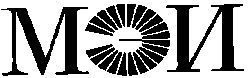
ФГБОУ ВО

Национальный исследовательский университет "МЭИ"



mpei_1

Кафедра Релейной Защиты и Автоматизации Энергосистем

**Лабораторная работа № 2**

**Наивный Байесовский классификатор. Метод опорных векторов**

|  |  |
| --- | --- |
| Выполнили: | Домрачев В.А.  Егоров Д.В.  Чернышева Н. А. |
| Группа: | Э-13м-22 |
| Проверил: | Дегтярев Д.А. |

**Москва 2023**

Целью работы является получение практических навыков работы с моделями байесовского классификатора и метода опорных векторов в программной среде Python.

Листинг код приведен в приложении 1.

Результаты работы каждого пункта задания в виде графиков Matplolib с подписанными осями:

Круговая диаграмма для принимаемых значений целевой переменной представлена на рисунке 1.

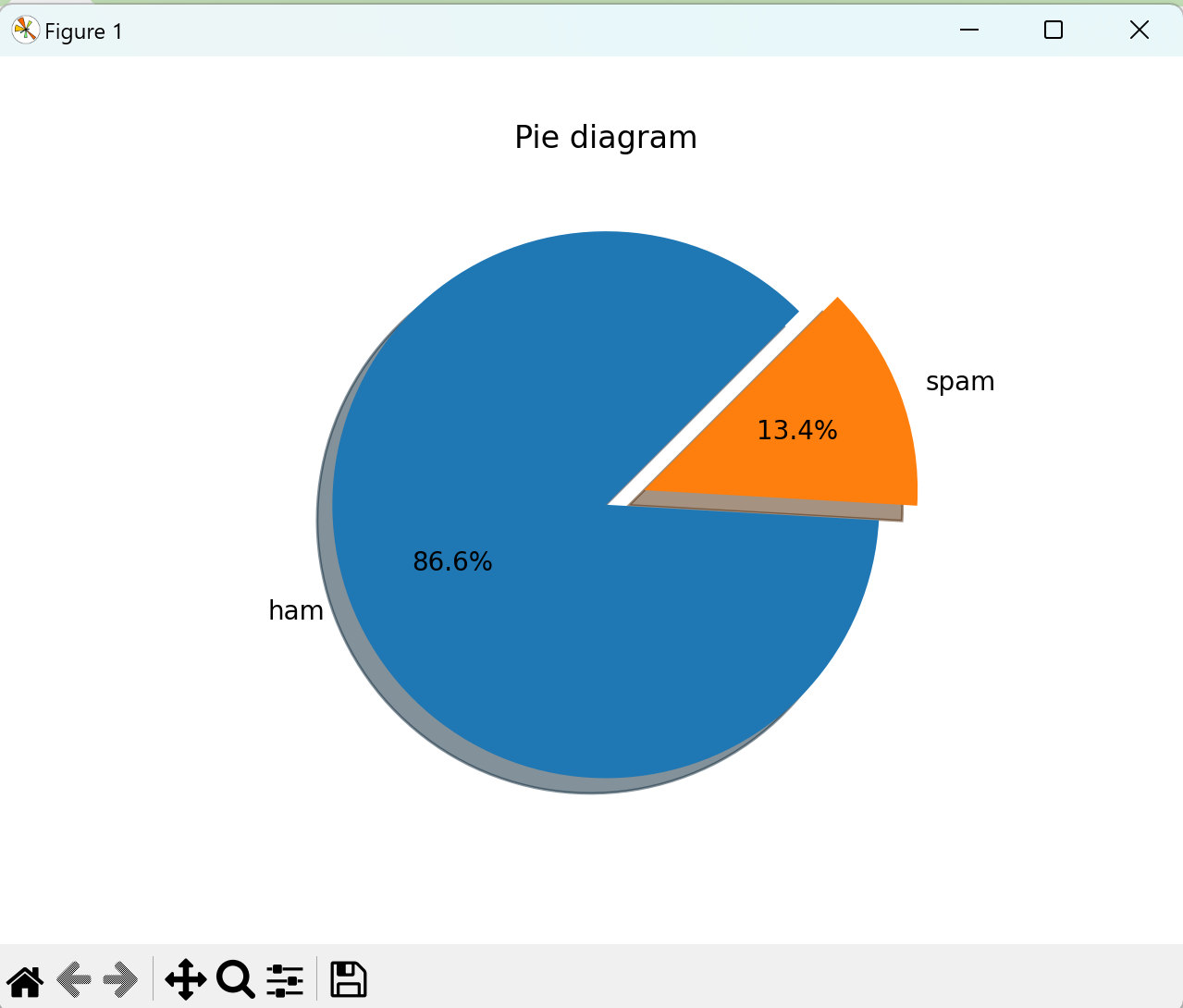


Рис. 1. Круговая диаграмма

Столбиковая диаграмма для двадцати наиболее часто встречающихся слов в классе ham представлена на рисунке 2.



Рис. 2. Столбиковая диаграмма для класса ham

Столбиковая диаграмма для двадцати наиболее часто встречающихся слов в классе spam представлена на рисунке 3.



Рис. 3. Столбиковая диаграмма для класса spam

Зависимость метрики accuracy на обучающих и тестовых данных от варьируемого параметра alpha представлена на рисунке 4.

