0.85      0.83     20844
           3       0.67      0.67      0.67     39872
           4       0.88      0.85      0.87     58209
           5       0.60      0.27      0.38      5995
           6       0.71      0.83      0.76     57654
           7       0.25      0.04      0.07      4035

    accuracy                           0.77    200000
   macro avg       0.70      0.59      0.62    200000
weighted avg       0.76      0.77      0.76    200000
```

Итоги решений

Задача 2 - TROLL

CountVectorizer + XGBClassifier

```
pipe5_2 = Pipeline([
                    ('CountVectChar', CountVectorizer(ngram_range=(1, 1))),
                    ('XGB', XGBClassifier(objective = 'multi:softprob' ,
                                          use_label_encoder=False,
                                          eval_metric='mlogloss'))
                   ])
```

```
pipe5_2.fit(X_train_E_2, y_train_E_2)
y_pred5_2 = pipe5_2.predict(X_test_E_2)
print(classification_report(y_pred5_2, y_test_E_2))
```

```
              precision    recall  f1-score   support

           0       0.77      0.94      0.85      8531
           1       0.50      0.82      0.62       643
           2       0.57      0.98      0.72     12713
           3       0.42      0.71      0.53     23654
           4       0.76      0.84      0.80     51063
           5       0.17      0.85      0.28       537
           6       0.86      0.56      0.68    102833
           7       0.03      0.65      0.05        26

    accuracy                           0.70    200000
   macro avg       0.51      0.79      0.57    200000
weighted avg       0.76      0.70      0.70    200000
```

CountVectorizer + XGBClassifier

```
pipe7_2 = Pipeline([
                    ('CountVectChar', CountVectorizer(ngram_range=(1, 1))),
                    ('CBC', CatBoostClassifier( learning_rate=1, depth=2,
                                                loss_function='MultiClass'))
                   ])
pipe7_2.fit(X_train_E_2, y_train_E_2)
y_pred7_2 = pipe7_2.predict(X_test_E_2)
print(classification_report(y_pred7_2, y_test_E_2))
```

```
              precision    recall  f1-score   support

           0       0.66      0.95      0.78      7365
           1       0.32      0.89      0.47       383
           2       0.37      1.00      0.54      8195
           3       0.25      0.60      0.36     16715
           4       0.64      0.82      0.72     43865
           5       0.00      0.25      0.00         4
           6       0.87      0.47      0.61    123473
           7       0.00      0.00      0.00         0

    accuracy                           0.60    200000
   macro avg       0.39      0.62      0.44    200000
weighted avg       0.74      0.60      0.62    200000
```

TfidfVectorizer + XGBClassifier

```
pipe6_2 = Pipeline([
                    ('tf-idf', TfidfVectorizer()),
                    ('XGB', XGBClassifier(objective = 'multi:softprob' ,
                                          use_label_encoder=False,
                                          eval_metric='mlogloss'))
                   ])
pipe6_2.fit(X_train_E_2, y_train_E_2)
y_pred6_2 = pipe6_2.predict(X_test_E_2)
print(classification_report(y_pred6_2, y_test_E_2))
```

```
              precision    recall  f1-score   support

           0       0.77      0.95      0.85      8498
           1       0.49      0.81      0.61       646
           2       0.57      0.99      0.72     12718
           3       0.42      0.72      0.53     23034
           4       0.77      0.84      0.80     51026
           5       0.18      0.80      0.29       600
           6       0.87      0.56      0.68    103444
           7       0.03      0.56      0.05        34

    accuracy                           0.70    200000
   macro avg       0.51      0.78      0.57    200000
weighted avg       0.76      0.70      0.70    200000
```

Итоги решений

Задача 2 - TROLL

TfidfVectorizer+ LGBMClassifier

```
pipe0 = Pipeline([
                ('tf-idf', TfidfVectorizer()),
                ('LGMClass', LGBMClassifier())
                 ])
pipe0.fit(X_train_E_2, y_train_E_2)
y_pred0 = pipe0.predict(X_test_E_2)
print(classification_report(y_pred0, y_test_E_2))
```

	precision	recall	f1-score	support
0	0.82	0.94	0.87	9129
1	0.40	0.63	0.49	670
2	0.63	0.98	0.76	14041
3	0.51	0.67	0.58	30104
4	0.82	0.83	0.83	55709
5	0.22	0.75	0.34	798
6	0.83	0.62	0.71	89419
7	0.06	0.32	0.10	130
accuracy			0.73	200000
macro avg	0.54	0.72	0.59	200000
weighted avg	0.76	0.73	0.73	200000

TfidfVectorizer + LGBMClassifier + SMOTE

