Лабораторная работа №5 “Метод опорных векторов”

1. Загрузите данные ex5data1.mat из файла.

In [5]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.io import loadmat
```

1. Загрузите данные ex5data1.mat из файла.

In [6]:

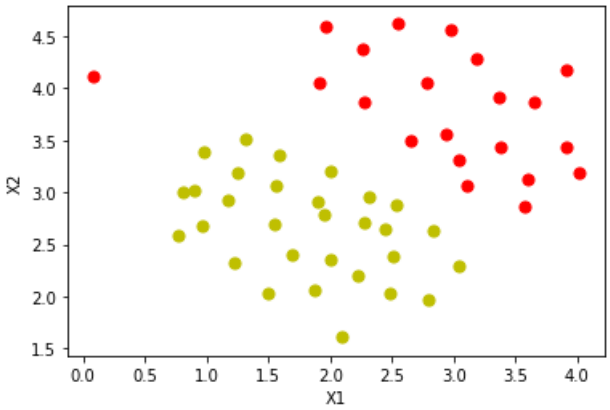
```
mat = loadmat("data/ex5data1.mat")
X = mat["X"]
y = mat["y"]
y = y.reshape(y.shape[0])
```

2. Постройте график для загруженного набора данных: по осям - переменные X1, X2, а точки, принадлежащие различным классам должны быть обозначены различными маркерами.

In [7]:

```
m,n = X.shape[0],X.shape[1]
pos, neg = (y==1).reshape(m, 1), (y==0).reshape(m, 1)

fig, ax = plt.subplots()
ax.scatter(X[pos[:,0],0],X[pos[:,0],1],c="r", s=50)
ax.scatter(X[neg[:,0],0],X[neg[:,0],1],c="y", s=50)
ax.set_xlabel('X1')
ax.set_ylabel('X2')
plt.show()
```



3. Обучите классификатор с помощью библиотечной реализации SVM с линейным ядром на данном наборе.

In [8]:

```
from sklearn.svm import SVC
classifier = SVC(kernel="linear", C=1)
classifier.fit(X,np.ravel(y))
```

Out[8]:

```
SVC(C=1, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
    kernel='linear', max_iter=-1, probability=False, random_state=None,
    shrinking=True, tol=0.001, verbose=False)
```

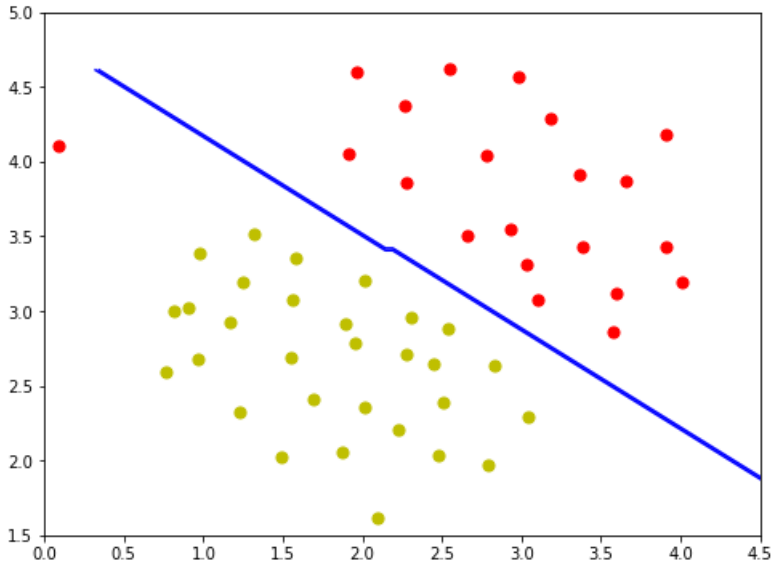
4. Постройте разделяющую прямую для классификаторов с различными параметрами C = 1, C = 100 (совместно с графиком из пункта 2). Объясните различия в полученных прямых?

In [219]:

```
def plot_decision_line(classifier):
    plt.figure(figsize=(8,6))
    plt.scatter(X[pos[:,0],0], X[pos[:,0],1], c="r", s=50)
    plt.scatter(X[neg[:,0],0], X[neg[:,0],1], c="y", s=50)

    # plotting the decision boundary
    X_1,X_2 = np.meshgrid(np.linspace(X[:,0].min(),X[:,1].max(),num=100),np.linspace(X[:,1].min(),X[
:,1].max(),num=100))
    plt.contour(X_1,X_2, classifier.predict(np.array([X_1.ravel(),X_2.ravel()]).T).reshape(X_1.shap
e),1,colors="b")
    plt.xlim(0,4.5)
    plt.ylim(1.5,5)

plot_decision_line(classifier)
```



In [220]:

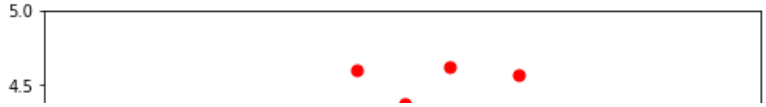
```
classifier = SVC(kernel="linear", C=100)
classifier.fit(X,np.ravel(y))
```

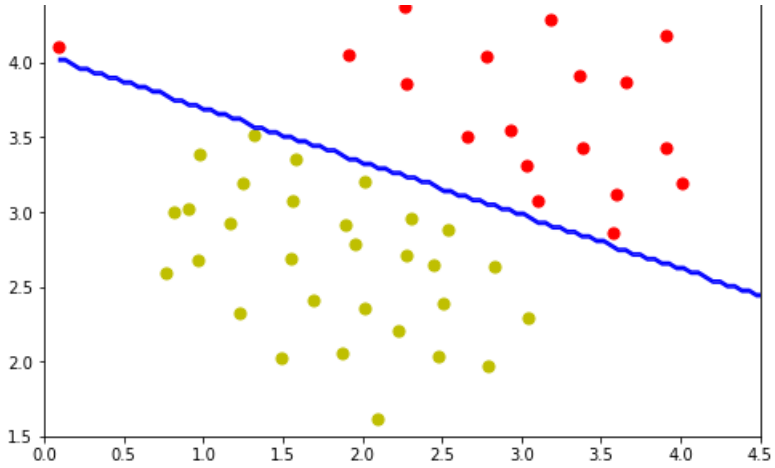
Out[220]:

```
SVC(C=100, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
    kernel='linear', max_iter=-1, probability=False, random_state=None,
    shrinking=True, tol=0.001, verbose=False)
```

In [221]:

```
plot_decision_line(classifier)
```





При C = 100 мы видим, что модель реагирует на один пример, который отклонился от всех остальных. Параметр C это штраф за ошибку и в sklearn значение по умолчанию для C равно 1. Таким образом, при C = 100 наша модель скорее всего переобучается.

5. Реализуйте функцию вычисления Гауссового ядра для алгоритма SVM.

In [222]:

```
def gaussian(x, l, sigma):
    degree = ((x - l)**2).sum(axis=1)
    return np.e ** (-degree) / (2 * sigma**2)
```

6. Загрузите данные ex5data2.mat из файла.

In [239]:

```
mat2 = loadmat("data/ex5data2.mat")
X = mat2["X"]
y = mat2["y"]
y = y.reshape(y.shape[0])
```

7. Обработайте данные с помощью функции Гауссового ядра.

In [240]:

```
X_gaussian = np.array([gaussian(X, l, 1) for l in X])
```

8. Обучите классификатор SVM.

In [241]:

```
clf_gaussian = SVC(kernel='rbf', C=1, gamma=30)
clf_gaussian.fit(X, y)
```

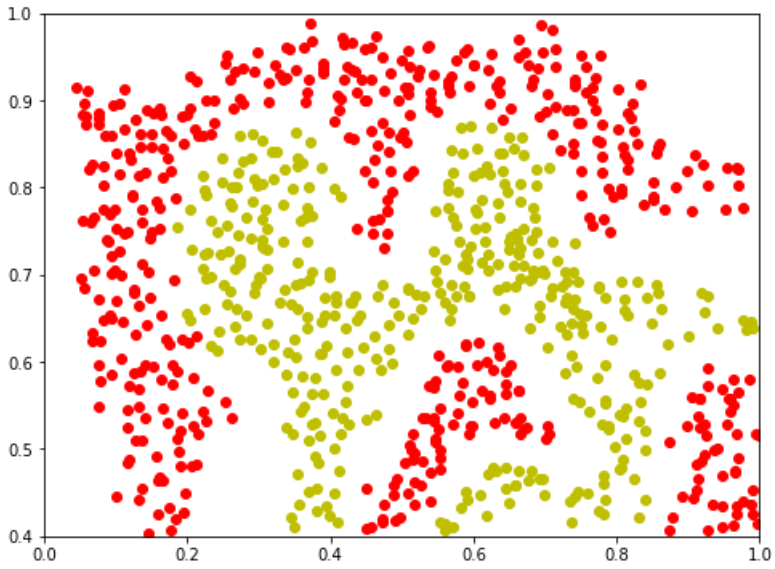
Out[241]:

```
SVC(C=1, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma=30, kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

9. Визуализируйте данные вместе с разделяющей кривой (аналогично пункту 4).

In [242]:

```
m, n = X.shape[0], X.shape[1]
pos, neg = (y==1).reshape(m,1), (y==0).reshape(m,1)
plt.figure(figsize=(8,6))
plt.scatter(X[pos[:,0],0], X[pos[:,0],1], c="r")
plt.scatter(X[neg[:,0],0], X[neg[:,0],1], c="y")
plt.xlim(0,1)
plt.ylim(0.4,1)
plt.show()
```



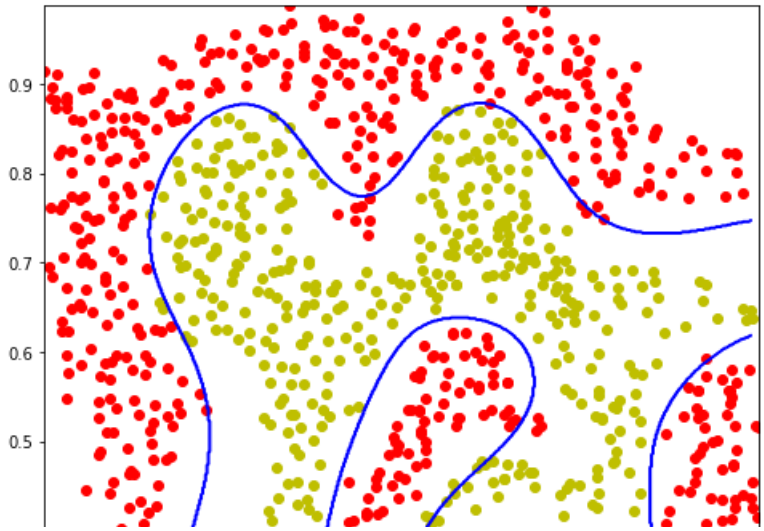
In [243]:

```
def plot_decision_line(classifier, X, y):
    m, n = X.shape[0], X.shape[1]
    pos, neg = (y==1).reshape(m, 1), (y==0).reshape(m, 1)

    plt.figure(figsize=(8,6))
    plt.scatter(X[pos[:,0],0], X[pos[:,0],1], c="r")
    plt.scatter(X[neg[:,0],0], X[neg[:,0],1], c="y")


    # plotting the decision boundary
    X_5, X_6 = np.meshgrid(np.linspace(X[:,0].min(), X[:,1].max(), num=500),
                           np.linspace(X[:,1].min(), X[:,1].max(), num=500))
    plt.contour(X_5, X_6, classifier.predict(np.array([X_5.ravel(), X_6.ravel()]).T).reshape(X_5.sh
ape), 1, colors="b")
    plt.xlim(X[:, 0].min(), X[:, 0].max())
    plt.ylim(X[:, 1].min(), X[:, 1].max())

plot_decision_line(clf_gaussian, X, y)
```



10. Загрузите данные ex5data3.mat из файла.

In [5]:

```
mat3 = loadmat("data/ex5data3.mat")
X = mat3["X"]
y = mat3["y"]
y = y.reshape(y.shape[0])

Xval = mat3["Xval"]
yval = mat3["yval"]
yval = yval.reshape(yval.shape[0])
```

11. Вычислите параметры классификатора SVM на обучающей выборке, а также подберите параметры C и σ2 на валидационной выборке.

In [79]:

```
def calculate_best_params(X, y, Xval, yval, C_list, gamma_list):
    best_score = -np.inf
    best_params = None
    for C in C_list:
        for gamma in gamma_list:
            s = SVC(kernel='rbf', C=C, gamma=gamma)
            s.fit(X, y)
            score = s.score(Xval, yval)
            if score > best_score:
                best_score = score
                best_params = (C, gamma)
    return best_params
```

In [7]:

```
C, gamma = calculate_best_params(X, y, Xval, yval,
                                 C_list=np.logspace(-1, 3, 100), gamma_list=np.linspace(0.0001, 10,
100))
```

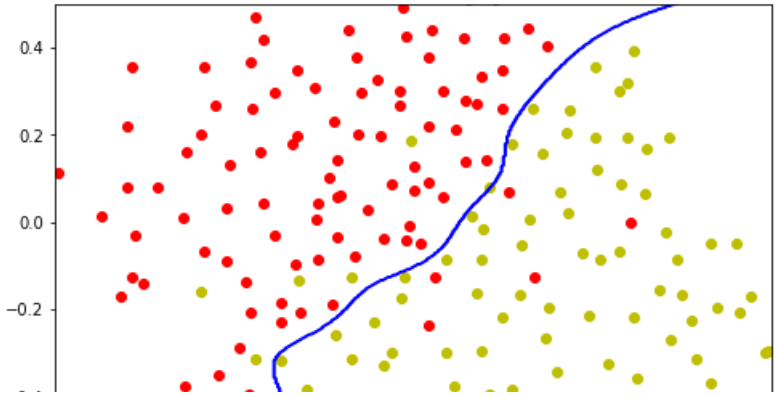
In [ ]:

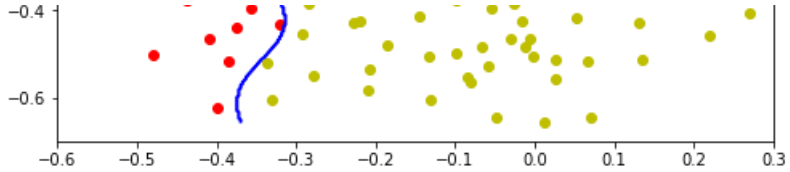
```
classifier = SVC(C = C, gamma = gamma)
classifier.fit(X, y.ravel())
```

12. Визуализируйте данные вместе с разделяющей кривой (аналогично пункту 4).

