**Московский государственный технический университет**

**им. Н.Э. Баумана**

**(МГТУ им. Н.Э. Баумана)**

**Радиотехнический факультет (РТ)**

Рубежный контроль №2

# «[Технологии](https://github.com/ugapanyuk/ml_course/wiki/LAB_EDA_VISUALIZATION) использования и оценки моделей машинного обучения.»

|  |  |
| --- | --- |
| Проверила: | Выполнил: |
| Преподаватель кафедры ИУ-5 | студент группы РТ5-61Б |
| Гапанюк Ю.Е. | Абраменков Г.Г. |
| Подпись: \_\_\_\_\_\_\_\_\_\_\_\_\_ | Подпись: \_\_\_\_\_\_\_\_\_\_\_\_\_ |
| «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 202\_ г. | «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 202\_ г. |

Москва, 2020

**Задание №1:**

Данный вариант выполняется на основе материалов лекции часть 1 и часть 2.

Необходимо решить задачу классификации текстов на основе любого выбранного Вами датасета (кроме примера, который рассматривался в лекции). Классификация может быть бинарной или многоклассовой. Целевой признак из выбранного Вами датасета может иметь любой физический смысл, примером является задача анализа тональности текста.

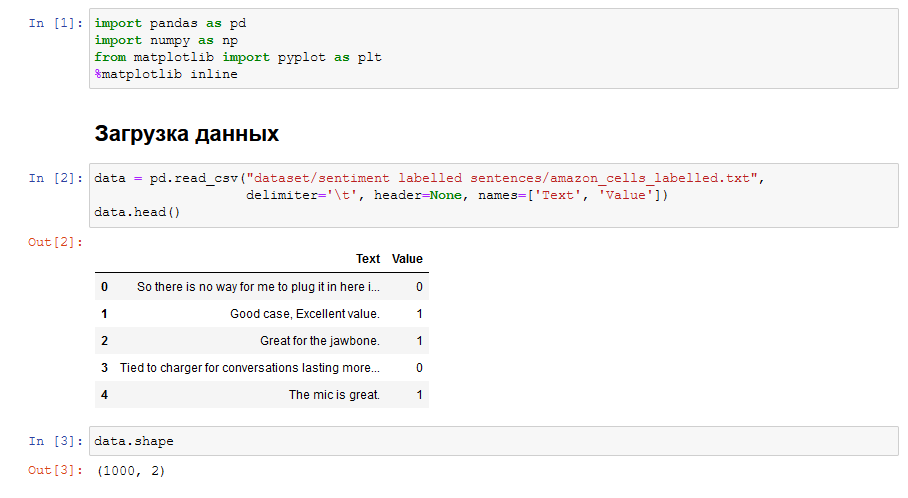
Необходимо сформировать признаки на основе CountVectorizer или TfidfVectorizer.

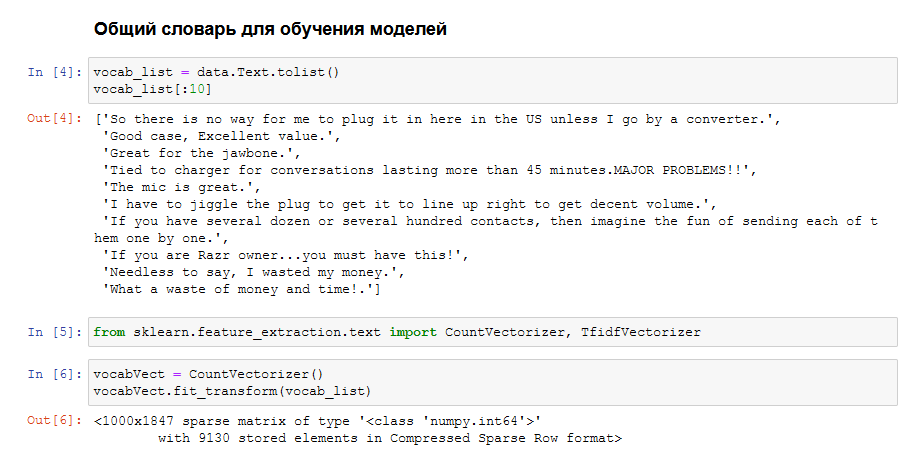
В качестве классификаторов необходимо использовать два классификатора, не относящихся к наивным Байесовским методам (например, LogisticRegression, LinearSVC), а также Multinomial Naive Bayes (MNB), Complement Naive Bayes (CNB), Bernoulli Naive Bayes.

