Индивидуальное задание

1. Изучить поставленную задачу и выбрать набор данных, на котором будет производиться исследование и обучение модели.
2. Исследовать и обработать набор данных
3. Построить модель для оценки тональности мнений
4. Тестирование и оценка качества построенной модели
5. Подведение итогов

Описание задачи и технологии программирования

Во время прохождения производственной практики была поставлена задача построить модель машинного обучения для определения эмоциональной окраски финансовых новостей, т.е. определения является ли новость позитивной, негативной или нейтральной, а также изучить основные процессы обработки естественного языка.

Средой разработки был выбран Jupiter Notebook. Это самый популярный инструмент, применяемый в сфере машинного обучения и анализа данных, позволяющий работать с отдельными фрагментами кода. Все результаты пользователь Jupiter Notebook может контролировать и сразу видеть после выполнения каждой ячейки, что помогает продуктивней работать над задачей и исправлять мелкие ошибки по ходу выполнения программы.

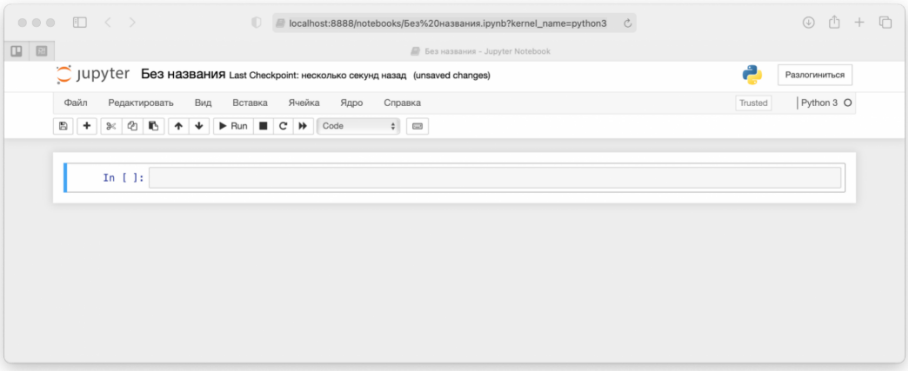


рис. 1. Среда Jupiter Notebook

Разработка и тестирование программы

Описание используемых библиотек.

Анализ тональности, по сути, представляет собой исследование неструктурированных текстовых данных. Он включает в себя сочетание таких дисциплин, как статистика, обработка естественного языка (NLP) и машинное обучение (Machine Learning). Применяется для извлечения субъективной информации из текста.

Точный или детальный анализ текста состоит из анализа предложений по частям. В этом случае предполагается анализ отдельных фраз, и каждая часть анализируется в связи с другими. Такой анализ помогает понять, почему автор дал такую оценку объекту или теме.

Текущая задача предполагает получение только тональности текста по трём категориям: негативный, позитивный и нейтральный. Это первый шаг, который необходимо сделать, чтобы в дальнейшем можно было на основе полученной тональности извлечь более подробные данные по объекту.

В данной задаче будут использоваться следующие библиотеки:

- Numpy — предназначен для поддержки многомерных массивов и высокоуровневых математических операций над ними.

- Pandas — в дополнении с Numpy предоставляет специальные структуры данных и операции для манипулирования числовыми таблицами. Предназначен не только для сбора и очистки данных, но для задач анализа, моделирования и представления данных.

- Seaborn — библиотека визуализации данных Python, основанная на matplotlib . Он предоставляет высокоуровневый интерфейс для рисования привлекательных и информативных статистических графиков.