```
vec_10_1 = TfidfVectorizer()
vec_10_1.fit(data_total_4_English['content'].values.astype('U'))
bow_10_1 = vec_10_1.transform(X_train_E_2)
```

```
sm_1 = SMOTE (#sampling_strategy = 0.9,
        random_state=0,
        k_neighbors=100)
X_train_res_1, y_train_res_1 = sm_1.fit_resample(bow_10_1, y_train_E_2)
```

```
pipe0_1 = LGBMClassifier()

pipe0_1.fit(X_train_res_1, y_train_res_1)
y_pred0_1 = pipe0_1.predict(vec_10_1.transform(X_test_E_2))
print(classification_report(y_pred0_1, y_test_E_2))
```

	precision	recall	f1-score	support
0	0.86	0.87	0.86	10437
1	0.64	0.38	0.48	1797
2	0.66	0.96	0.78	15032
3	0.62	0.52	0.56	47534
4	0.81	0.84	0.82	54046
5	0.51	0.16	0.25	8533
6	0.62	0.80	0.70	51678
7	0.34	0.02	0.04	10943
accuracy			0.69	200000
macro avg	0.63	0.57	0.56	200000
weighted avg	0.67	0.69	0.66	200000

Итоги решений

Задача 2 - TROLL

SMOTE + CountVectorizer + LogisticRegression (300 000)

```
X_train_E_2, X_test_E_2, y_train_E_2, y_test_E_2 = train_test_split(data_total_4_English['content'].values.astype('U'),
                                                                     data_total_4_English['account_category'] ,
                                                                     test_size = 0.2)
```

```
vec_10_E = CountVectorizer( ngram_range=(1, 1))
vec_10_E.fit(data_total_4_English['content'].values.astype('U'))
bow_10_E = vec_10_E.transform(X_train_E_2)
```

```
sm = SMOTE (#sampling_strategy = 0.9,
        random_state=0,
        k_neighbors=25)
X_train_res_E, y_train_res_E = sm.fit_resample(bow_10_E, y_train_E_2)
```

```
print('\t\tДО балансировки \tПОСЛЕ балансировки ')
print('y :\t\t{}\t\t{}'.format(y_train_E_2.shape, y_train_res_E.shape))
print('X :\t\t{}\t\t{}'.format(bow_10_E.shape, X_train_res_E.shape))
```

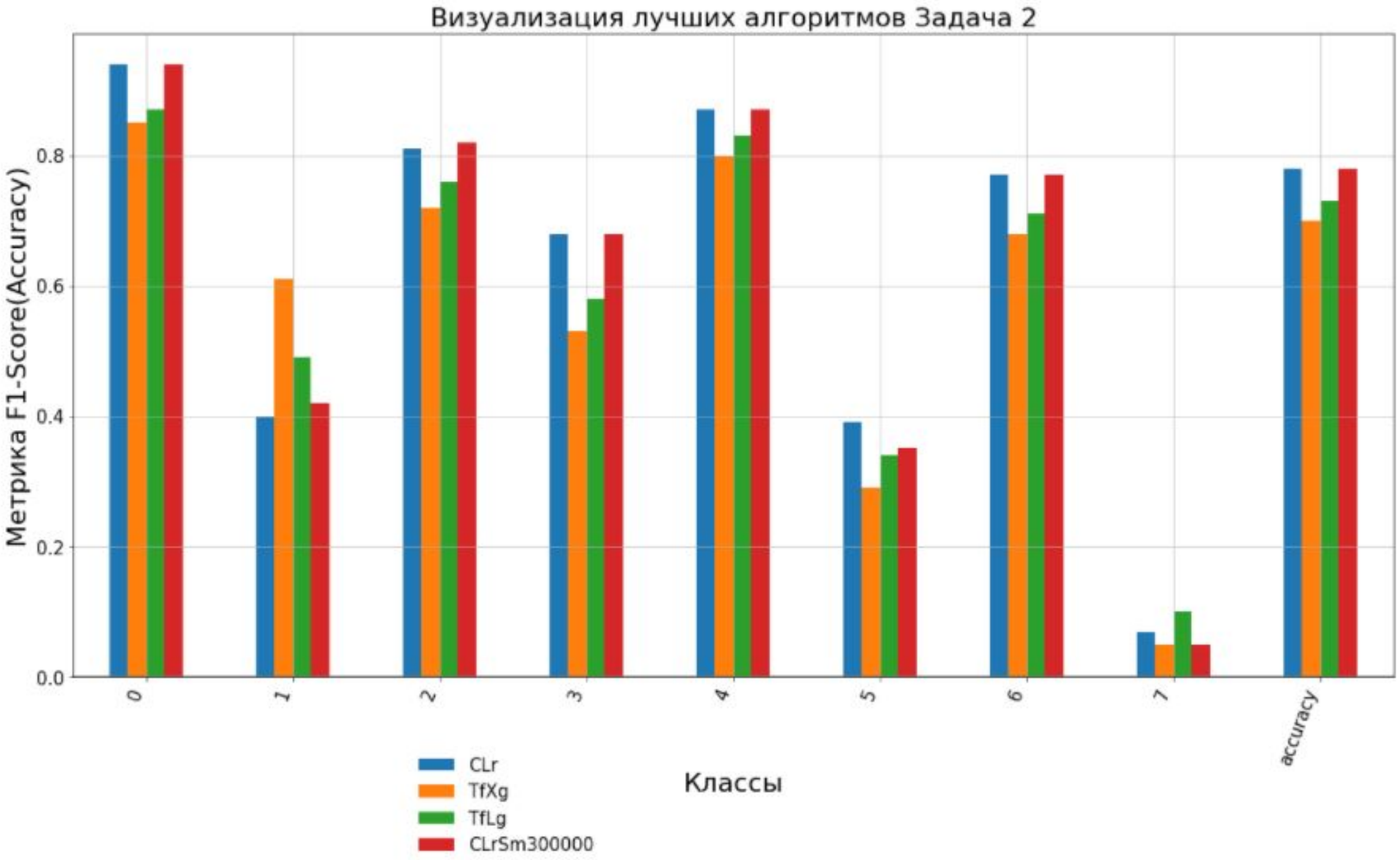
```
                ДО балансировки         ПОСЛЕ балансировки
y :             (800000,)               (2137584,)
X :             (800000, 212850)                (2137584, 212850)
```