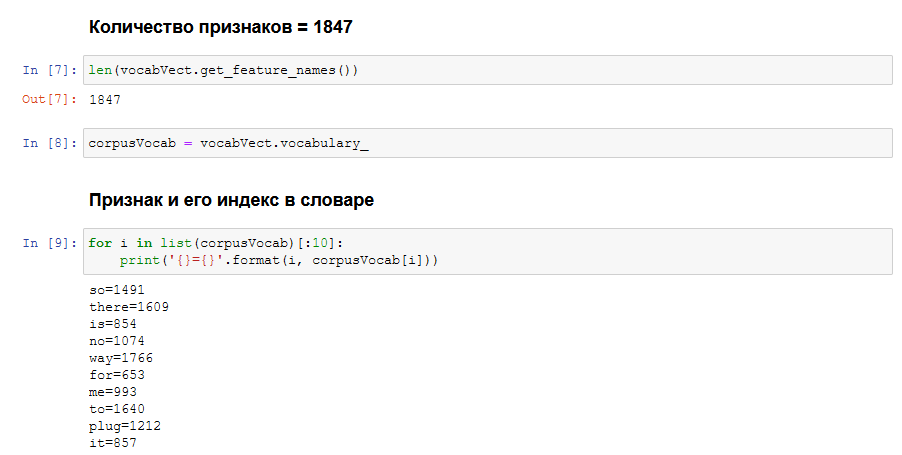
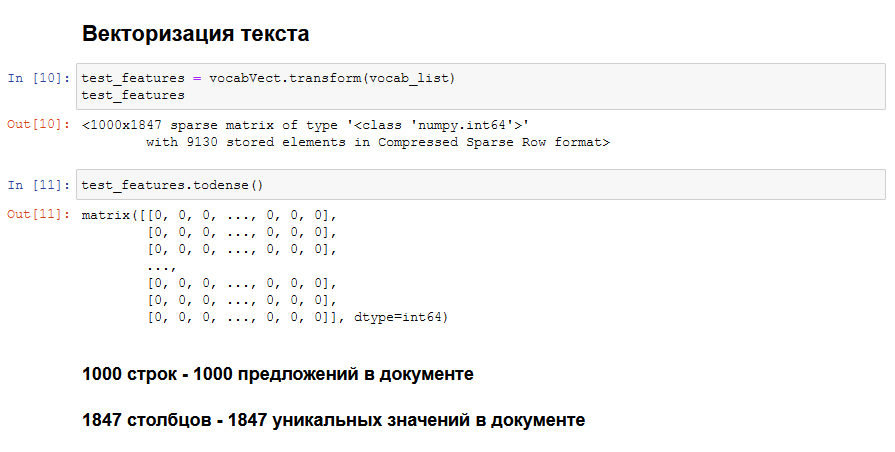
Для каждого метода необходимо оценить качество классификации с помощью хотя бы одной метрики качества классификации (например, Accuracy).

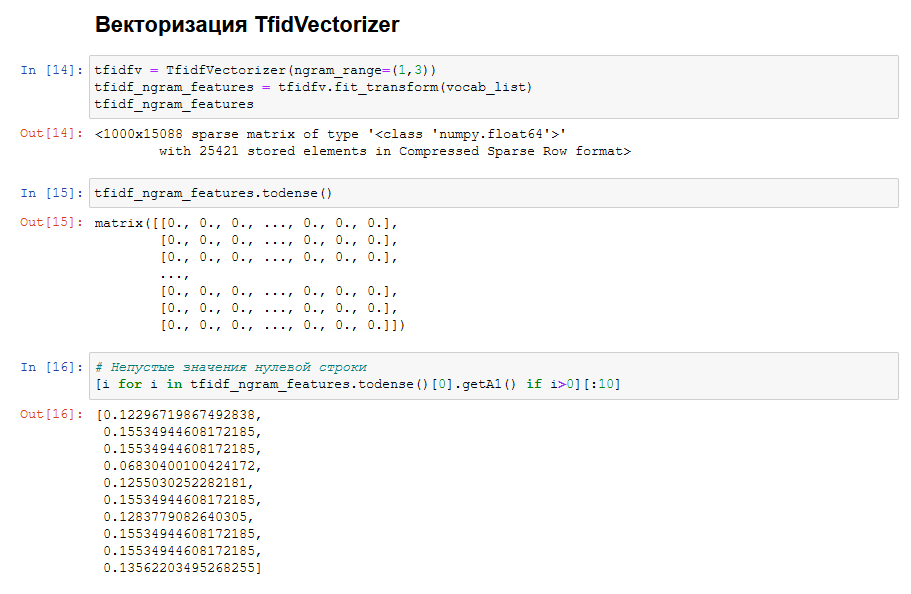
Сделать выводы о том, какой классификатор осуществляет более качественную классификацию на Вашем наборе данных.

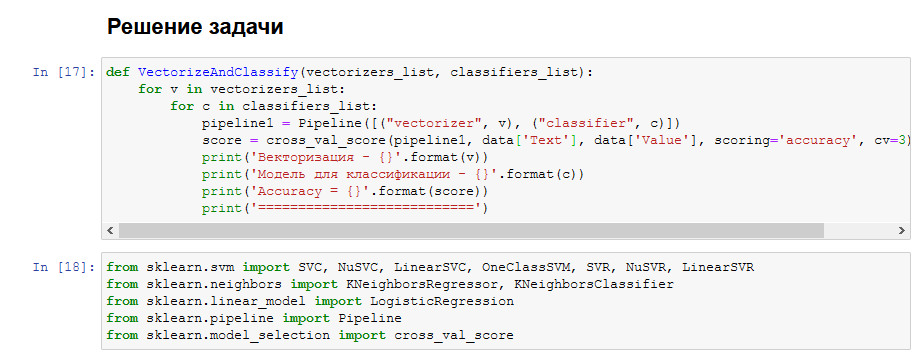
**Выполненная работа:**

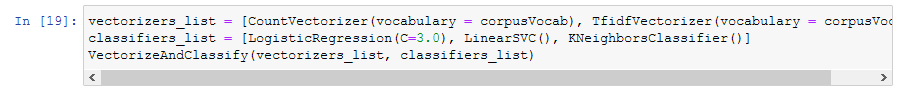








Векторизация - CountVectorizer(analyzer='word', binary=False, decode\_error='strict',

dtype=<class 'numpy.int64'>, encoding='utf-8', input='content',

lowercase=True, max\_df=1.0, max\_features=None, min\_df=1,

ngram\_range=(1, 1), preprocessor=None, stop\_words=None,

strip\_accents=None, token\_pattern='(?u)\\b\\w\\w+\\b',

tokenizer=None,

vocabulary={'10': 0, '100': 1, '11': 2, '12': 3, '13': 4,

'15': 5, '15g': 6, '18': 7, '20': 8, '2000': 9,

'2005': 10, '2160': 11, '24': 12, '2mp': 13,

'325': 14, '350': 15, '375': 16, '3o': 17, '42': 18,

'44': 19, '45': 20, '4s': 21, '50': 22, '5020': 23,

'510': 24, '5320': 25, '680': 26, '700w': 27,

'8125': 28, '8525': 29, ...})

Модель для классификации - LogisticRegression(C=3.0, class\_weight=None, dual=False, fit\_intercept=True,

intercept\_scaling=1, l1\_ratio=None, max\_iter=100,

multi\_class='auto', n\_jobs=None, penalty='l2',

random\_state=None, solver='lbfgs', tol=0.0001, verbose=0,

warm\_start=False)

Accuracy = 0.8069896243548937

===========================

Векторизация - CountVectorizer(analyzer='word', binary=False, decode\_error='strict',

dtype=<class 'numpy.int64'>, encoding='utf-8', input='content',

lowercase=True, max\_df=1.0, max\_features=None, min\_df=1,

ngram\_range=(1, 1), preprocessor=None, stop\_words=None,

strip\_accents=None, token\_pattern='(?u)\\b\\w\\w+\\b',

tokenizer=None,

vocabulary={'10': 0, '100': 1, '11': 2, '12': 3, '13': 4,

'15': 5, '15g': 6, '18': 7, '20': 8, '2000': 9,

'2005': 10, '2160': 11, '24': 12, '2mp': 13,

'325': 14, '350': 15, '375': 16, '3o': 17, '42': 18,

'44': 19, '45': 20, '4s': 21, '50': 22, '5020': 23,

'510': 24, '5320': 25, '680': 26, '700w': 27,

'8125': 28, '8525': 29, ...})

Модель для классификации - LinearSVC(C=1.0, class\_weight=None, dual=True, fit\_intercept=True,

intercept\_scaling=1, loss='squared\_hinge', max\_iter=1000,

multi\_class='ovr', penalty='l2', random\_state=None, tol=0.0001,

verbose=0)

Accuracy = 0.8249956543369716

===========================

Векторизация - CountVectorizer(analyzer='word', binary=False, decode\_error='strict',

dtype=<class 'numpy.int64'>, encoding='utf-8', input='content',

lowercase=True, max\_df=1.0, max\_features=None, min\_df=1,

ngram\_range=(1, 1), preprocessor=None, stop\_words=None,

strip\_accents=None, token\_pattern='(?u)\\b\\w\\w+\\b',

tokenizer=None,

vocabulary={'10': 0, '100': 1, '11': 2, '12': 3, '13': 4,

'15': 5, '15g': 6, '18': 7, '20': 8, '2000': 9,

'2005': 10, '2160': 11, '24': 12, '2mp': 13,

'325': 14, '350': 15, '375': 16, '3o': 17, '42': 18,

'44': 19, '45': 20, '4s': 21, '50': 22, '5020': 23,

'510': 24, '5320': 25, '680': 26, '700w': 27,

'8125': 28, '8525': 29, ...})