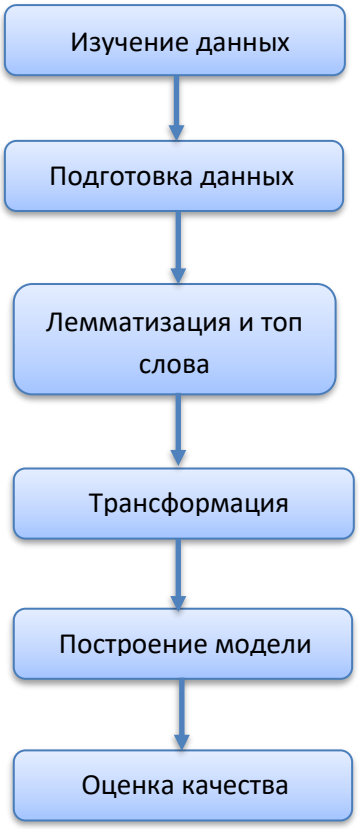
- Wordcloud - помогает узнать частоту появления слова в текстовом контенте с помощью визуализации.

- Nltk(Natural Language Toolkit) – библиотека для символьной и статистической обработки естественного языка. Содержит графическое представление, наборы готовых данных и многочисленные инструменты для анализа текста и его представления.

- Sklearn – широко используемый пакет для анализа данных и машинного обучения. Содержит функции и алгоритмы для машинного обучения: классификации, прогнозирования, разбивки данных на группы, а также различные метрики для оценки качества обученной модели.

Алгоритм выполнения.

Основные этапы построения модели для оценки тональности мнений:



1. Изучение данных

Набор данных взят с Kaggle под названием – Sentiment Analysis for Financial News(ссылка в списке литературы). Содержит оценки заголовков финансовых новостей с точки зрения розничного инвестора.

	Sentiment	Sentence
0	neutral	According to Gran , the company has no plans t...
1	neutral	Technopolis plans to develop in stages an area...
2	negative	The international electronic industry company ...
3	positive	With the new production plant the company woul...
4	positive	According to the company 's updated strategy f...
...	...	...
4841	negative	LONDON MarketWatch -- Share prices ended lower...
4842	neutral	Rinkuskiai 's beer sales fell by 6.5 per cent ...
4843	negative	Operating profit fell to EUR 35.4 mn from EUR ...
4844	negative	Net sales of the Paper segment decreased to EU...
4845	negative	Sales in Finland decreased by 10.5 % in Januar...

Рис.2 – набор данных

Набор данных содержит 4846 строк информации.

Применив функцию describe к данным, получим статистические оценки для анализа полноты информации.

```
data.describe()
```

	Sentiment	Sentence
count	4846	4846
unique	3	4838
top	neutral	Ahlstrom 's share is quoted on the NASDAQ OMX ...
freq	2879	2

рис. 3. Применение describe()

Как видно по результату, имеется 4838 уникальных заголовков финансовых новостей из 4846. Следовательно, существуют 8 дубликатов заголовков, которые следует удалить.

```
data = data.drop_duplicates().copy(deep=True)
data
```

	Sentiment	Sentence
0	neutral	According to Gran , the company has no plans t...
1	neutral	Technopolis plans to develop in stages an area...
2	negative	The international electronic industry company ...
3	positive	With the new production plant the company woul...
4	positive	According to the company 's updated strategy f...
...	...	...
4841	negative	LONDON MarketWatch -- Share prices ended lower...
4842	neutral	Rinkuskiai 's beer sales fell by 6.5 per cent ...
4843	negative	Operating profit fell to EUR 35.4 mn from EUR ...
4844	negative	Net sales of the Paper segment decreased to EU...
4845	negative	Sales in Finland decreased by 10.5 % in Januar...

рис. 4. Применение drop_duplicates()

В результате получаем:

```
data.describe()
```

	Sentiment	Sentence
count	4840	4840
unique	3	4838
top	neutral	TELECOMWORLDWIRE-7 April 2006-TJ Group Plc sel...
freq	2873	2

рис. 5. применение повторно describe()

Это значит, что еще 2 значения так и не были удалены. Функция drop_duplicates() осуществляет поиск дубликатов по обоим имеющим столбцам. Следовательно, 2 заголовка имеют разные размеченные тональности.