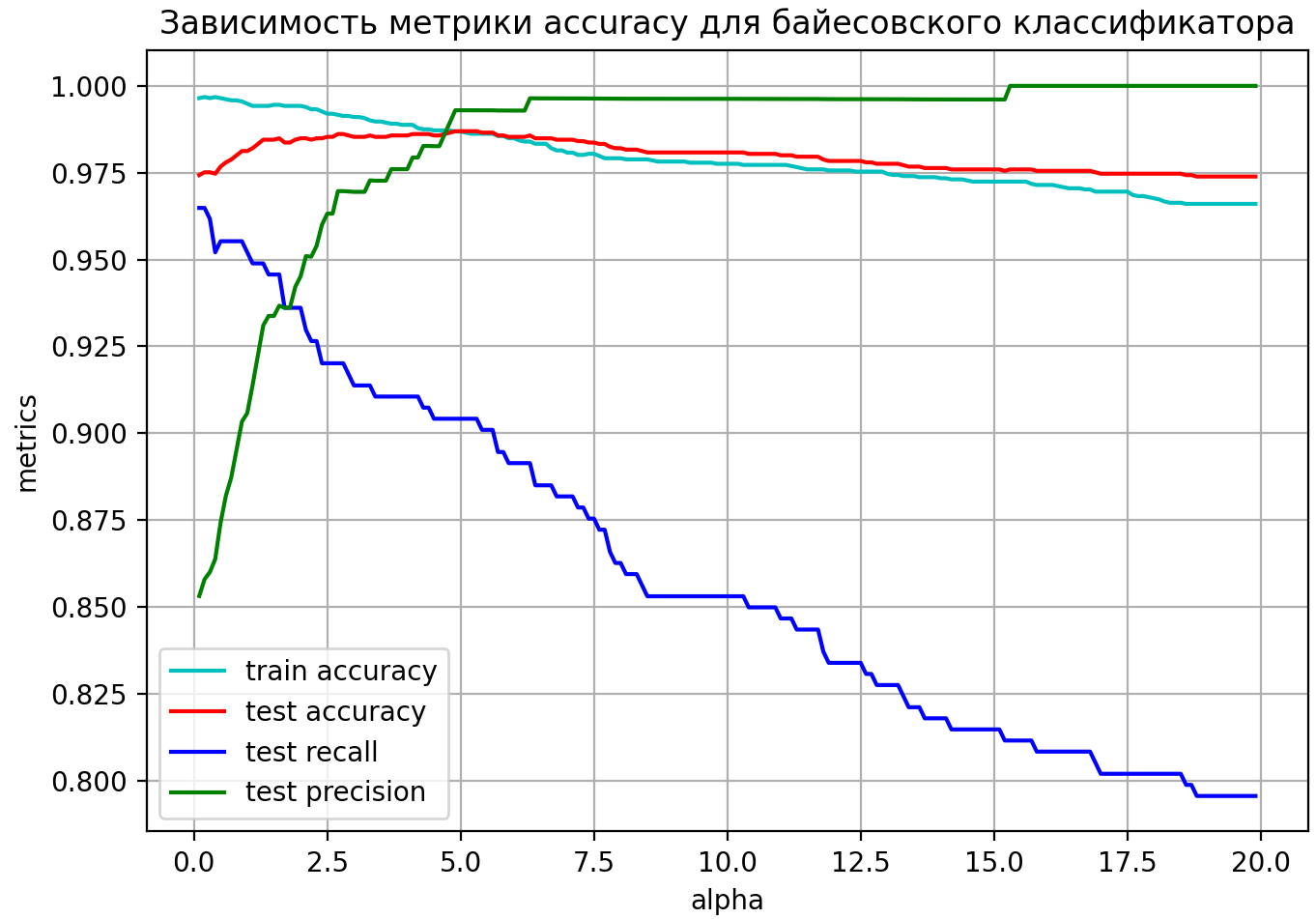


Рис. 4. Зависимость метрики accuracy на обучающих и тестовых данных

ROC-кривая с оптимальным параметром alpha представлена на рисунке 5.

