МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
им. Н.Э. Баумана

Факультет «Информатика и системы управления»
Кафедра «Систем обработки информации и управления»

ОТЧЕТ

Рубежный контроль №__2__
по дисциплине «Методы машинного обучения»

ИСПОЛНИТЕЛЬ: Егоров С.А.
ФИО
группа ИУ5-22М
_____
подпись
"__"_____2020 г.

ПРЕПОДАВАТЕЛЬ: Гапанюк Ю.Е.
ФИО
_____
подпись
"__"_____2020 г.

Москва - 2020

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 двух метрик качества классификации (например, Accuracy, ROC-AUC).

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

Реализация задания

Подключение используемых библиотек и набора данных:

```
import numpy as np
from sklearn.datasets import fetch_20newsgroups
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics import accuracy_score, confusion_matrix, plot_confusion_matrix
from sklearn.svm import LinearSVC
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB, ComplementNB, BernoulliNB
import matplotlib.pyplot as plt
```

```
newsgroups_train = fetch_20newsgroups(subset='train', remove=('headers', 'footers'))
newsgroups_test = fetch_20newsgroups(subset='test', remove=('headers', 'footers'))
```

Для формирования признаков я выбрал TfidfVectorizer:

```
vectorizer = TfidfVectorizer()
vectorizer.fit(newsgroups_train.data + newsgroups_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)
```

```
X_train = vectorizer.transform(newsgroups_train.data)
X_test = vectorizer.transform(newsgroups_test.data)

y_train = newsgroups_train.target
y_test = newsgroups_test.target
```

Создадим функцию для оценки каждого классификатара, а в качестве метрик оценки точности возьмём Accuracy и Confusion_matrix:

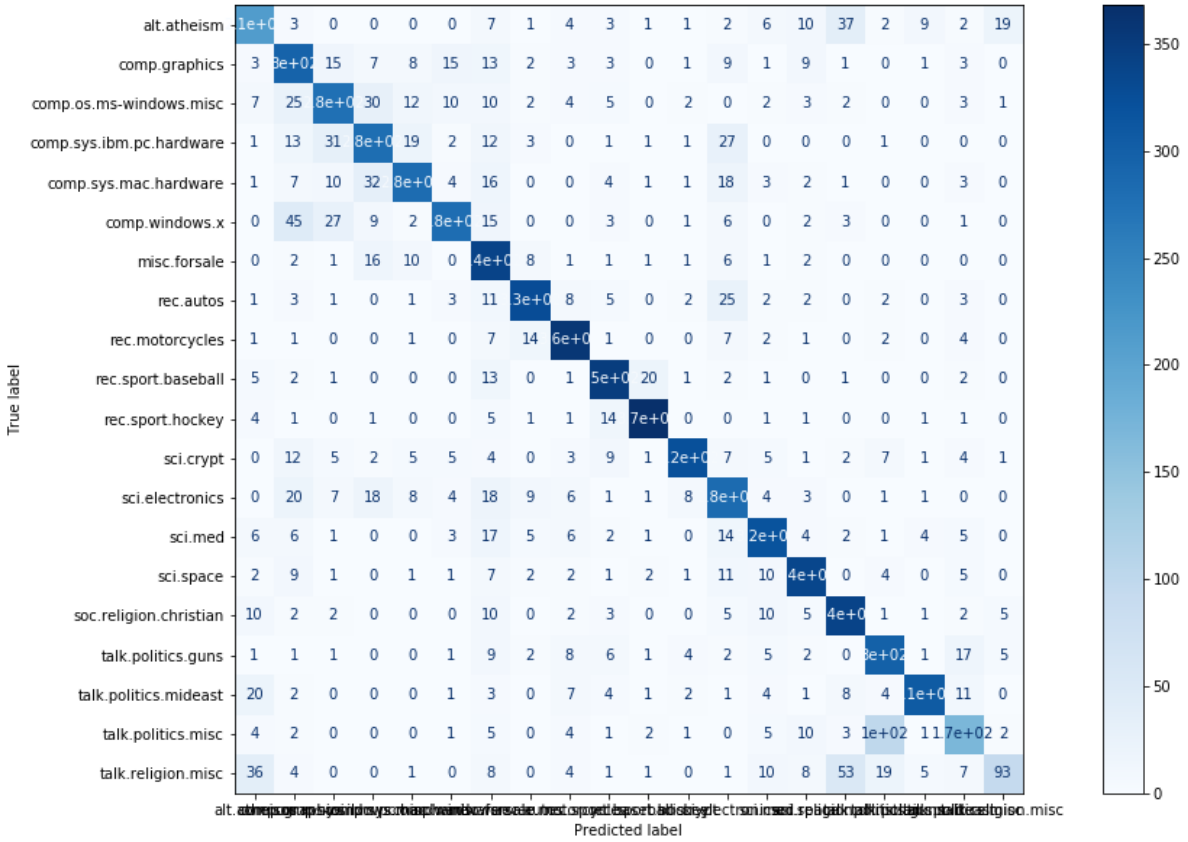
```
def test(model,ax):
    print(model)
    model.fit(X_train, y_train)
    print("Accuracy:", accuracy_score(y_test, model.predict(X_test)))
    #print("Confusion_matrix:", confusion_matrix(y_test, model.predict(X_test), labels = np.unique(model.predict(X_test))))
    plot_confusion_matrix(model, X_test, y_test,
                          display_labels=newsgroups_test.target_names,
                          cmap=plt.cm.Blues, ax=ax)
```

