



Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ _____ Информатика и системы управления _____
КАФЕДРА _____ Системы обработки информации и управления _____

Рубежный контроль №2

«Классификация текстов на основе методов наивного Байеса.»
по курсу «Технологии машинного обучения»

Выполнил:

Калин
РТ5-61

Проверил:
Преподаватель кафедры ИУ5
Гапанюк Ю.Е.

1. Задание

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`).

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

2. Выполнение рубежного контроля

Подключим необходимые библиотеки и загрузим набор данных

```
[22] from sklearn.linear_model import LogisticRegression
      from sklearn.svm import LinearSVC
      from sklearn.naive_bayes import MultinomialNB, ComplementNB, BernoulliNB
      from sklearn.metrics import accuracy_score
      from sklearn.datasets import fetch_20newsgroups
      from sklearn.feature_extraction.text import TfidfVectorizer
      import pandas as pd
      import seaborn as sns
      import matplotlib.pyplot as plt

      %matplotlib inline

      # Устанавливаем тип графиков
      sns.set(style="ticks")

      # Для лучшего качества графиков
      from IPython.display import set_matplotlib_formats
      set_matplotlib_formats("retina")

      # Устанавливаем ширину экрана для отчета
      pd.set_option("display.width", 70)

      # Загружаем данные
      data_train = fetch_20newsgroups(subset='train', remove=('headers', 'footers'))
      data_test = fetch_20newsgroups(subset='test', remove=('headers', 'footers'))
```

```
[23] data_train.target.shape
```

```
↳ (11314,)
```

```
[24] data_train.data[:3]
```

```
↳ ['I was wondering if anyone out there could enlighten me on this car I saw\nthe
"A fair number of brave souls who upgraded their SI clock oscillator have\nshar
'well folks, my mac plus finally gave up the ghost this weekend after\nstarting
```



```
[25] vectorizer = TfidfVectorizer()
      vectorizer.fit(data_train.data + data_test.data)
```

```
↳ 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=None)
```

```
[26] X_train = vectorizer.transform(data_train.data)
      X_test = vectorizer.transform(data_test.data)

      y_train = data_train.target
      y_test = data_test.target
```

```
[27] X_train
↳ <11314x152843 sparse matrix of type '<class 'numpy.float64'>'
   with 1467517 stored elements in Compressed Sparse Row format>
```

```
[28] X_test
↳ <7532x152843 sparse matrix of type '<class 'numpy.float64'>'
   with 951914 stored elements in Compressed Sparse Row format>
```

```
[29] def test(model):
      print(model)
      model.fit(X_train, y_train)
      print("accuracy:", accuracy_score(y_test, model.predict(X_test)))
```

```
[30] test(LogisticRegression(solver='lbfgs', multi_class='auto'))
↳ LogisticRegression(C=1.0, 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: 0.774429102496017
```

```
[31] test(LinearSVC())
↳ 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: 0.8048327137546468
```

```
[32] test(MultinomialNB())
↳ MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)
accuracy: 0.72623473181094
```

```
[33] test(ComplementNB())
↳ ComplementNB(alpha=1.0, class_prior=None, fit_prior=True, norm=False)
accuracy: 0.8089484864577802
```

```
[34] test(BernoulliNB())
↳ BernoulliNB(alpha=1.0, binarize=0.0, class_prior=None, fit_prior=True)
accuracy: 0.5371747211895911
```

Вывод: Видим, что метод Complement Naive Bayes лучше всего решает поставленную задачу многоклассовой классификации в условиях дисбаланса классов, но стоит отметить, что LinearSVC также показал отличный результат и показал такой же результат, как метод Complement Naive Bayes.