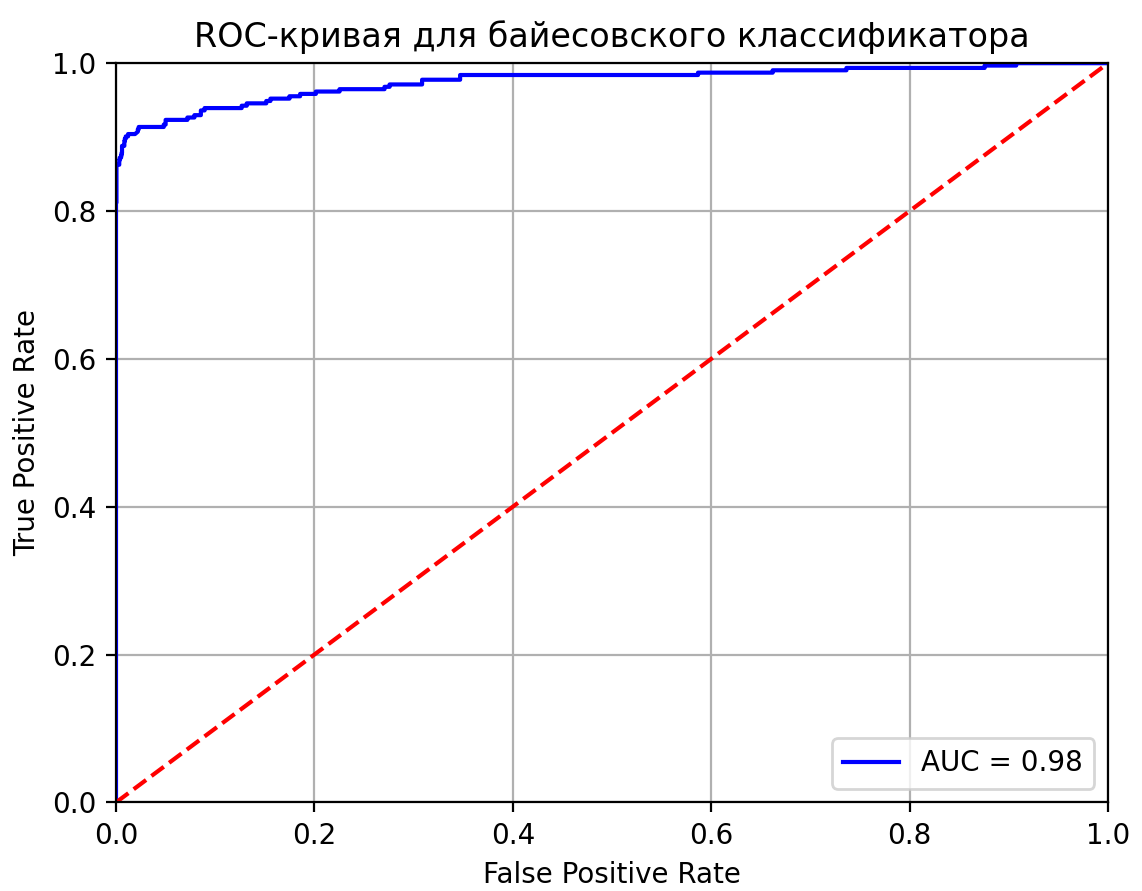


Рис. 5. ROC-кривая с оптимальным параметром alpha

Зависимость метрики accuracy на обучающих и тестовых данных от варьируемого параметра регуляризатора С представлена на рисунке 6.