Модель для классификации - KNeighborsClassifier(algorithm='auto', leaf\_size=30, metric='minkowski',

metric\_params=None, n\_jobs=None, n\_neighbors=5, p=2,

weights='uniform')

Accuracy = 0.6619733505961051

===========================

Векторизация - TfidfVectorizer(analyzer='word', binary=False, decode\_error='strict',

dtype=<class 'numpy.float64'>, encoding='utf-8',

input='content', lowercase=True, max\_df=1.0, max\_features=None,

min\_df=1, ngram\_range=(1, 1), norm='l2', preprocessor=None,

smooth\_idf=True, stop\_words=None, strip\_accents=None,

sublinear\_tf=False, token\_pattern='(?u)\\b\\w\\w+\\b',

tokenizer=None, use\_idf=True,

vocabulary={'10': 0, '100': 1, '11': 2, '12': 3, '13': 4,

'15': 5, '15g': 6, '18': 7, '20': 8, '2000': 9,

'2005': 10, '2160': 11, '24': 12, '2mp': 13,

'325': 14, '350': 15, '375': 16, '3o': 17, '42': 18,

'44': 19, '45': 20, '4s': 21, '50': 22, '5020': 23,

'510': 24, '5320': 25, '680': 26, '700w': 27,

'8125': 28, '8525': 29, ...})

Модель для классификации - LogisticRegression(C=3.0, class\_weight=None, dual=False, fit\_intercept=True,

intercept\_scaling=1, l1\_ratio=None, max\_iter=100,

multi\_class='auto', n\_jobs=None, penalty='l2',

random\_state=None, solver='lbfgs', tol=0.0001, verbose=0,

warm\_start=False)

Accuracy = 0.8109936283588978

===========================

Векторизация - TfidfVectorizer(analyzer='word', binary=False, decode\_error='strict',

dtype=<class 'numpy.float64'>, encoding='utf-8',

input='content', lowercase=True, max\_df=1.0, max\_features=None,

min\_df=1, ngram\_range=(1, 1), norm='l2', preprocessor=None,

smooth\_idf=True, stop\_words=None, strip\_accents=None,

sublinear\_tf=False, token\_pattern='(?u)\\b\\w\\w+\\b',

tokenizer=None, use\_idf=True,

vocabulary={'10': 0, '100': 1, '11': 2, '12': 3, '13': 4,

'15': 5, '15g': 6, '18': 7, '20': 8, '2000': 9,

'2005': 10, '2160': 11, '24': 12, '2mp': 13,

'325': 14, '350': 15, '375': 16, '3o': 17, '42': 18,

'44': 19, '45': 20, '4s': 21, '50': 22, '5020': 23,

'510': 24, '5320': 25, '680': 26, '700w': 27,

'8125': 28, '8525': 29, ...})

Модель для классификации - LinearSVC(C=1.0, class\_weight=None, dual=True, fit\_intercept=True,

intercept\_scaling=1, loss='squared\_hinge', max\_iter=1000,

multi\_class='ovr', penalty='l2', random\_state=None, tol=0.0001,

verbose=0)

Accuracy = 0.8109816403229577

===========================

Векторизация - TfidfVectorizer(analyzer='word', binary=False, decode\_error='strict',

dtype=<class 'numpy.float64'>, encoding='utf-8',

input='content', lowercase=True, max\_df=1.0, max\_features=None,

min\_df=1, ngram\_range=(1, 1), norm='l2', preprocessor=None,

smooth\_idf=True, stop\_words=None, strip\_accents=None,

sublinear\_tf=False, token\_pattern='(?u)\\b\\w\\w+\\b',

tokenizer=None, use\_idf=True,

vocabulary={'10': 0, '100': 1, '11': 2, '12': 3, '13': 4,

'15': 5, '15g': 6, '18': 7, '20': 8, '2000': 9,

'2005': 10, '2160': 11, '24': 12, '2mp': 13,

'325': 14, '350': 15, '375': 16, '3o': 17, '42': 18,

'44': 19, '45': 20, '4s': 21, '50': 22, '5020': 23,

'510': 24, '5320': 25, '680': 26, '700w': 27,

'8125': 28, '8525': 29, ...})

Модель для классификации - KNeighborsClassifier(algorithm='auto', leaf\_size=30, metric='minkowski',

metric\_params=None, n\_jobs=None, n\_neighbors=5, p=2,

weights='uniform')

Accuracy = 0.7709895524266782

===========================