Классификатор LogisticRegression:

```
fig, ax = plt.subplots(figsize=(20,10))
test(LogisticRegression(), ax)
```

Результат:

```
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
```

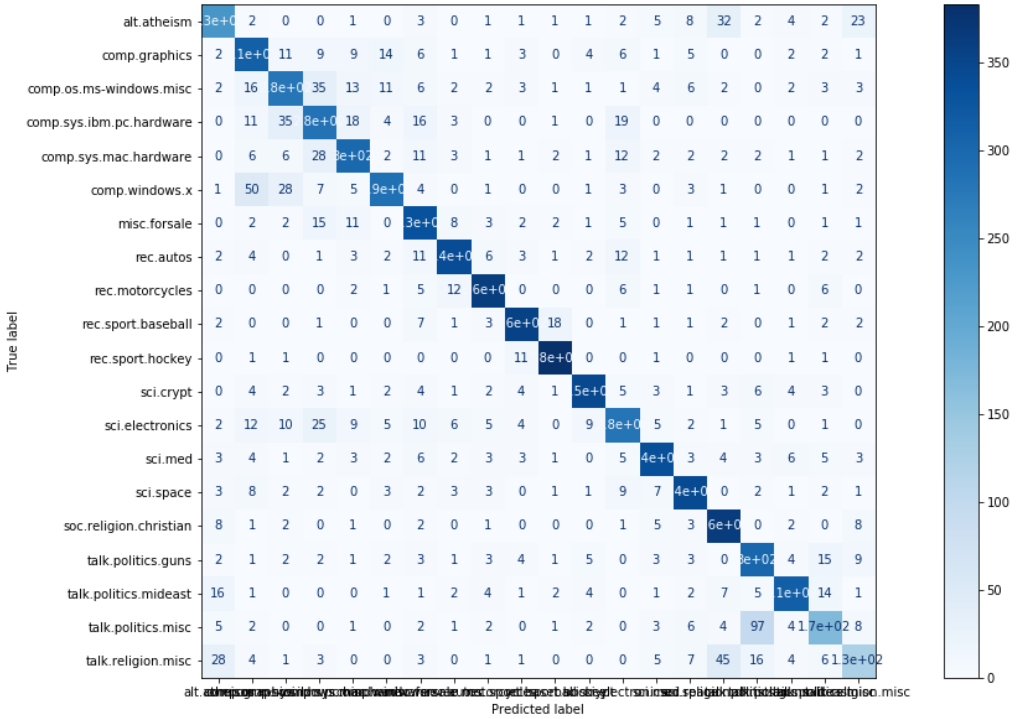


Классификатор LinearSVC:

```
fig, ax = plt.subplots(figsize=(20,10))
test(LinearSVC(), ax)
```