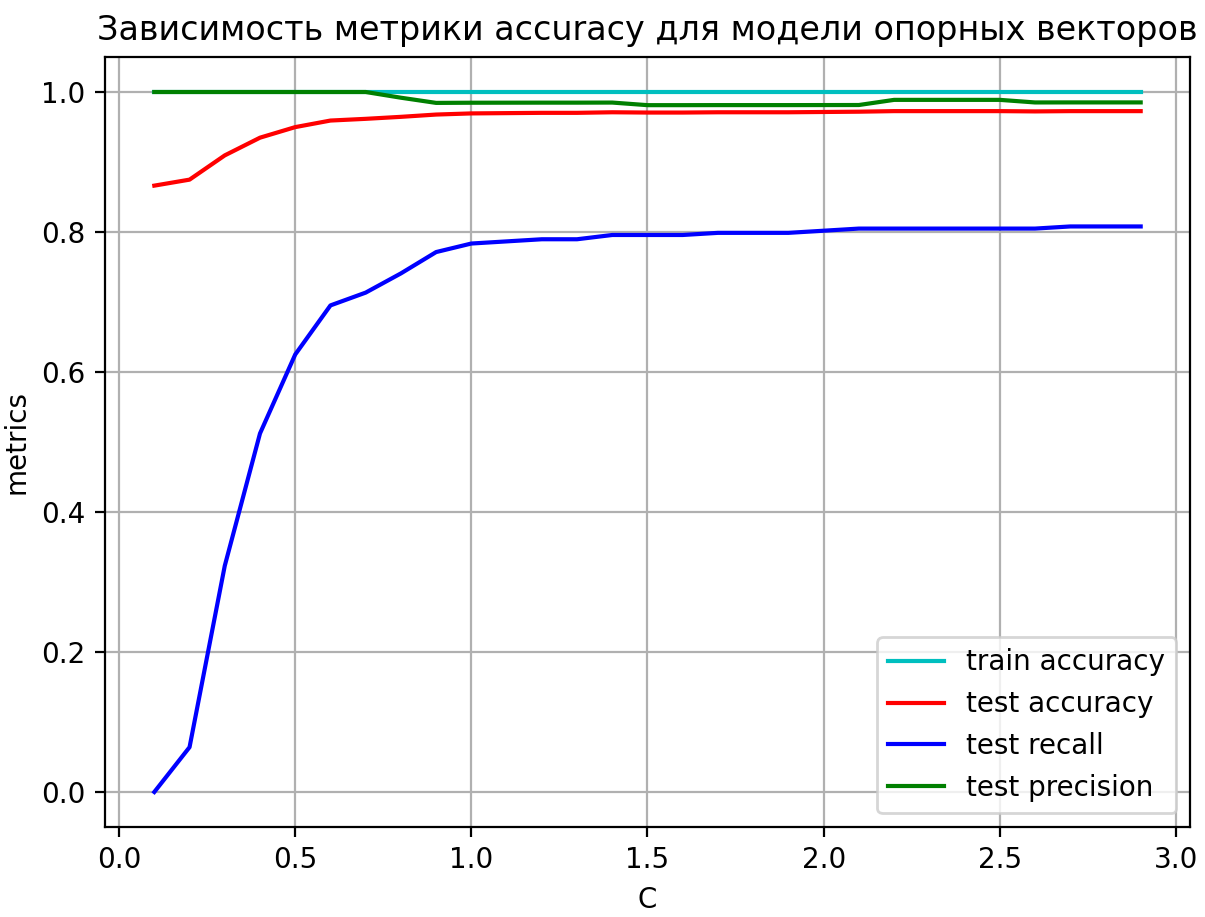
ы

Рис. 6. Зависимость метрики accuracy на обучающих и тестовых данных

ROC-кривая с оптимальным параметром регуляризатора С представлена на рисунке 7.