Определим какое из них оставить, а какое удалить, посмотрев на каждый текст детально.

```
# найдем эти два дубликата
data[data['Sentence'].duplicated()]
```

	Sentiment	Sentence
79	positive	TELECOMWORLDWIRE-7 April 2006-TJ Group Plc sel...
789	neutral	The Group 's business is balanced by its broad...

рис. 6. Удаление дубликатов

Вот два полученных дубликата.

1. 'TELECOMWORLDWIRE-7 April 2006-TJ Group Plc sells stake in Morning Digital Design Oy Finnish IT company TJ Group Plc said on Friday 7 April tha t it had signed an agreement on selling its shares of Morning Digital Design Oy to Edita Oyj .'
   Тональность: нейтральный или позитивный.

2. "The Group 's business is balanced by its broad portfolio of sports and presence in all major markets ."
   Тональность: нейтральный или позитивный.

Имеет смысл оставить оба заголовка под меткой тональности – нейтральный.

Теперь имеем чистый и готовый набор данных, содержащий 4838 строк информации.

2. Подготовка данных.

Подготовка данных предполагает очистку текста от нерелевантных слов и символов, препятствующих дальнейшему анализу текста и обучению модели.

Такая очистка предполагает следующие действия:

1. Очистка неподходящих символов
   - Очистка текста от пунктуаций.
   - Перевод слов в нижний регистр.
   - Удаление набора символов описывающих ссылки
   - Удаление символов, находящихся за скобками(как круглыми, так и квадратными и угловыми)

Со всем этим справится следующая функция:

```
def clean_data(text):
    text = str(text).lower()
    text = re.sub('\[.*?\]', '', text)
    text = re.sub('\(.*?\)', '', text)
    text = re.sub('https?://\S+|www\.\S+', '', text)
    text = re.sub('<.*?>+', '', text)
    text = re.sub('[%s]' % re.escape(string.punctuation), '', text)
    text = re.sub('\n', '', text)
    text = re.sub('\w*\d\w*', '', text)
    return text
```

```
data['Sentence'] = data['Sentence'].apply(lambda x:clean_data(x))
```

рис. 7. Функция очистки данных от нерелевантных символов и ее применение.

2. Удаление стоп слов

Стоп слова, это такие слова, которые не способствуют более глубокому значению фразы, такие как «а», «я», «и» и т.д. Для языка Python используется библиотека NLTK которая предоставляет список общепринятых стоп– слов для различных языков, таких как английский.

```
import nltk
from nltk.corpus import stopwords
nltk.download('stopwords')

def remove_stopwords(x):
    return [y for y in x if y not in stopwords.words('english')]
```

```
from collections import Counter

data['wordlist'] = data['Sentence'].apply(lambda x:str(x).split())
data['wordlist'] = data['wordlist'].apply(lambda x: remove_stopwords(x))

top = Counter([item for sublist in data['wordlist'] for item in sublist])
```

рис. 8. Функция удаления стоп слов и ее применение на столбец заголовков, включая создания массива подсчета слов, встречающихся в наборе данных

Применим написанную функцию на столбец заголовков. Сформировав отдельную колонку для записи полученного массива и разделив предложения на токены, получим список слов для каждого заголовка.

3. Лемматизация.

В области обработки натурального языка используются два метода текстовой нормализации - стемминг и лемматизация, используемые для приготовления текстов, документов для дальнейшего анализа.

Стемминг производит сокращение слов до своих грамматических основ. Основа слова – стем, не обязательно совпадает с корнем, он может включать и суффиксы. Это неизменяемая при склонении часть.

Алгоритмы стемминга обычно основаны на правилах: слово проходит через ряд условных предложений, которые определяют, как его сократить. Например, существует правило суффиксов: в английском языке «-ed» и «-ing» отрезают, чтобы сопоставить "cooking" и "cooked" с одной и той же основой "cook".