```
model_8_1 = LogisticRegression(random_state=42, solver='liblinear')
model_8_1.fit(X_train_res_E, y_train_res_E)
pred_8_1 = model_8_1.predict(vec_10_E.transform(X_test_E_2))
print(classification_report(pred_8_1, y_test_E_2))
```

```
              precision    recall  f1-score   support

           0       0.93      0.94      0.94     10575
           1       0.58      0.33      0.42      1768
           2       0.83      0.81      0.82     22708
           3       0.66      0.69      0.68     38335
           4       0.89      0.85      0.87     58710
           5       0.45      0.29      0.35      4008
           6       0.74      0.82      0.77     60677
           7       0.14      0.03      0.05      3219

    accuracy                           0.78    200000
   macro avg       0.65      0.60      0.61    200000
weighted avg       0.77      0.78      0.78    200000
```

Итоги решений

Задача 2 - TROLL



	CLr	TfXg	TfLg	CLrSm300000
0	0.94	0.85	0.87	0.94
1	0.40	0.61	0.49	0.42
2	0.81	0.72	0.76	0.82
3	0.68	0.53	0.58	0.68
4	0.87	0.80	0.83	0.87
5	0.39	0.29	0.34	0.35
6	0.77	0.68	0.71	0.77
7	0.07	0.05	0.10	0.05
accuracy	0.78	0.70	0.73	0.78
	✓	✗	✗	✗
	1	3	4	2

Выводы

6

Выводы

1. Лучше всего для задач определения тональности текста подходит алгоритм:
    a. LogisticRegression
2. Ансамблевые модели показывают не лучшие результаты на большом количестве данных, чем логистическая регрессия.
3. В дальнейшем, повысить качество можно:
    a. путём использованием всех столбцов данных
    b. настройкой гиперпараметров при помощи GridSearchCV, RandomizedSearchCV
    c. запуском обучения, используя большее количество ресурсов

Список источников

7

Список источников

- http://neerc.ifmo.ru/wiki/
- http://neerc.ifmo.ru/wiki/index.php
- https://proglib.io/p/analiz-tonalnosti-teksta-proshloe-nastoyashchee-i-budushchee-2020-11-30

Николай Павлов

Junior Data Scientist

Аккаунты в соцсетях

vk.com/id51647681

t.me/NGPavlov

github.com/PNikolayG

Почта

pavlovnikg@gmail.com

Резюме

https://docs.google.com/document/d/1OUWezQMI4xVgL7o3v7PJ_VxnIhk5cvKbFqJiRjLK4kA/edit?usp=sharing

“Буду Вам очень благодарен за рекомендацию!”

Анализ эмоциональной окраски текста

Спасибо за внимание!

Николай Павлов
группа DS-27

pavlovnikg@gmail.com

t.me/NGPavlov