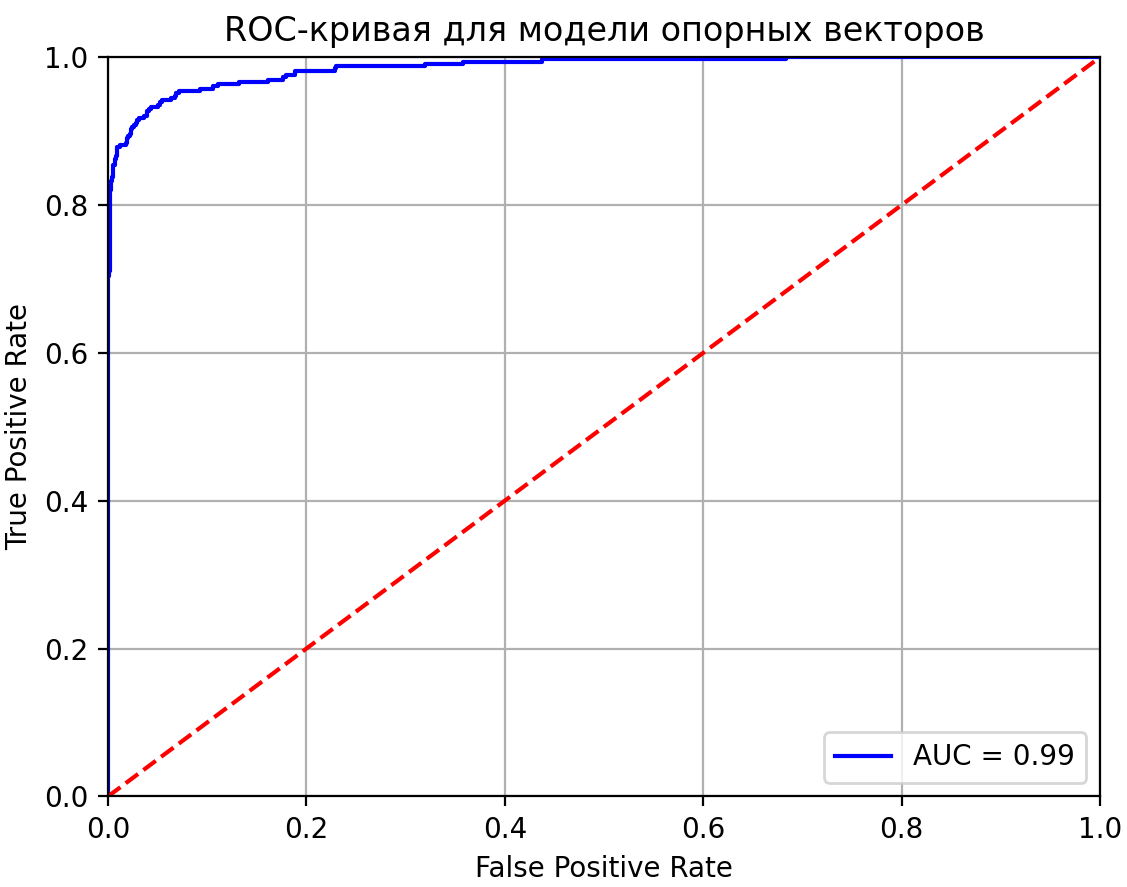


Рис. 7. ROC-кривая с оптимальным параметром регуляризатора С

Вывод: в ходе лабораторной работы было выполнено:

1. Анализ, предварительная предобработка и визуализация данных;
2. Обучение и применение наивного байесовского классификатора и модели опорных векторов с оптимальным параметром;
3. Построение и визуализация матрицы ошибок и ROC-кривой для моделей с оптимальными выбранными параметрами, а также расчет метрики AUC-ROC.

Приложение 1.

Листинг кода.

import pandas as pd  
import matplotlib.pyplot as plt  
from collections import Counter  
import numpy as np  
from sklearn import feature\_extraction  
from sklearn import model\_selection  
from sklearn.naive\_bayes import MultinomialNB  
from sklearn import metrics  
from sklearn.metrics import confusion\_matrix as confusion\_matrix\_metric  
from sklearn.svm import SVC  
  
# *TODO: 1. Загрузить анализируемые данные, выданные преподавателем*# ham - real msg; smap - fake msg  
data = pd.read\_csv('spam.csv', encoding="latin-1").drop(['Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'], axis=1)  
  
# *TODO: 2. Построить круговую диаграмму для принимаемых значений целевой переменной*explode = (0, 0.15)  
target = pd.Series(data['v1']).value\_counts()  
target.plot(kind='pie', autopct='%1.1f%%', shadow=True, startangle=45, explode=explode)  
plt.title('Pie diagram')  
plt.ylabel('')  
  