Лемматизация – это алгоритмический процесс нахождения леммы слова – это означает, что в отличие от stemming, лемматизация уменьшает слово в зависимости от его значения. Это помогает в возвращении основания или словарной формы слова, которое и известно как лемма.

Почему лемматизация лучше стемминга?

Алгоритм Stemming работает, вырезая суффикс из слова. В более широком смысле отсекает начало или конец слова.

Лемматизация наоборот является более мощной операцией и учитывает морфологический анализ слов. Для создания словарей и поиска правильной формы слова необходимы глубокие лингвистические знания. Стемминг – это общая операция, а лемматизация – интеллектуальная операция, в которой правильная форма будет выглядеть в словаре. Следовательно, лемматизация помогает в формировании лучших возможностей машинного обучения. Ее и будем использовать далее.

Применим следующий фрагмент кода.

```
from nltk.stem import WordNetLemmatizer # lemmatizer

lemmatizer = WordNetLemmatizer()
def lemmatize(text):
    word = [lemmatizer.lemmatize(w) for w in text]
    return word
data['wordlist'] = data['wordlist'].apply(lambda x: lemmatize(x))
```

рис.9. Функция лемматизации и ее применение

И получим чистый массив лемм для каждого заголовка. На примере первого заголовка результат выглядит так:

['according', 'gran', 'company', 'plan',
'move', 'production', 'russia','although', 'company', 'growing']

Очистив и подготовив данные к построению модели, посмотрим для начала распределение слов к тексте по той или иной тональности и самые часто встречающиеся слова во всем наборе данных.

	Word	Count
0	eur	1004
1	company	968
2	said	545
3	finnish	511
4	mn	510
5	sale	500
6	share	433
7	million	431
8	profit	413
9	net	412
10	service	339
11	finland	332
12	year	323
13	group	321
14	operating	299
15	market	287
16	business	284
17	new	267
18	period	266
19	quarter	242

рис. 10. Топ 20 слов.

Видно, что полученный список самых часто встречаемых слов, как и предполагалось содержит в большем числе слова, используемые в финансовой сфере и политике. Это видно по словам: «eur», «company», «million», «sales», «profit» и др.

Этот массив слов можно изобразить в другом, более презентабельном виде.

рис. 11 – Карта слов.

Данная карта помогает визуализировать частоту появления тех или иных слов. То слово, что встречается наиболее часто, чем другое будет иметь больший размер.

Аналогично рис. 10. можно посмотреть на массив распространенных слов для каждой тональности.

Позитивный набор слов – рис. 12

	Word	Count
0	eur	448
1	company	272
2	mn	241
3	said	230
4	sale	203
5	finnish	198
6	net	196
7	profit	192
8	million	169
9	year	156
10	period	140
11	operating	122
12	quarter	110
13	service	108
14	mln	103
15	oyj	97
16	group	96
17	rose	94
18	increased	89
19	market	84

рис. 12. Позитивный набор

Нейтральный набор слов – рис. 13.

	Word	Count
1	said	238
2	eur	233
3	finnish	214
4	finland	214
5	business	188
6	million	187
7	group	184
8	new	179
9	sales	163
10	shares	163
11	services	151
12	also	145
13	share	137
14	market	131
15	net	112
16	total	100
17	oyj	93
18	nokia	90
19	financial	87

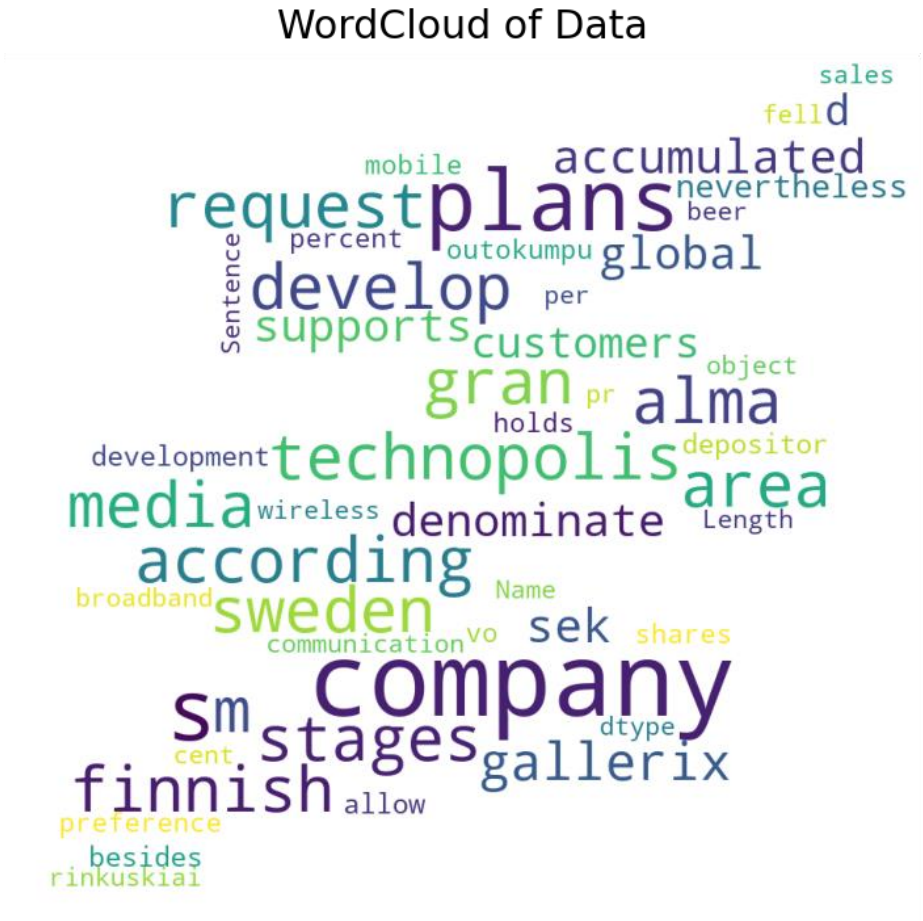
рис. 13. Нейтральный набор

Негативный набор слов – рис. 14.

	Word	Count
1	mn	224
2	profit	159
3	company	108
4	net	104
5	sale	102
6	finnish	99
7	operating	97
8	period	89
9	quarter	80
10	year	77
11	said	77
12	million	75
13	loss	74
14	compared	68

рис. 14. Негативный набор

Для более наглядной иллюстрации приведу пример визуализации облака слов нейтрального сентимента.



Чем больше шрифт, тем чаще оно встречается в выбранном наборе.

4. Трансформация

В большинстве алгоритмов машинного обучения набор данных может содержать текстовые или категориальные значения (в основном не числовые значения). Например, как в текущей задаче, тональность определяется тремя показателями – позитивный, негативный, отрицательный. Несколько алгоритмов, например таких как CatBoost, могут достаточно хорошо обрабатывать категориальные значения, но большинство алгоритмов работают лучше с числовыми данными, т.к. не могут выполнять каких-либо вычислений со строкой. Следовательно, необходимо преобразовать категориальные данные в числовые и при этом создать алгоритм/модель, чтобы из них получить необходимый результат.

Есть много способов преобразовать категориальные значения в числовые значения. Каждый подход имеет свои тонкости и влияет на набор функций модели. Остановимся на методе из библиотеки Scikit-learn – LabelEncoder(). Этот подход очень прост и включает преобразование каждого значения в столбце в число, по нумерации от 0 до n-1, где n – кол-во категориальных признаков.

Выполним данное преобразованием над столбцом тональности заголовков

```
le = LabelEncoder()
data['Sentiment'] = le.fit_transform(data['Sentiment'])
```

рис. 15. Кодирование

	Sentiment	Sentence	wordlist
0	1	according to gran the company has no plans to...	[according, gran, company, plan, move, product...
1	1	technopolis plans to develop in stages an area...	[technopolis, plan, develop, stage, area, le, ...
2	0	the international electronic industry company ...	[international, electronic, industry, company,...
3	2	with the new production plant the company woul...	[new, production, plant, company, would, incre...
4	2	according to the company s updated strategy fo...	[according, company, updated, strategy, year, ...
...	...	...	...
4841	0	london marketwatch share prices ended lower i...	[london, marketwatch, share, price, ended, low...
4842	1	rinkuskiai s beer sales fell by per cent to ...	[rinkuskiai, beer, sale, fell, per, cent, mill...
4843	0	operating profit fell to eur mn from eur mn ...	[operating, profit, fell, eur, mn, eur, mn, in...
4844	0	net sales of the paper segment decreased to eu...	[net, sale, paper, segment, decreased, eur, mn...
4845	0	sales in finland decreased by in january wh...	[sale, finland, decreased, january, sale, outs...

рис. 16. DataFrame после преобразования

Значения – негативный, нейтральный, позитивный были заменены на 0,1, 2 соответственно.

Теперь аналогично необходимо преобразовать слова, которые мы сгруппировали в массив для каждого имеющегося заголовка в набор чисел, чтобы машина могла их понять и обучиться на этом наборе данных.

Этот процесс называется векторизацией – извлечением признаков из текстовой информации. Для этого процесса будут использоваться методы библиотеки Scikit-learn – CountVectorizer и TfidTransformer.

- Класс **CountVectorizer** преобразует слова в тексте в матрицу частотности слов. Например, матрица содержит элемент a[i][j], который представляет частоту появления слова j в тексте типа i. Он вычисляет количество вхождений каждого слова с помощью функции fit_transform и получает ключевые слова всех текстов в сумке слов с помощью get_feature_names(). Все это далее приведено в рис. 17.

```
cv = CountVectorizer(lowercase = False,stop_words = 'english', min_df=2)
x = cv.fit_transform(x)

word = cv.get_feature_names()
```

рис. 17. CountVectorizer

Сумка слов будет выглядеть так

```
word[:10]
['ab', 'abb', 'abc', 'ability', 'able', 'abn', 'abp', 'abroad', 'ac', 'access']
```

рис. 18. Сумка слов.

Для наглядности показаны только первые 10 слов.

- **TfidTransformer** – используется для подсчета значения каждого слова в векторизаторе.

```
tfid = TfidfTransformer()
x_vector = tfid.fit_transform(x)
```

рис. 19. Подсчет каждого слова

5. Построение и обучение модели

Теперь, когда мы выделили признаки, можно обучать классификатор предсказывать категорию текста.

В качестве алгоритма машинного обучения для данной задачи был выбран мультиклассовый полиномиальный Наивный Байесовский классификатор(NB). Алгоритм основан на теореме Байеса и может предсказывать тег текста – в данной задаче тональность. Он вычисляет вероятность каждой тональности для данной выборки, а затем выдает тот, что имеет наибольшую вероятность.

Теорема Байеса, сформулированная Томасом Байесом, вычисляет вероятность события на основе предварительного знания условий, связанных с событием. Он основан на следующей формуле:

P(A|B) = P(A) * P(B|A)/P(B)

Где вычисляется вероятность класса A, когда предиктор B уже предоставлен.
P(B) = априорная вероятность B
P(A) = априорная вероятность класса A
P (B | A) = появление предиктора B с учетом вероятности класса A

Эта формула помогает в расчете вероятности тональности текста.

Разделив набор данных на тестовую и обучающую выборки следующим способом:

```
x_train, x_test, y_train, y_test = train_test_split(x_vector, y, test_size = 0.2, random_state = 5, stratify = y)
```

можем приступать к объявлению классификатора - MultinomialNB() и обучению

```
gnb = MultinomialNB()
model = gnb.fit(x_train, y_train)
```

рис. 20. процесс обучения классификатора и предсказание на тестовой выборке

На обучающей выборке модель обучена.

Теперь выполним предсказание для тестовой выборки и сравним с результатами, чтобы оценить качество обученной модели.

```
pred = model.predict(x_test)
score = accuracy_score(y_test, pred)
score
```

```
0.6859504132231405
```

рис. 21. Accuracy

Модель предсказывает 68% правильных ответов.

Метрика – accuracy_score показывает долю правильных ответов модели. Ее значение равно отношению числа правильных ответов, которые дала модель, к числу всех объектов.

6. Оценка качества

Метрика – accuracy_score не полностью отражает качество модели, поэтому используем функцию – classification_report() для более глубокого представления о поведении классификатора в отношении глобальной точности.

```
class_report = classification_report(y_test, pred)
print(class_report)
```

```
              precision    recall  f1-score   support

           0       0.80      0.10      0.18       121
           1       0.70      0.95      0.81       574
           2       0.61      0.40      0.48       273

    accuracy                           0.69       968
   macro avg       0.70      0.48      0.49       968
weighted avg       0.69      0.69      0.64       968
```

рис. 22 – отчет классификации

Отчет о классификации показывает представление основных метрик классификации для каждого класса.

Метрики определяются с точки зрения истинных и ложноположительных результатов, а также истинных и ложноотрицательных результатов.

Положительные и отрицательные в этом случае являются общими именами для предсказанных классов. Есть четыре способа проверить правильность прогнозов с помощью матрицы ошибок(рис. 23)

1. **TP / True Positive:** когда случай был положительным и прогнозировался положительным
2. **TN / True Negative:** когда случай был отрицательным и прогнозировался отрицательным
3. **FP / Ложноположительный:** когда случай был отрицательным, но прогнозировался положительным
4. **FN / False Negative:** когда случай был положительным, но прогнозировался отрицательным .

Используя эту терминологию, метрики определяются следующим образом:

- precision (точность) — можно рассматривать как меру точности классификатора. Для каждого класса он определяется как отношение истинных срабатываний к сумме истинных и ложных срабатываний. Другими словами, «для всех случаев, классифицированных как положительные, какой процент был правильным?»
- recall(полнота) — это мера полноты классификатора; способность классификатора правильно находить все положительные экземпляры. Для каждого класса он определяется как отношение истинно положительных результатов к сумме истинно положительных и ложноотрицательных результатов. Другими словами, «для всех случаев, которые были действительно положительными, какой процент был классифицирован правильно?»
- Показатель f1-score(мера) — это объединение precision и recall.
- Support(поддержка) — это количество фактических вхождений класса в указанном наборе данных. Несбалансированная поддержка в обучающих данных может указывать на структурные недостатки в сообщаемых оценках классификатора и может указывать на необходимость стратифицированной выборки или повторной балансировки.

Обращая внимание на значения precision для каждой тональности, можно сделать вывод, что наиболее точный прогноз был для позитивных тональностей. Это объясняется тем, что распределение количества тональностей не равномерно(рис. 23)