Результат:

```
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
```

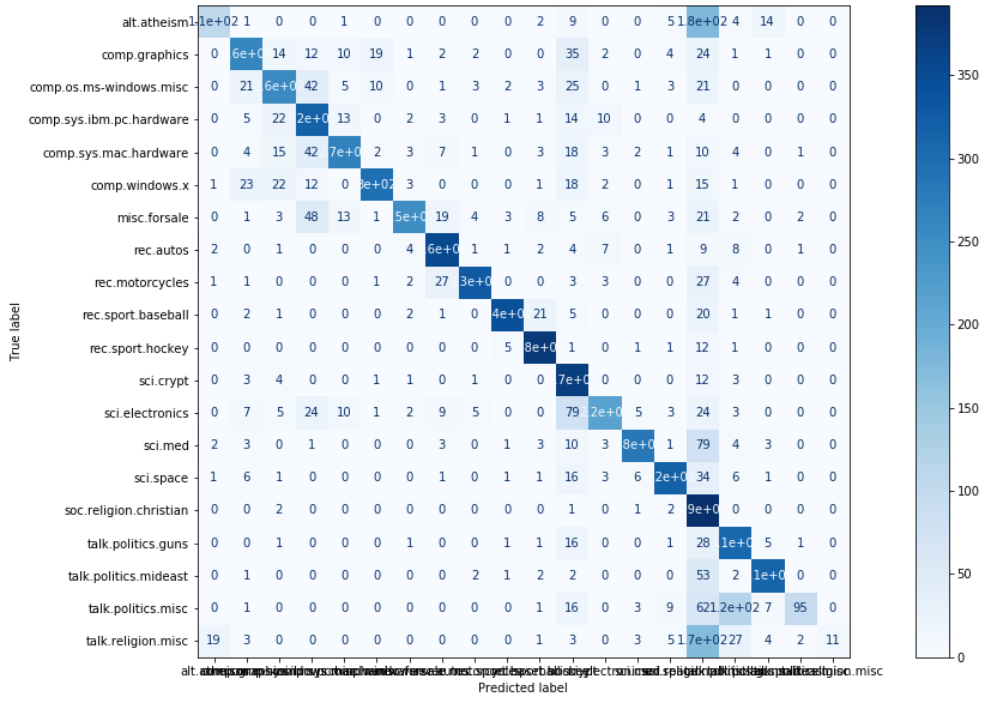


Классификатор Multinomial Naive Bayes:

```
fig, ax = plt.subplots(figsize=(20,10))
test(MultinomialNB(),ax)
```

Результат:

```
MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)
Accuracy: 0.72623473181094
```

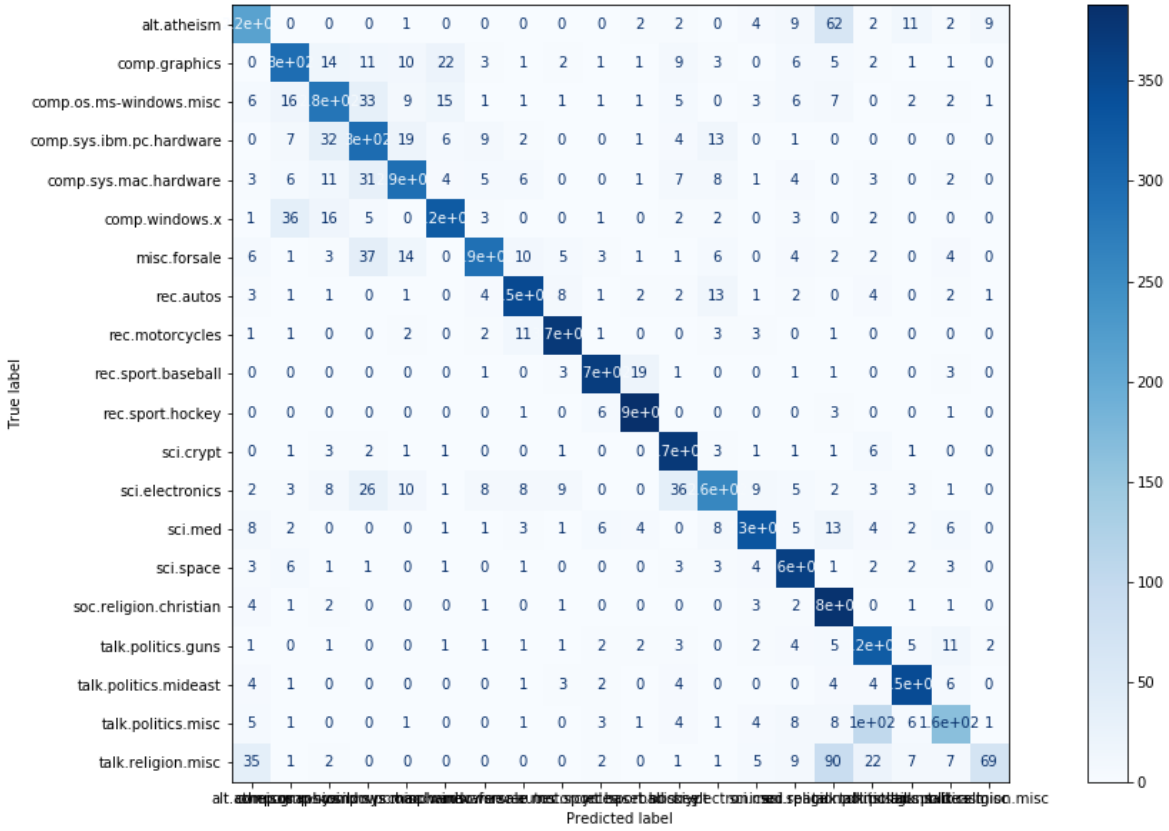


Классификатор Complement Naive Bayes:

```
fig, ax = plt.subplots(figsize=(20,10))
test(ComplementNB(),ax)
```

Результат:

```
ComplementNB(alpha=1.0, class_prior=None, fit_prior=True, norm=False)
Accuracy: 0.8089484864577802
```

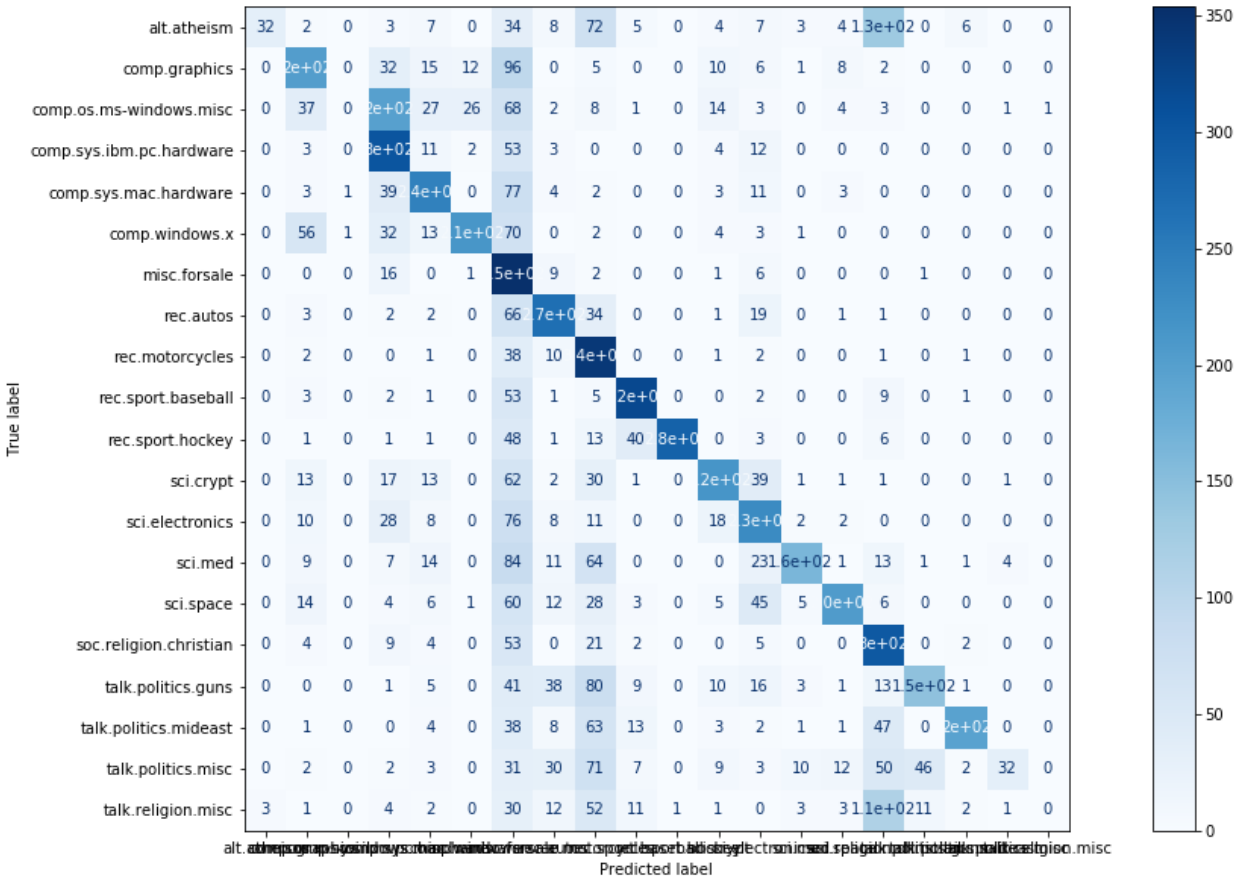


Классификатор Bernoulli Naive Bayes:

```
fig, ax = plt.subplots(figsize=(20,10))
test(BernoulliNB(),ax)
```

Результат:

```
BernoulliNB(alpha=1.0, binarize=0.0, class_prior=None, fit_prior=True)
Accuracy: 0.5371747211895911
```



Выводы

Для выбранного набора данных более качественная классификация была получена при помощи метода Complement Naive Bayes.