# *TODO: 3. Построить столбиковую диаграмму для двадцати наиболее часто встречающихся слов в обоих классах*ham\_words = Counter(''.join(data[data['v1'] == 'ham']['v2']).split()).most\_common(20)  
spam\_words = Counter(''.join(data[data['v1'] == 'spam']['v2']).split()).most\_common(20)  
df\_ham\_words = pd.DataFrame(ham\_words)  
df\_ham\_words = df\_ham\_words.rename(columns={0: 'Слова в не спаме', 1: 'count'})  
  
df\_spam\_words = pd.DataFrame(spam\_words)  
df\_spam\_words = df\_spam\_words.rename(columns={0: 'Слова в спаме', 1: 'count'})  
  
y\_pos = np.arange(len(df\_ham\_words['Слова в не спаме']))  
df\_ham\_words.plot.bar(legend=False, title='Самые часто встречаемые слова в не спам сообщениях', xlabel='words', ylabel='number')  
plt.xticks(y\_pos, df\_ham\_words['Слова в не спаме'])  
plt.tight\_layout()  
  
y\_pos = np.arange(len(df\_spam\_words['Слова в спаме']))  
df\_spam\_words.plot.bar(legend=False, title='Самые часто встречаемые слова в спам сообщениях', xlabel='words', ylabel='number')  
plt.xticks(y\_pos, df\_spam\_words['Слова в спаме'])  
plt.tight\_layout()  
  
# *TODO: 4. Выполнить токенизацию текстового признака, исключив неинформативные часто встречающиеся слова*tokenizer = feature\_extraction.text.CountVectorizer(stop\_words='english')  
X = tokenizer.fit\_transform(data['v2'])  
  
# *TODO: 5. Найти оптимальный параметр сглаживания alpha для наивного байесовского классификатора по метрикам*# *precision и accuracy.*train\_score = []  
test\_recall = []  
test\_precision = []  
test\_score = []  
data['v1'] = data['v1'].map({'spam': 1, 'ham': 0})  
X\_train, X\_test, Y\_train, Y\_test = model\_selection.train\_test\_split(X, data['v1'], test\_size=0.44)  
alpha\_range = np.arange(0.1, 20, 0.1)  
# Попробуйте найти оптимальный параметр alpha в диапазоне от 0,1 до 20 с шагом 0,1:  
for alpha in alpha\_range:  
 mnb = MultinomialNB(alpha=alpha).fit(X\_train, Y\_train)  
 y\_train = mnb.predict(X\_train)  
 y\_test = mnb.predict(X\_test)  
 train\_score.append(metrics.accuracy\_score(Y\_train, y\_train))  
 test\_recall.append(metrics.recall\_score(Y\_test, y\_test))  
 test\_precision.append(metrics.precision\_score(Y\_test, y\_test))  
 test\_score.append(metrics.accuracy\_score(Y\_test, y\_test))  
 y\_train = 0  
 y\_test = 0  
  
matrix = np.matrix(np.c\_[alpha\_range, train\_score, test\_score, test\_recall, test\_precision])  
models = pd.DataFrame(data=matrix,  
 columns=['alpha', 'train accuracy', 'test accuracy', 'test recall', 'test precision'])  
  
# *TODO: 6. Построить зависимость метрики accuracy на обучающих и тестовых данных от варьируемого параметра*f = plt.figure(figsize=(7, 5))  
plt.plot(alpha\_range, models['train accuracy'], "c", label='train accuracy')  
plt.plot(alpha\_range, models['test accuracy'], "r", label='test accuracy')  
plt.plot(alpha\_range, models['test recall'], "b", label='test recall')  
plt.plot(alpha\_range, models['test precision'], "g", label='test precision')  
plt.ylabel('metrics')  
plt.xlabel('alpha')  
plt.title('Зависимость метрики accuracy для байесовского классификатора')  
plt.legend()  
plt.grid()  
plt.tight\_layout()  
plt.show()  
  