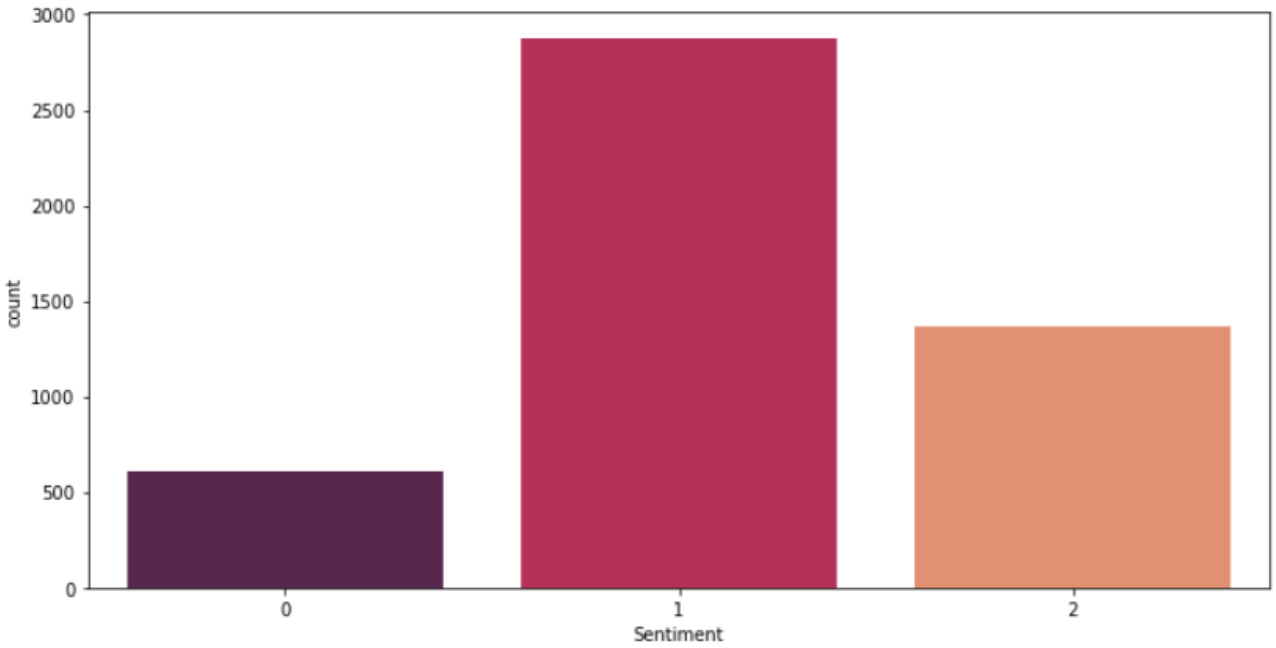


рис. 24. Кол-во тональностей в наборе

Где 1- negative, 0 – neutral, 2 – positive.

Также матрица ошибок содержит полезную информацию о классификации. Матрица ошибок показывает комбинацию фактического и предсказанного классов. Каждая строка матрицы представляет экземпляры в предсказанном классе, а каждый столбец - экземпляры в фактическом классе. Это хороший показатель того, могут ли модели учитывать совпадение свойств классов и понимать, какие классы легче всего ошибаются.

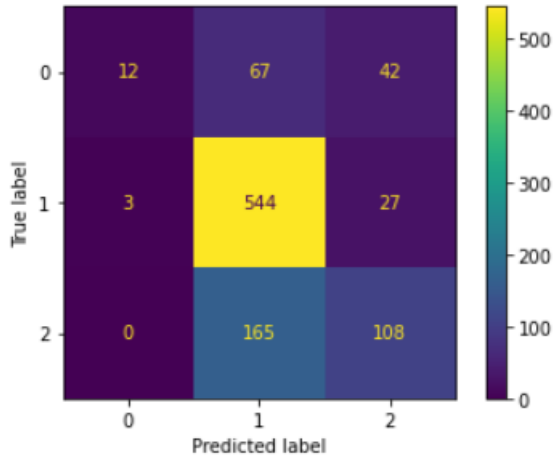


рис. 23. Матрица ошибок для трехклассовой классификации

где:

	0	1	2
TP	12	544	108
TN	844	138	626
FP	109	30	165
FN	3	232	69

В общем виде модель ошибалась гораздо меньше, чем правильно классифицировала. Из-за несбалансированности выборок тональностей модель предсказывает чуть лучше заголовки, имеющие негативную тональность и чуть хуже выборку с нейтральными заголовками из-за ее малочисленности в обучающем наборе.

Общая точность классификации составляет 68%, что является хорошим результатом для данной задачи.