In [234]:

```
plot_decision_line(classifier, X, y)
```





13. Загрузите данные spamTrain.mat из файла.

In [24]:

```
spam_mat = loadmat("data/spamTrain.mat")
X_train =spam_mat["X"]
y_train = spam_mat["y"]
y_train = y_train.reshape(y_train.shape[0])
```

14. Обучите классификатор SVM.

In [25]:

```
svm_spam_train = SVC(kernel='rbf')
svm_spam_train.fit(X_train, y_train)
```

```
/Users/anton/Documents/Maga/ml_venv/lib/python3.7/site-packages/sklearn/svm/base.py:193:
FutureWarning: The default value of gamma will change from 'auto' to 'scale' in version 0.22 to
account better for unscaled features. Set gamma explicitly to 'auto' or 'scale' to avoid this warn
ing.
  "avoid this warning.", FutureWarning)
```

Out[25]:

```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
    kernel='rbf', max_iter=-1, probability=False, random_state=None,
    shrinking=True, tol=0.001, verbose=False)
```

15. Загрузите данные spamTest.mat из файла.

In [80]:

```
spam_mat = loadmat("data/spamTest.mat")
X_test =spam_mat["Xtest"]
y_test = spam_mat["ytest"]
y_test = y_test.reshape(y_test.shape[0])
```

16. Подберите параметры C и σ2.

In [81]:

```
vals = [150, 200, 300]
C, gamma = calculate_best_params(
    X_train, y_train, X_test, y_test,
    C_list=np.logspace(2, 3, 10), gamma_list=np.linspace(0.0001, 0.0003, 10)
)
```

In [82]:

```
C, gamma
```

Out[82]:

```
(100.0, 0.0002111111111111111)
```

17. Реализуйте функцию предобработки текста письма.

In [83]:

```
import html
import re
import nltk
from nltk.stem.snowball import SnowballStemmer

from nltk.corpus import stopwords
nltk.download("stopwords")

def preprocess(body):
    body = body.lower()

    text = html.unescape(body)
    body = re.sub(r'<[^>]+?>', '', text)

    regx = re.compile(r"(http|https)://[^\s]*")
    body = regx.sub(repl=" httpaddr ", string=body)

    regx = re.compile(r"\b[^\s]+@[^\s]+[.][^\s]+\b")
    body = regx.sub(repl=" emailaddr ", string=body)

    regx = re.compile(r"\b[\d.]+\b")
    body = regx.sub(repl=" number ", string=body)

    regx = re.compile(r"[$]")
    body = regx.sub(repl=" dollar ", string=body)

    regx = re.compile(r"([^\w\s]+)|([_-]+)")
    body = regx.sub(repl=" ", string=body)
    regx = re.compile(r"\s+")
    body = regx.sub(repl=" ", string=body)

    body = body.strip(" ")
    bodywords = body.split(" ")
    keepwords = [word for word in bodywords if word not in stopwords.words('english')]
    stemmer = SnowballStemmer("english")
    stemwords = [stemmer.stem(wd) for wd in keepwords]
    body = " ".join(stemwords)

    return body
```

```
[nltk_data] Downloading package stopwords to /Users/anton/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

18. Загрузите коды слов из словаря vocab.txt.

In [84]:

```
def load_vocabulary(filename):
    vocab = {}

    with open(filename, 'r') as f:
        for line in f.readlines():
            l = line.replace('\n', '').split('\t')
            vocab[l[1]] = int(l[0])

    return vocab
```

In [85]:

```
vocab = load_vocabulary('data/vocab.txt')
```

In [94]:

```
vocab
```

Out[94]:

```
vocab[:5]
```

```
{'aa': 1,
 'ab': 2,
 'abil': 3,
 'abl': 4,
 'about': 5,
 'abov': 6,
 'absolut': 7,
 'abus': 8,
 'ac': 9,
 'accept': 10,
 'access': 11,
 'accord': 12,
 'account': 13,
 'achiev': 14,
 'acquir': 15,
 'across': 16,
 'act': 17,
 'action': 18,
 'activ': 19,
 'actual': 20,
 'ad': 21,
 'adam': 22,
 'add': 23,
 'addit': 24,
 'address': 25,
 'administr': 26,
 'adult': 27,
 'advanc': 28,
 'advantag': 29,
 'advertis': 30,
 'advic': 31,
 'advis': 32,
 'ae': 33,
 'af': 34,
 'affect': 35,
 'affili': 36,
 'afford': 37,
 'africa': 38,
 'after': 39,
 'ag': 40,
 'again': 41,
 'against': 42,
 'agenc': 43,
 'agent': 44,
 'ago': 45,
 'agre': 46,
 'agreement': 47,
 'aid': 48,
 'air': 49,
 'al': 50,
 'alb': 51,
 'align': 52,
 'all': 53,
 'allow': 54,
 'almost': 55,
 'alon': 56,
 'along': 57,
 'alreadi': 58,
 'alsa': 59,
 'also': 60,
 'altern': 61,
 'although': 62,
 'alwai': 63,
 'am': 64,
 'amaz': 65,
 'america': 66,
 'american': 67,
 'among': 68,
 'amount': 69,
 'amp': 70,
 'an': 71,
 'analysi': 72,
 'analyst': 73,
 'and': 74,
 'ani': 75,
 'anim': 76,
```

```
'announc': 77,
'annual': 78,
'annuiti': 79,
'anoth': 80,
'answer': 81,
'anti': 82,
'anumb': 83,
'anybodi': 84,
'anymor': 85,
'anyon': 86,
'anyth': 87,
'anywai': 88,
'anywher': 89,
'aol': 90,
'ap': 91,
'apolog': 92,
'app': 93,
'appar': 94,
'appear': 95,
'appl': 96,
'appli': 97,
'applic': 98,
'appreci': 99,
'approach': 100,
'approv': 101,
'apt': 102,
'ar': 103,
'archiv': 104,
'area': 105,
'aren': 106,
'argument': 107,
'arial': 108,
'arm': 109,
'around': 110,
'arrai': 111,
'arriv': 112,
'art': 113,
'articl': 114,
'artist': 115,
'as': 116,
'ascii': 117,
'ask': 118,
'asset': 119,
'assist': 120,
'associ': 121,
'assum': 122,
'assur': 123,
'at': 124,
'atol': 125,
'attach': 126,
'attack': 127,
'attempt': 128,
'attent': 129,
'attornei': 130,
'attract': 131,
'audio': 132,
'aug': 133,
'august': 134,
'author': 135,
'auto': 136,
'autom': 137,
'automat': 138,
'avail': 139,
'averag': 140,
'avoid': 141,
'awai': 142,
'awar': 143,
'award': 144,
'ba': 145,
'babi': 146,
'back': 147,
'background': 148,
'backup': 149,
'bad': 150,
'balanc': 151,
'ban': 152,
'bank': 153,
```