# *TODO: Построить матрицы ошибок для модели с оптимальным выбранным параметром 2*best\_alpha = models['alpha'][models['test precision'].idxmax()]  
print("Best alpha: ", best\_alpha, '\n')  
mnb = MultinomialNB(alpha=best\_alpha).fit(X\_train, Y\_train)  
  
print("Confusion matrix for MultinomialNB:")  
confusion\_matrix = confusion\_matrix\_metric(Y\_test, mnb.predict(X\_test))  
print(pd.DataFrame(data=confusion\_matrix, columns=['predicted ham', 'predicted spam'],  
 index=['actual ham', 'actual spam']))  
  
# *TODO: 7. Построить ROC-кривую и рассчитать метрику AUC-ROC*y\_pred\_pr = mnb.predict\_proba(X\_test)[:, 1]  
fpr, tpr, threshold = metrics.roc\_curve(Y\_test, y\_pred\_pr)  
roc\_auc = metrics.auc(fpr, tpr)  
plt.title('Receiver Operating Characteristic')  
plt.plot(fpr, tpr, 'b', label='AUC = %0.2f' % roc\_auc)  
plt.legend(loc='lower right')  
plt.plot([0, 1], [0, 1], 'r--')  
plt.xlim([0, 1])  
plt.ylim([0, 1])  
plt.ylabel('True Positive Rate')  
plt.xlabel('False Positive Rate')  
plt.title('ROC-кривая для байесовского классификатора')  
plt.grid()  
plt.show()  
  
# *TODO: 8. Найти оптимальный параметр регуляризатора С для модели опорных векторов по метрикам precision и accuracy*c\_range = np.arange(0.1, 3, 0.1)  
train\_score = []  
test\_recall = []  
test\_precision = []  
test\_score = []  
# Попробуйте найти оптимальный параметр c в диапазоне от 0,1 до 3 с шагом 0,1:  
for C in c\_range:  
 svc = SVC(C=C).fit(X\_train, Y\_train)  
 y\_train = svc.predict(X\_train)  
 y\_test = svc.predict(X\_test)  
 train\_score.append(metrics.accuracy\_score(Y\_train, Y\_train))  
 test\_recall.append(metrics.recall\_score(Y\_test, y\_test))  
 test\_precision.append(metrics.precision\_score(Y\_test, y\_test, zero\_division=1))  
 test\_score.append(metrics.accuracy\_score(Y\_test, y\_test))  
  
  
# *TODO: 9. Повторить пункты 6 и 7 для модели опорных векторов*matrix = np.matrix(np.c\_[c\_range, train\_score, test\_score, test\_recall, test\_precision])  
models = pd.DataFrame(data=matrix, columns=['C', 'train accuracy', 'test accuracy', 'test recall', 'test precision'])  
f = plt.figure(figsize=(7, 5))  
plt.plot(c\_range, models['train accuracy'], "c", label='train accuracy')  
plt.plot(c\_range, models['test accuracy'], "r", label='test accuracy')  
plt.plot(c\_range, models['test recall'], "b", label='test recall')  
plt.plot(c\_range, models['test precision'], "g", label='test precision')  
plt.ylabel('metrics')  
plt.xlabel('C')  
plt.title('Зависимость метрики accuracy для модели опорных векторов')  
plt.legend()  
plt.grid()  
plt.show()  
  
best\_c = models['C'][models['test precision'].idxmax()]  
print("Best c: ", best\_c, '\n')  
svc = SVC(C=best\_c, probability=True).fit(X\_train, Y\_train)  
  
print("Confusion matrix for SVC:")  
confusion\_matrix = confusion\_matrix\_metric(Y\_test, svc.predict(X\_test))  
print(pd.DataFrame(data=confusion\_matrix, columns=['predicted ham', 'predicted spam'],  
 index=['actual ham', 'actual spam']))  
  
y\_pred\_pr = svc.predict\_proba(X\_test)[:, 1]  
fpr, tpr, threshold = metrics.roc\_curve(Y\_test, y\_pred\_pr)  
roc\_auc = metrics.auc(fpr, tpr)  
plt.title('Receiver Operating Characteristic')  
plt.plot(fpr, tpr, 'b', label='AUC = %0.2f' % roc\_auc)  
plt.legend(loc='lower right')  
plt.plot([0, 1], [0, 1], 'r--')  
plt.xlim([0, 1])  
plt.ylim([0, 1])  
plt.ylabel('True Positive Rate')  
plt.xlabel('False Positive Rate')  
plt.title('ROC-кривая для модели опорных векторов')  
plt.grid()  
plt.show()

Результат:

