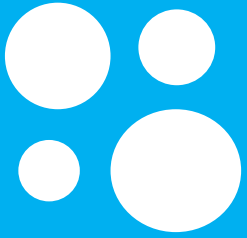


Содержание

- Описание данных и постановка задачи
- Обработка текста
 - Нормализация слов
 - Корректировка опечаток
 - Фильтрация слов
- Построение модели
 - Подготовка признаков
 - Выбор моделей
- Оценка модели
 - Метрика качества
 - Сравнение качества на разных уровнях иерархии



Описание данных и постановка задачи

Описание данных и постановка задачи

Данные представляют из себя объявления, характеризующиеся тремя колонками: оглавлением (title), описанием (description) и ценой.

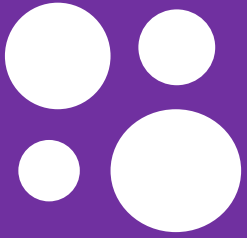
Для объявлений известны категории которые имеют иерархическую структуру.

Всего 489057 объявления с известными категориями (train.csv) и 242956 объявления с неизвестными категориями (test.csv)

Задача:

Построить классификатор для определения неизвестных категорий для объявлений

	title	description	price	category_id
	Картина	Гобелен. Размеры 139x84см.	1000.0	19
	Стулья из прессованной кожи	Продам недорого 4 стула из светлой прессованно...	1250.0	22
	Домашняя мини баня	Мини баня МБ-1(мини сауна), предназначена для ...	13000.0	37
	Эксклюзивная коллекция книг "Трансаэро" + подарок	Продам эксклюзивную коллекцию книг, выпущенную...	4000.0	43
	Ноутбук aser	Продаётся ноутбук ACER e5-511C2TA. Куплен в ко...	19000.0	1
	Бас гитара invasion bg110	Состояние хорошее. Имеется теплый чехол .	3999.0	50
	Смесь "Грудничок" г. Зеленодольск	Смесь молочная адаптированная ультрапастеризов...	15.0	41
	G-shock	Часы абсолютно новые! с коробкой. Часы Китай...	2500.0	36
	Санатории Белоруссии. - "Лепельский военный"	Санатории Белоруссии! - «Лепельский военный» ...	1090.0	48
	Фотохолст	Фотохолст на подрамнике. 36x58см. Галерейная н...	1250.0	19
	Ботильоны Nando Muzi	В хорошем состоянии,тёмно-коричневый цвет.Юмр	5000.0	39
	Игрушка playGro, Лев	Игрушка playGro, б/у, состояние отличное. (Вид...	400.0	30
	Кроватка для младенца	Кроватка для новорожденного регулируется в 2 п...	2999.0	34
	Продам утяжелители поясные новые	Продам утяжелители поясные новые, 2 кг. 400р. ...	500.0	46
	Цилиндрическая люстра	продам люстру,в отличном состоянии	2000.0	28
	Фацелия пижмолистная	Семена фацелии-отличный медонос,нетребовательн...	275.0	24
	Стол письменный Икеа	В хорошем состоянии . С полочками .105 на 50 С...	2300.0	22



Обработка текста

Нормализация слов

В процессе работы классификатора может получиться так, что одинаковые по смыслу слова будут иметь разные формы и поэтому алгоритм будет считать их разными.

Например,

```
In [25]: s1 = 'В Комплекте Два Джойстика'
```

```
In [26]: s2 = 'комплект из двух Джойстиков'
```

Предложения несут одинаковый смысл, однако так как все слова написаны по разному для алгоритма они абсолютно разные

После приведения слов к нижнему регистру, удаления стоп-слов и применения лемматизации предложения будут не только одинаковы по смыслу, но и по написанию

```
In [55]: s1 = deleteExtraSymbols(s1)
s1 = lemmatization(s1)
s1 = stopword(s1)
print(s1)
```

комплект джойстик

```
In [56]: s2 = deleteExtraSymbols(s2)
s2 = lemmatization(s2)
s2 = stopword(s2)
print(s2)
```