```
'bar': 154,
'base': 155,
'basenumb': 156,
'basi': 157,
'basic': 158,
'bb': 159,
'bc': 160,
'bd': 161,
'be': 162,
'beat': 163,
'beberg': 164,
'becaus': 165,
'becom': 166,
'been': 167,
'befor': 168,
'begin': 169,
'behalf': 170,
'behavior': 171,
'behind': 172,
'believ': 173,
'below': 174,
'benefit': 175,
'best': 176,
'beta': 177,
'better': 178,
'between': 179,
'bf': 180,
'big': 181,
'bill': 182,
'billion': 183,
'bin': 184,
'binari': 185,
'bit': 186,
'black': 187,
'blank': 188,
'block': 189,
'blog': 190,
'blood': 191,
'blue': 192,
'bnumber': 193,
'board': 194,
'bodi': 195,
'boi': 196,
'bonu': 197,
'book': 198,
'boot': 199,
'border': 200,
'boss': 201,
'boston': 202,
'botan': 203,
'both': 204,
'bottl': 205,
'bottom': 206,
'boundari': 207,
'box': 208,
'brain': 209,
'brand': 210,
'break': 211,
'brian': 212,
'bring': 213,
'broadcast': 214,
'broker': 215,
'browser': 216,
'bug': 217,
'bui': 218,
'build': 219,
'built': 220,
'bulk': 221,
'burn': 222,
'bush': 223,
'busi': 224,
'but': 225,
'button': 226,
'by': 227,
'byte': 228,
'ca': 229,
'cabl': 230,
```

```
'cach': 231,
'calcul': 232,
'california': 233,
'call': 234,
'came': 235,
'camera': 236,
'campaign': 237,
'can': 238,
'canada': 239,
'cannot': 240,
'canon': 241,
'capabl': 242,
'capillari': 243,
'capit': 244,
'car': 245,
'card': 246,
'care': 247,
'career': 248,
'carri': 249,
'cartridg': 250,
'case': 251,
'cash': 252,
'cat': 253,
'catch': 254,
'categori': 255,
'caus': 256,
'cb': 257,
'cc': 258,
'cd': 259,
'ce': 260,
'cell': 261,
'cent': 262,
'center': 263,
'central': 264,
'centuri': 265,
'ceo': 266,
'certain': 267,
'certainli': 268,
'cf': 269,
'challeng': 270,
'chanc': 271,
'chang': 272,
'channel': 273,
'char': 274,
'charact': 275,
'charg': 276,
'charset': 277,
'chat': 278,
'cheap': 279,
'check': 280,
'cheer': 281,
'chief': 282,
'children': 283,
'china': 284,
'chip': 285,
'choic': 286,
'choos': 287,
'chri': 288,
'citi': 289,
'citizen': 290,
'civil': 291,
'claim': 292,
'class': 293,
'classifi': 294,
'clean': 295,
'clear': 296,
'clearli': 297,
'click': 298,
'client': 299,
'close': 300,
'clue': 301,
'cnet': 302,
'cnumber': 303,
'co': 304,
'code': 305,
'collect': 306,
'colleg': 307,
```

```
'color': 308,
'com': 309,
'combin': 310,
'come': 311,
'comfort': 312,
'command': 313,
'comment': 314,
'commentari': 315,
'commerci': 316,
'commiss': 317,
'commit': 318,
'common': 319,
'commun': 320,
'compani': 321,
'compar': 322,
'comparison': 323,
'compat': 324,
'compet': 325,
'competit': 326,
'compil': 327,
'complet': 328,
'comprehens': 329,
'comput': 330,
'concentr': 331,
'concept': 332,
'concern': 333,
'condit': 334,
'conf': 335,
'confer': 336,
'confid': 337,
'confidenti': 338,
'config': 339,
'configur': 340,
'confirm': 341,
'conflict': 342,
'confus': 343,
'congress': 344,
'connect': 345,
'consid': 346,
'consolid': 347,
'constitut': 348,
'construct': 349,
'consult': 350,
'consum': 351,
'contact': 352,
'contain': 353,
'content': 354,
'continu': 355,
'contract': 356,
'contribut': 357,
'control': 358,
'conveni': 359,
'convers': 360,
'convert': 361,
'cool': 362,
'cooper': 363,
'copi': 364,
'copyright': 365,
'core': 366,
'corpor': 367,
'correct': 368,
'correspond': 369,
'cost': 370,
'could': 371,
'couldn': 372,
'count': 373,
'countri': 374,
'coupl': 375,
'cours': 376,
'court': 377,
'cover': 378,
'coverag': 379,
'crash': 380,
'creat': 381,
'creativ': 382,
'credit': 383,
'critic': 384,
```

```
'cross': 385,
'cultur': 386,
'current': 387,
'custom': 388,
'cut': 389,
'cv': 390,
'da': 391,
'dagga': 392,
'dai': 393,
'daili': 394,
'dan': 395,
'danger': 396,
'dark': 397,
'data': 398,
'databas': 399,
'datapow': 400,
'date': 401,
'dave': 402,
'david': 403,
'dc': 404,
'de': 405,
'dead': 406,
'deal': 407,
'dear': 408,
'death': 409,
'debt': 410,
'decad': 411,
'decid': 412,
'decis': 413,
'declar': 414,
'declin': 415,
'decor': 416,
'default': 417,
'defend': 418,
'defens': 419,
'defin': 420,
'definit': 421,
'degre': 422,
'delai': 423,
'delet': 424,
'deliv': 425,
'deliveri': 426,
'dell': 427,
'demand': 428,
'democrat': 429,
'depart': 430,
'depend': 431,
'deposit': 432,
'describ': 433,
'descript': 434,
'deserv': 435,
'design': 436,
'desir': 437,
'desktop': 438,
'despit': 439,
'detail': 440,
'detect': 441,
'determin': 442,
'dev': 443,
'devel': 444,
'develop': 445,
'devic': 446,
'di': 447,
'dial': 448,
'did': 449,
'didn': 450,
'diet': 451,
'differ': 452,
'difficult': 453,
'digit': 454,
'direct': 455,
'directli': 456,
'director': 457,
'directori': 458,
'disabl': 459,
'discount': 460,
'discov': 461,
```

```
'discoveri': 462,
'discuss': 463,
'disk': 464,
'displai': 465,
'disposit': 466,
'distanc': 467,
'distribut': 468,
'dn': 469,
'dnumber': 470,
'do': 471,
'doc': 472,
'document': 473,
'doe': 474,
'doer': 475,
'doesn': 476,
'dollar': 477,
'dollarac': 478,
'dollarnumb': 479,
'domain': 480,
'don': 481,
'done': 482,
'dont': 483,
'doubl': 484,
'doubt': 485,
'down': 486,
'download': 487,
'dr': 488,
'draw': 489,
'dream': 490,
'drive': 491,
'driver': 492,
'drop': 493,
'drug': 494,
'due': 495,
'dure': 496,
'dvd': 497,
'dw': 498,
'dynam': 499,
'ea': 500,
'each': 501,
'earli': 502,
'earlier': 503,
'earn': 504,
'earth': 505,
'easi': 506,
'easier': 507,
'easili': 508,
'eat': 509,
'eb': 510,
'ebai': 511,
'ec': 512,
'echo': 513,
'econom': 514,
'economi': 515,
'ed': 516,
'edg': 517,
'edit': 518,
'editor': 519,
'educ': 520,
'eff': 521,
'effect': 522,
'effici': 523,
'effort': 524,
'either': 525,
'el': 526,
'electron': 527,
'elimin': 528,
'els': 529,
'email': 530,
'emailaddr': 531,
'emerg': 532,
'empir': 533,
'employ': 534,
'employe': 535,
'en': 536,
'enabl': 537,
'encod': 538,
```

```
'encourag': 539,
'end': 540,
'enemi': 541,
'enenkio': 542,
'energi': 543,
'engin': 544,
'english': 545,
'enhanc': 546,
'enjoi': 547,
'enough': 548,
'ensur': 549,
'enter': 550,
'enterpris': 551,
'entertain': 552,
'entir': 553,
'entri': 554,
'enumb': 555,
'environ': 556,
'equal': 557,
'equip': 558,
'equival': 559,
'error': 560,
'especi': 561,
'essenti': 562,
'establish': 563,
'estat': 564,
'estim': 565,
'et': 566,
'etc': 567,
'euro': 568,
'europ': 569,
'european': 570,
'even': 571,
'event': 572,
'eventu': 573,
'ever': 574,
'everi': 575,
'everyon': 576,
'everyth': 577,
'evid': 578,
'evil': 579,
'exactli': 580,
'exampl': 581,
'excel': 582,
'except': 583,
'exchang': 584,
'excit': 585,
'exclus': 586,
'execut': 587,
'exercis': 588,
'exist': 589,
'exmh': 590,
'expand': 591,
'expect': 592,
'expens': 593,
'experi': 594,
'expert': 595,
'expir': 596,
'explain': 597,
'explor': 598,
'express': 599,
'extend': 600,
'extens': 601,
'extra': 602,
'extract': 603,
'extrem': 604,
'ey': 605,
'fa': 606,
'face': 607,
'fact': 608,
'factor': 609,
'fail': 610,
'fair': 611,
'fall': 612,
'fals': 613,
'famili': 614,
'fag': 615
```

```
'faq': 615,
'far': 616,
'fast': 617,
'faster': 618,
'fastest': 619,
'fat': 620,
'father': 621,
'favorit': 622,
'fax': 623,
'fb': 624,
'fd': 625,
'featur': 626,
'feder': 627,
'fee': 628,
'feed': 629,
'feedback': 630,
'feel': 631,
'femal': 632,
'few': 633,
'ffffff': 634,
'ffnumber': 635,
'field': 636,
'fight': 637,
'figur': 638,
'file': 639,
'fill': 640,
'film': 641,
'filter': 642,
'final': 643,
'financ': 644,
'financi': 645,
'find': 646,
'fine': 647,
'finish': 648,
'fire': 649,
'firewal': 650,
'firm': 651,
'first': 652,
'fit': 653,
'five': 654,
'fix': 655,
'flag': 656,
'flash': 657,
'flow': 658,
'fnumber': 659,
'focu': 660,
'folder': 661,
'folk': 662,
'follow': 663,
'font': 664,
'food': 665,
'for': 666,
'forc': 667,
'foreign': 668,
'forev': 669,
'forget': 670,
'fork': 671,
'form': 672,
'format': 673,
'former': 674,
'fortun': 675,
'forward': 676,
'found': 677,
'foundat': 678,
'four': 679,
'franc': 680,
'free': 681,
'freedom': 682,
'french': 683,
'freshrpm': 684,
'fri': 685,
'fridai': 686,
'friend': 687,
'from': 688,
'front': 689,
'ftoc': 690,
'ftp': 691,
'full': 692,
```

```
 'full': 692,
'fulli': 693,
'fun': 694,
'function': 695,
'fund': 696,
'further': 697,
'futur': 698,
'ga': 699,
'gain': 700,
'game': 701,
'gari': 702,
'garrigu': 703,
'gave': 704,
'gcc': 705,
'geek': 706,
'gener': 707,
'get': 708,
'gif': 709,
'gift': 710,
'girl': 711,
'give': 712,
'given': 713,
'global': 714,
'gnome': 715,
'gnu': 716,
'gnupg': 717,
'go': 718,
'goal': 719,
'god': 720,
'goe': 721,
'gold': 722,
'gone': 723,
'good': 724,
'googl': 725,
'got': 726,
'govern': 727,
'gpl': 728,
'grand': 729,
'grant': 730,
'graphic': 731,
'great': 732,
'greater': 733,
'ground': 734,
'group': 735,
'grow': 736,
'growth': 737,
'gt': 738,
'guarante': 739,
'guess': 740,
'gui': 741,
'guid': 742,
'ha': 743,
'hack': 744,
'had': 745,
'half': 746,
'ham': 747,
'hand': 748,
'handl': 749,
'happen': 750,
'happi': 751,
'hard': 752,
'hardwar': 753,
'hat': 754,
'hate': 755,
'have': 756,
'haven': 757,
'he': 758,
'head': 759,
'header': 760,
'headlin': 761,
'health': 762,
'hear': 763,
'heard': 764,
'heart': 765,
'heaven': 766,
'hei': 767,
'height': 768,
'held': 769,
```

```
'held': 769,
'hello': 770,
'help': 771,
'helvetica': 772,
'her': 773,
'herba': 774,
'here': 775,
'hermio': 776,
'hettinga': 777,
'hi': 778,
'high': 779,
'higher': 780,
'highli': 781,
'highlight': 782,
'him': 783,
'histori': 784,
'hit': 785,
'hold': 786,
'home': 787,
'honor': 788,
'hope': 789,
'host': 790,
'hot': 791,
'hour': 792,
'hous': 793,
'how': 794,
'howev': 795,
'hp': 796,
'html': 797,
'http': 798,
'httpaddr': 799,
'huge': 800,
'human': 801,
'hundr': 802,
'ibm': 803,
'id': 804,
'idea': 805,
'ident': 806,
'identifi': 807,
'idnumb': 808,
'ie': 809,
'if': 810,
'ignor': 811,
'ii': 812,
'iii': 813,
'iiiiiiihnumberjnumberhnumberjnumberhnumb': 814,
'illeg': 815,
'im': 816,
'imag': 817,
'imagin': 818,
'immedi': 819,
'impact': 820,
'implement': 821,
'import': 822,
'impress': 823,
'improv': 824,
'in': 825,
'inc': 826,
'includ': 827,
'incom': 828,
'increas': 829,
'incred': 830,
'inde': 831,
'independ': 832,
'index': 833,
'india': 834,
'indian': 835,
'indic': 836,
'individu': 837,
'industri': 838,
'info': 839,
'inform': 840,
'initi': 841,
'inlin': 842,
'innov': 843,
'input': 844,
'insert': 845,
'inside': 846,
```

```
'insid': 846,
'instal': 847,
'instanc': 848,
'instant': 849,
'instead': 850,
'institut': 851,
'instruct': 852,
'insur': 853,
'int': 854,
'integr': 855,
'intel': 856,
'intellig': 857,
'intend': 858,
'interact': 859,
'interest': 860,
'interfac': 861,
'intern': 862,
'internet': 863,
'interview': 864,
'into': 865,
'intro': 866,
'introduc': 867,
'inumb': 868,
'invest': 869,
'investig': 870,
'investor': 871,
'invok': 872,
'involv': 873,
'ip': 874,
'ireland': 875,
'irish': 876,
'is': 877,
'island': 878,
'isn': 879,
'iso': 880,
'isp': 881,
'issu': 882,
'it': 883,
'item': 884,
'itself': 885,
'jabber': 886,
'jame': 887,
'java': 888,
'jim': 889,
'jnumberiiiiiiihepihepihf': 890,
'job': 891,
'joe': 892,
'john': 893,
'join': 894,
'journal': 895,
'judg': 896,
'judgment': 897,
'jul': 898,
'juli': 899,
'jump': 900,
'june': 901,
'just': 902,
'justin': 903,
'keep': 904,
'kei': 905,
'kept': 906,
'kernel': 907,
'kevin': 908,
'keyboard': 909,
'kid': 910,
'kill': 911,
'kind': 912,
'king': 913,
'kingdom': 914,
'knew': 915,
'know': 916,
'knowledg': 917,
'known': 918,
'la': 919,
'lack': 920,
'land': 921,
'languag': 922,
```

```
'laptop': 923,
'larg': 924,
'larger': 925,
'largest': 926,
'laser': 927,
'last': 928,
'late': 929,
'later': 930,
'latest': 931,
'launch': 932,
'law': 933,
'lawrenc': 934,
'le': 935,
'lead': 936,
'leader': 937,
'learn': 938,
'least': 939,
'leav': 940,
'left': 941,
'legal': 942,
'lender': 943,
'length': 944,
'less': 945,
'lesson': 946,
'let': 947,
'letter': 948,
'level': 949,
'lib': 950,
'librari': 951,
'licens': 952,
'life': 953,
'lifetim': 954,
'light': 955,
'like': 956,
'limit': 957,
'line': 958,
'link': 959,
'linux': 960,
'list': 961,
'listen': 962,
'littl': 963,
'live': 964,
'll': 965,
'lo': 966,
'load': 967,
'loan': 968,
'local': 969,
'locat': 970,
'lock': 971,
'lockergnom': 972,
'log': 973,
'long': 974,
'longer': 975,
'look': 976,
'lose': 977,
'loss': 978,
'lost': 979,
'lot': 980,
'love': 981,
'low': 982,
'lower': 983,
'lowest': 984,
'lt': 985,
'ma': 986,
'mac': 987,
'machin': 988,
'made': 989,
'magazin': 990,
'mai': 991,
'mail': 992,
'mailer': 993,
'main': 994,
'maintain': 995,
'major': 996,
'make': 997,
'maker': 998,
'male': 999,
```

```
 'man': 1000,
 ...}
```

19. Реализуйте функцию замены слов в тексте письма после предобработки на их соответствующие коды.

In [131]:

```
def get_codes_vector(body, vocab):
    vector = []
    for word in body.split(' '):
        code = vocab.get(word, None)
        if code:
            vector.append(code)

    return vector
```

20. Реализуйте функцию преобразования текста письма в вектор признаков (в таком же формате как в файлах spamTrain.mat и spamTest.mat).

In [135]:

```
def get_features_vector(codes_vector, vocab):
    codes = set(codes_vector)
    vec = np.zeros(len(vocab), dtype=int)

    for word_code in vocab.values():
        vec[word_code - 1] = int(word_code in codes)

    return vec
```

21. Проверьте работу классификатора на письмах из файлов emailSample1.txt, emailSample2.txt, spamSample1.txt и spamSample2.txt.

In [167]:

```
import os

def make_set_texts(texts, vocab):
    texts_features_set = []
    for email in texts:
        clean_text = preprocess(email)
        codes_vector = get_codes_vector(clean_text, vocab)
        features_vector = get_features_vector(codes_vector, vocab)
        texts_features_set.append(features_vector)

    return texts_features_set

def make_set(files, vocab):
    test_emails = []
    for file in files:
        with open(os.path.join('data', file), 'r') as f:
            test_emails.append(f.read())

    text_set = make_set_texts(test_emails, vocab)

    return np.array(text_set)
```

In [168]:

```
test_files = ['emailSample1.txt', 'spamSample1.txt', 'emailSample2.txt', 'spamSample2.txt']

test_set = make_set(test_files, vocab)
```

In [169]:

```
svm_spam = SVC(kernel='rbf', C=C, gamma=gamma)
svm_spam.fit(X_train, y_train)
svm_spam.predict(test_set)
```

Out[169]:

```
array([0, 1, 0, 1], dtype=uint8)
```

Как видим, наш классификатор работает корректно на данных примерах.

22. Также можете проверить его работу на собственных примерах.

In [170]:

```
test_spam_email = """
    You are not authorized to view this page.
    Please contact us at 717-291-4689 if you feel you are receiving this message in error
"""

test_not_spam = """
Hi, my development site is running on IIS with SMTP server.
Mail is working fine when creating new users or getting a password reset.
I have built a webform and upon submitting I'll get this error.
"""

def predict(text):
    clean_text = preprocess(text)
    codes_vector = get_codes_vector(clean_text, vocab)
    features_vector = get_features_vector(codes_vector, vocab)
    features_vector = np.array([features_vector])

    return svm_spam.predict(features_vector)

print(predict(test_spam_email))
print(predict(test_not_spam))
```

```
[1]
[0]
```

На наших примерах мы также получаем корректный результат.

23. Создайте свой набор данных из оригинального корпуса текстов - http://spamassassin.apache.org/old/publiccorpus/.

In [171]:

```
spam_emails_path = os.path.join('data', 'spam')
ham_emails_path = os.path.join('data', 'easy_ham')

spam_files = [os.path.join(spam_emails_path, fname) for fname in os.listdir(spam_emails_path)]
ham_files = [os.path.join(ham_emails_path, fname) for fname in os.listdir(ham_emails_path)]

def read_files_texts(files):
    texts_corpus = []
    for filename in files:
        with open(filename, "r") as f:
            try:
                lines = f.readlines()
                idx = lines.index("\n")
                texts_corpus.append(preprocess(''.join(lines[idx:])))
            except:
                pass

    return texts_corpus

spam_files_texts = read_files_texts(spam_files)
ham_files_texts = read_files_texts(ham_files)
```

24. Постройте собственный словарь.

In [172]:

```
def make_vocab(texts_corpus):
    import collections

    all_words = [word for text in texts_corpus for word in text.split(" ")]
    words_counter = collections.Counter(all_words)
    vocab_list = [key for key in words_counter if words_counter[key] > 100 and len(key) > 1]
    return {word: i for i, word in enumerate(vocab_list)}

vocab_v2 = make_vocab(spam_files_texts + ham_files_texts)
vocab_v2
```

Out[172]:


```
{'interest': 0,
 'rate': 1,
 'point': 2,
 'number': 3,
 'year': 4,
 'help': 5,
 'find': 6,
 'best': 7,
 'need': 8,
 'hundr': 9,
 'home': 10,
 'improv': 11,
 'second': 12,
 'even': 13,
 'less': 14,
 'servic': 15,
 'free': 16,
 'new': 17,
 'without': 18,
 'fill': 19,
 'quick': 20,
 'simpl': 21,
 'form': 22,
 'start': 23,
 'futur': 24,
 'plan': 25,
 'today': 26,
 'visit': 27,
 'httpaddr': 28,
 'unsubscrib': 29,
 'pleas': 30,
 'must': 31,
 'comput': 32,
 'user': 33,
 'special': 34,
 'packag': 35,
 'deal': 36,
 'softwar': 37,
 'profession': 38,
 'includ': 39,
 'yes': 40,
 'featur': 41,
 'low': 42,
 'price': 43,
 'protect': 44,
 'secur': 45,
 'privat': 46,
 'inform': 47,
 'allow': 48,
 'transfer': 49,
 'file': 50,
 'send': 51,
 'mail': 52,
 'data': 53,
 'easili': 54,
 'perform': 55,
```

```
'great': 56,
'dollar': 57,
'valu': 58,
'get': 59,
'copi': 60,
'email': 61,
'filter': 62,
'opt': 63,
'system': 64,
'state': 65,
'law': 66,
'wish': 67,
'well': 68,
'list': 69,
'thousand': 70,
'provid': 71,
'part': 72,
'messag': 73,
'mime': 74,
'format': 75,
'nextpart': 76,
'000': 77,
'content': 78,
'type': 79,
'text': 80,
'plain': 81,
'charset': 82,
'window': 83,
'encod': 84,
'quot': 85,
'printabl': 86,
'tri': 87,
'better': 88,
'guarante': 89,
'agent': 90,
'age': 91,
'call': 92,
'name': 93,
'phone': 94,
'citi': 95,
'financi': 96,
'total': 97,
'want': 98,
'anyon': 99,
'receiv': 100,
'communic': 101,
'sent': 102,
'insur': 103,
'remov': 104,
'repli': 105,
'instead': 106,
'go': 107,
'legal': 108,
'notic': 109,
'html': 110,
'iso': 111,
'import': 112,
'domain': 113,
'final': 114,
'avail': 115,
'general': 116,
'public': 117,
'regist': 118,
'one': 119,
'info': 120,
'origin': 121,
'com': 122,
'net': 123,
'recent': 124,
'right': 125,
'benefit': 126,
'cours': 127,
'current': 128,
'much': 129,
'easi': 130,
'rememb': 131,
'full': 132,
```

```
'access': 133,
'use': 134,
'control': 135,
'manag': 136,
'address': 137,
'promot': 138,
'compani': 139,
'click': 140,
'line': 141,
'give': 142,
'away': 143,
'cd': 144,
'peopl': 145,
'like': 146,
'month': 147,
'let': 148,
'talk': 149,
'product': 150,
'next': 151,
'day': 152,
'think': 153,
'first': 154,
'check': 155,
'onlin': 156,
'see': 157,
'thing': 158,
'littl': 159,
'busi': 160,
'good': 161,
'us': 162,
'work': 163,
'time': 164,
'other': 165,
'week': 166,
'exact': 167,
'tell': 168,
'expect': 169,
'done': 170,
'would': 171,
'look': 172,
'follow': 173,
'area': 174,
'code': 175,
'emailaddr': 176,
'ad': 177,
'bill': 178,
'person': 179,
'may': 180,
'stop': 181,
'cost': 182,
'word': 183,
'subject': 184,
'invest': 185,
'risk': 186,
'america': 187,
'turn': 188,
'tax': 189,
'fail': 190,
'pay': 191,
'later': 192,
'properti': 193,
'kind': 194,
'money': 195,
'buy': 196,
'end': 197,
'know': 198,
'make': 199,
'chang': 200,
'live': 201,
'problem': 202,
'lot': 203,
'requir': 204,
'direct': 205,
'support': 206,
'local': 207,
'school': 208,
'thank': 209,
```

```
'simpli': 210,
'increas': 211,
'target': 212,
'million': 213,
'via': 214,
'act': 215,
'result': 216,
'realli': 217,
'regard': 218,
'way': 219,
'advertis': 220,
'custom': 221,
'web': 222,
'site': 223,
'take': 224,
'search': 225,
'engin': 226,
'contact': 227,
'read': 228,
'offer': 229,
'rather': 230,
'post': 231,
'group': 232,
'order': 233,
'hour': 234,
'addit': 235,
'internet': 236,
'book': 237,
'report': 238,
'big': 239,
'mass': 240,
'oper': 241,
'guid': 242,
'worth': 243,
'limit': 244,
'suppli': 245,
'last': 246,
'fax': 247,
'card': 248,
'place': 249,
'cut': 250,
'past': 251,
'offic': 252,
'effect': 253,
'back': 254,
'accept': 255,
'everyth': 256,
'within': 257,
'clean': 258,
'select': 259,
'option': 260,
'intern': 261,
'add': 262,
'care': 263,
'note': 264,
'complet': 265,
'signatur': 266,
'except': 267,
'street': 268,
'box': 269,
'case': 270,
'question': 271,
'date': 272,
'amount': 273,
'understand': 274,
'author': 275,
'consid': 276,
'respons': 277,
'applic': 278,
'open': 279,
'return': 280,
'howev': 281,
'found': 282,
'bank': 283,
'tire': 284,
'program': 285,
'per': 286,
```

```
'view': 287,
'never': 288,
'anoth': 289,
'got': 290,
'etc': 291,
'life': 292,
'technolog': 293,
'subscrib': 294,
'connect': 295,
'meet': 296,
'share': 297,
'platform': 298,
'interact': 299,
'save': 300,
'fact': 301,
'distribut': 302,
'reason': 303,
'specif': 304,
'immedi': 305,
'grow': 306,
'keep': 307,
'individu': 308,
'incom': 309,
'real': 310,
'opportun': 311,
'experi': 312,
'train': 313,
'link': 314,
'request': 315,
'non': 316,
'face': 317,
'major': 318,
'answer': 319,
'standard': 320,
'client': 321,
'thought': 322,
'might': 323,
'leav': 324,
'irish': 325,
'linux': 326,
'un': 327,
'subscript': 328,
'maintain': 329,
'assist': 330,
'due': 331,
'mr': 332,
'base': 333,
'econom': 334,
'unit': 335,
'account': 336,
'fund': 337,
'also': 338,
'partner': 339,
'countri': 340,
'ask': 341,
'sinc': 342,
'come': 343,
'effort': 344,
'success': 345,
'believ': 346,
'sign': 347,
'copyright': 348,
'feel': 349,
'error': 350,
'decor': 351,
'famili': 352,
'red': 353,
'activ': 354,
'top': 355,
'news': 356,
'noth': 357,
'quit': 358,
'market': 359,
'govern': 360,
'polit': 361,
'discuss': 362,
'altern': 363,
```

```
'friend': 364,
'high': 365,
'old': 366,
'love': 367,
'websit': 368,
'page': 369,
'java': 370,
'center': 371,
'yahoo': 372,
'ac': 373,
'project': 374,
'comment': 375,
'member': 376,
'made': 377,
'two': 378,
'said': 379,
'around': 380,
'cash': 381,
'bit': 382,
'came': 383,
'went': 384,
'parti': 385,
'mind': 386,
'stori': 387,
'ever': 388,
'learn': 389,
'wait': 390,
'alway': 391,
'guy': 392,
'sell': 393,
'put': 394,
'probabl': 395,
'hard': 396,
'could': 397,
'larg': 398,
'long': 399,
'trade': 400,
'term': 401,
'certain': 402,
'whether': 403,
'basic': 404,
'societi': 405,
'strategi': 406,
'sure': 407,
'execut': 408,
'averag': 409,
'absolut': 410,
'posit': 411,
'becom': 412,
'drive': 413,
'practic': 414,
'close': 415,
'figur': 416,
'three': 417,
'lost': 418,
'differ': 419,
'game': 420,
'yet': 421,
'power': 422,
'method': 423,
'almost': 424,
'everi': 425,
'tool': 426,
'develop': 427,
'say': 428,
'forward': 429,
'mean': 430,
'possibl': 431,
'anyth': 432,
'move': 433,
'world': 434,
'actual': 435,
'exampl': 436,
'appear': 437,
'normal': 438,
'key': 439,
'detail': 440,
```

```
'similar': 441,
'spam': 442,
'hope': 443,
'cell': 444,
'bad': 445,
'instal': 446,
'small': 447,
'seen': 448,
'databas': 449,
'present': 450,
'generat': 451,
'mani': 452,
'record': 453,
'alreadi': 454,
'either': 455,
'releas': 456,
'contain': 457,
'directori': 458,
'run': 459,
'show': 460,
'design': 461,
'sort': 462,
'compil': 463,
'process': 464,
'given': 465,
'industri': 466,
'machin': 467,
'begin': 468,
'american': 469,
'clear': 470,
'digit': 471,
'agre': 472,
'organ': 473,
'nation': 474,
'dvd': 475,
'septemb': 476,
'level': 477,
'often': 478,
'hand': 479,
'women': 480,
'sex': 481,
'claim': 482,
'least': 483,
'someon': 484,
'polici': 485,
'play': 486,
'though': 487,
'abl': 488,
'pick': 489,
'log': 490,
'url': 491,
'issu': 492,
'still': 493,
'univers': 494,
'sourc': 495,
'research': 496,
'august': 497,
'forc': 498,
'3d': 499,
'true': 500,
'rule': 501,
'hi': 502,
'enough': 503,
'continu': 504,
'join': 505,
'job': 506,
'exist': 507,
'decid': 508,
'id': 509,
'sever': 510,
'network': 511,
'pretti': 512,
'test': 513,
'threat': 514,
'corpor': 515,
'freedom': 516,
'creat': 517,
```

```
'mayb': 518,
'stuff': 519,
'matter': 520,
'build': 521,
'wonder': 522,
'step': 523,
'els': 524,
'load': 525,
'guess': 526,
'version': 527,
'whole': 528,
'server': 529,
'fix': 530,
'upon': 531,
'relat': 532,
'document': 533,
'tag': 534,
'write': 535,
'man': 536,
'respect': 537,
'prefer': 538,
'caus': 539,
'seem': 540,
'happen': 541,
'hit': 542,
'mark': 543,
'singl': 544,
'night': 545,
'www': 546,
'presid': 547,
'idea': 548,
'someth': 549,
'script': 550,
'welcom': 551,
'imag': 552,
'across': 553,
'worker': 554,
'set': 555,
'updat': 556,
'attack': 557,
'action': 558,
'upgrad': 559,
'wrong': 560,
'global': 561,
'oct': 562,
'bush': 563,
'john': 564,
'entir': 565,
'far': 566,
'sequenc': 567,
'half': 568,
'suggest': 569,
'size': 570,
'nice': 571,
'human': 572,
'grant': 573,
'monday': 574,
'sound': 575,
'ok': 576,
'sf': 577,
'sponsor': 578,
'thinkgeek': 579,
'geek': 580,
'heaven': 581,
'sa': 582,
'depend': 583,
'memori': 584,
'war': 585,
'tim': 586,
'common': 587,
'command': 588,
'terrorist': 589,
'chris': 590,
'pm': 591,
'modul': 592,
'perhap': 593,
'ago': 594,
```

```
 '3e': 595,
 'sep': 596,
 'osdn': 597,
 'spamassassin': 598,
 'particular': 599,
 'wed': 600,
 'fals': 601,
 'tie': 602,
 'cc': 603,
 'root': 604,
 'apt': 605,
 'redhat': 606,
 'i386': 607,
 'os': 608,
 'header': 609,
 'rpm': 610,
 'fri': 611,
 'wrote': 612,
 'ham': 613,
 'msgs': 614,
 'token': 615,
 'spammer': 616,
 'anyway': 617,
 'spec': 618,
 'razor': 619,
 'languag': 620,
 'configur': 621,
 'usr': 622,
 'folder': 623,
 'procmail': 624,
 'score': 625,
 'thu': 626,
 'sun': 627,
 'aug': 628,
 'fork': 629,
 'perl': 630,
 'bug': 631,
 'mon': 632,
 'exmh': 633,
 'driver': 634,
 'matthia': 635,
 'kernel': 636,
 'lib': 637,
 'alsa': 638,
 'dave': 639,
 'cvs': 640,
 'hat': 641,
 'devel': 642,
 'adam': 643,
 'tue': 644,
 'pgp': 645,
 'terror': 646,
 'unseen': 647,
 'invok': 648,
 'echo': 649,
 'alb': 650}
```

25. Как изменилось качество классификации? Почему?

In [177]:

```
y_test_v2 = [0] * len(ham_files_texts) + [1] * len(spam_files_texts)

x_vocab2 = make_set_texts(spam_files_texts + ham_files_texts, vocab_v2)
svm_spam_vocab_2 = SVC(kernel='rbf', C=C, gamma=gamma)
svm_spam_vocab_2.fit(x_vocab2, y_test_v2)

svm_spam_vocab_1 = SVC(kernel='rbf', C=C, gamma=gamma)
x_vocab1 = make_set_texts(spam_files_texts + ham_files_texts, vocab)
svm_spam_vocab_1.fit(x_vocab1, y_test_v2)

print(f'Score of classicator v2: {svm_spam_vocab_2.score(x_vocab2, y_test_v2)}')
print(f'Score of classicator v1: {svm_spam_vocab_1.score(x_vocab1, y_test_v2)}')
```

```
Score of classicator v2: 0.8525594808940159
Score of classicator v1: 0.8713049747656814
```

Как видим, на двух идентичных сетах данных мы получаем выше точность, когда строим тренировочный сет с помощью первого словаря данных. Это достаточно логично, так как первый словарь содержит больше слов => мы можем выделить больше признаков.

In [ ]:

```
```