комплект джойстик

Корректировка опечаток

В тексте много опечаток, это плохо не только тем, что влияет на точность классификации, но и генерирует не несущие смысла признаки.

```
In [16]: df[0].head(10)
Out[16]: 0    стоик журнальный сталь продам журнальный столи...
         1    iphone gb телефон в хорошем состоянии комплект...

In [74]: word_frequency_little_test.get('пидаль')
Out[74]: 3

In [75]: word_frequency_little_test.get('педаль')
Out[75]: 301

In [76]: word_frequency_little_test.get('комплект')
Out[76]: 3532

In [77]: word_frequency_little_test.get('камплект')
Out[77]: 1

In [344]: df['0'][487325:487326]
Out[344]: 487325    x box elit продавать xbox elit g состояние хор...
```

Функция корректировки ищет из всех слов данного текста те, которые находятся в пределах одной или двух корректировок и выбирает наиболее встречающееся из них



В дополнение, так как мы применяем функцию после выделения словосочетаний, то некоторые названия моделей которые кто-то пишет раздельно объединятся в одно слово, что тоже может быть полезно

```
In [56]: correct('стоик')
Out[56]: 'столик'

In [375]: s = 'комплект камплект пидаль педаль'
In [377]: wordcorrection(s)
Out[377]: 'комплект комплект педаль педаль'

In [393]: correct('x-box')
Out[393]: 'xbox'

In [395]: correct('x_box')
Out[395]: 'xbox'

In [396]: correct('wi-fi')
Out[396]: 'wifi'
```

Фильтрация слов

Некоторые слова употребляются слишком часто, при этом не несут смысла с точки зрения принадлежности объявления к классу. Например, слово 'состояние'

```
In [22]: new_word_frequency.get('состояние')
```

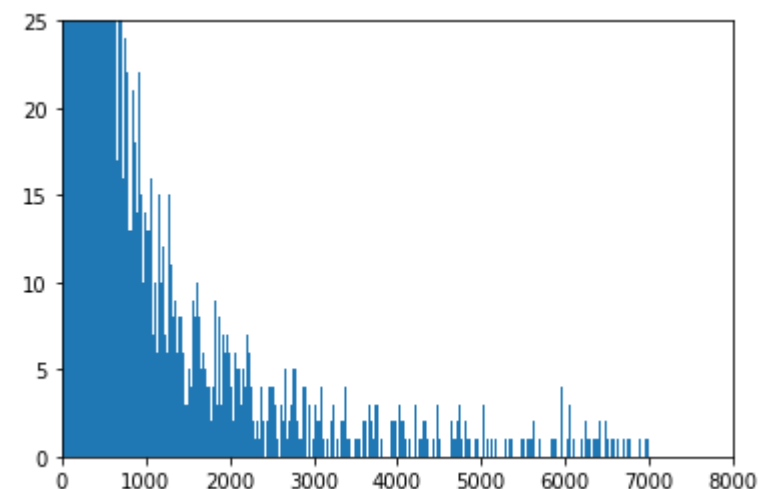
```
Out[22]: 164983
```

Как видно число употреблений в объявлениях больше чем среднее число объявлений для каждой группы. Это может негативно повлиять при классификации, так как будет путать классификатор потому что одно и то же слово употребляется в разных категориях

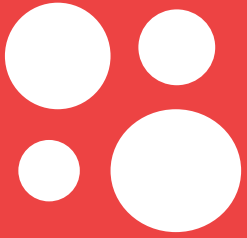
```
In [52]: new_word_frequency_selected = {word:
        frequency for word, frequency in new_word_frequency.items() if (frequency < 70000)}
```

Так же удалим низкочастотные слова, так как они не внесут статистической силы, а лишь увеличат потребление ресурсов

Теперь мы готовы к построению моделей



Распределение по частоте слов.
Обрезали до порога



Построение модели

Подготовка признаков

Так как алгоритмы машинного обучения работают с векторами признаков, то важным этапом является представление данных объектов в виде векторов.

Для этого воспользуемся методом CountVectorizer

Каждый текст при таком подходе представляется в виде вектора, где координата, это число вхождений того или иного уникального слова

```
In [70]: example = ['раз два три',  
                  'три четыре два два',  
                  'раз раз раз четыре']
```

Уникальные ключи
['раз', 'два', 'три', 'четыре']

```
In [71]: count_vect.fit_transform(example).toarray()  
Out[71]: array([[1, 1, 1, 0],  
               [2, 0, 1, 1],  
               [0, 3, 0, 1]], dtype=int64)
```

Для того чтобы учитывать, частоту числа вхождения слов в объявления, преобразуем наши векторы
С помощью TF-IDF:

```
In [74]: tfidf_transformer.fit_transform(example_count_vect).toarray()  
Out[74]: array([[ 0.57735027,  0.57735027,  0.57735027,  0.          ],  
               [ 0.81649658,  0.          ,  0.40824829,  0.40824829],  
               [ 0.          ,  0.9486833 ,  0.          ,  0.31622777]])
```

$$tf(t, d) = \frac{n_t}{\sum_k n_k} \quad \text{где } n_t \text{ есть число вхождений слова } t \text{ в документ,}$$

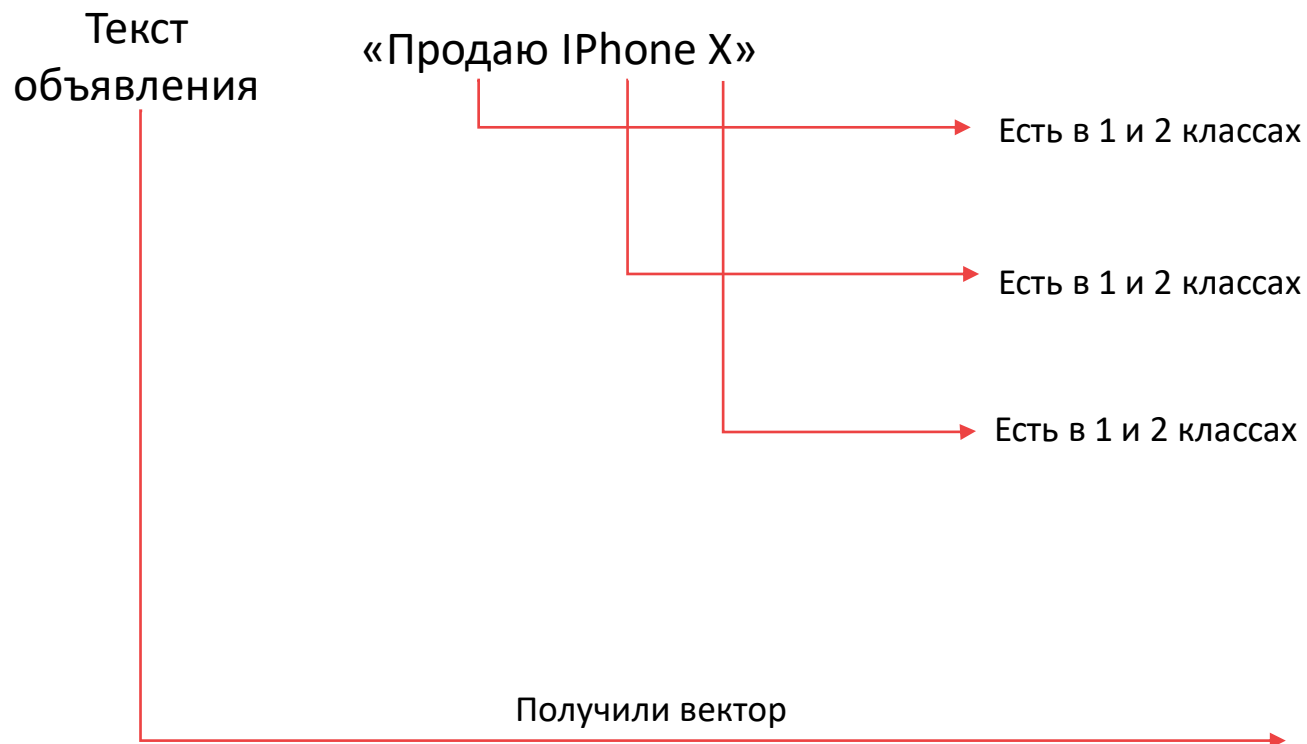
$$idf(t, D) = \log \frac{|D|}{|\{d_i \in D \mid t \in d_i\}|}$$

- $|D|$ — число документов в коллекции;
- $|\{d_i \in D \mid t \in d_i\}|$ — число документов из коллекции D , в которых встречается t

$$tf-idf(t, d, D) = tf(t, d) \times idf(t, D)$$

Future Engineering

Создадим дополнительные признаки, которые помогут улучшить качество алгоритмов.



1	2	3	54
0	0	0	0
1	1	0	0
2	1	1	0
3	1	1	0

Разделим каждый элемент вектора на сумму его значений чтобы получить частоту категории в тексте

$3 / 5, 1 / 5, 1 / 5, 0$ 0

Получили вектор в котором каждому элементу соответствует процент слов объявления соответствующей категории

Используемые алгоритмы и параметры для GridSearch

Наивный Байесовский классификатор (MultinomialNB)

Перебираемые параметры:

MultinomialNB : alpha (0.01, 0.001,)

CountVectorizer: max_df (0.5, 0.75, 1.0) -- max_features: (None, 5000, 10000, 50000) -- ngram_range: ((1, 1), (1, 2))

TfidfTransformer: use_idf: (True, False) -- norm: ('l1', 'l2')

$$P(C_k | \mathbf{x}) = \frac{p(\mathbf{x} | C_k)p(C_k)}{p(\mathbf{x})} \propto p(\mathbf{x} | C_k)p(C_k)$$

$$C_{MAP} = \arg \max_k p(C_k | \mathbf{x})$$

SGDClassifier

SGDClassifier: alpha (0.0001, 0.00001, 0.000001) -- penalty ('l2', 'elasticnet') -- n_iter (10, 50, 80)

CountVectorizer: max_df (0.5, 0.75, 1.0) -- max_features: (None, 5000, 10000, 50000) -- ngram_range: ((1, 1), (1, 2))

TfidfTransformer: use_idf: (True, False) -- norm: ('l1', 'l2')

LinearSVC

LinearSVC: alpha (0.0001, 0.00001, 0.000001) -- penalty ('l2', 'elasticnet') -- n_iter (10, 50, 80)

HashingVectorizer: n_features: (None, 5000, 10000, 50000) -- ngram_range: ((1, 1), (1, 2))

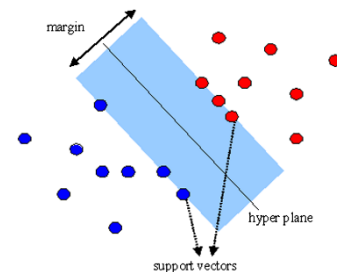
TfidfTransformer: use_idf: (True, False) -- norm: ('l1', 'l2')

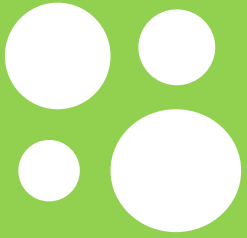
PassiveAggressiveClassifier

PassiveAggressiveClassifier: n_iter (10, 50, 80)

HashingVectorizer: ngram_range = (1, 2)

TfidfTransformer: use_idf(True, False) -- norm('l1', 'l2')





Оценка модели

Метрика качества

	$y = 1$	$y = 0$
$\hat{y} = 1$	True Positive (TP)	False Positive (FP)
$\hat{y} = 0$	False Negative (FN)	True Negative (TN)

Здесь \hat{y} — это ответ алгоритма на объекте, а y — истинная метка класса на этом объекте.
False Positive (FP).

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

В результате перебора GridSearchCV, значения Accuracy на лучших моделях на тестовой выборке (разделил train.csv в отношении 1 к 4):

Best_SGDClassifier	0.88114070926622001	'elasticnet', alpha = 0.000001
Best_MultinomialNB	0.87295309691126	alpha = 0.01
Best_LinearSVC	0.89227815002451383	loss = 'l2', tol=1e-3
Best_PassiveAggressiveClassifier	0.88728550416734764	C=1.0, n_iter=None

Возьмем модель **SGDClassifier**
с выбранными в GridSearch параметрами за
основную

Подсчет Ассигасы для алгоритма Naïve Bayes

Объявление

Ассигасу для
разных уровней
иерархии

Бытовая Электроника

Для Дома и дачи

Личные вещи

Хобби и отдых

0.9573623

Телефоны

Ноутбуки

Аудио и видео

Ремонт

Мебель и интерьер

Растения

Одежда, обувь

Часы и украшения


Спорт и отдых

Коллекционирование

Охота и рыбалка

Музыка

0.9328811



Выполнил:
Шевцов Антон
shevan05@gmail.com
+7(919) 723-72-44