НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Дисциплина: «Анализ данных»

Домашнее задание на тему:

«Лабораторная работа №5»

Выполнил: Осипов Лев,

студент группы 301ПИ (1).

Москва, 2015 г.

**СОДЕРЖАНИЕ**

**Теоретическая часть3**

**Практическая часть3**

**Список литературы7**

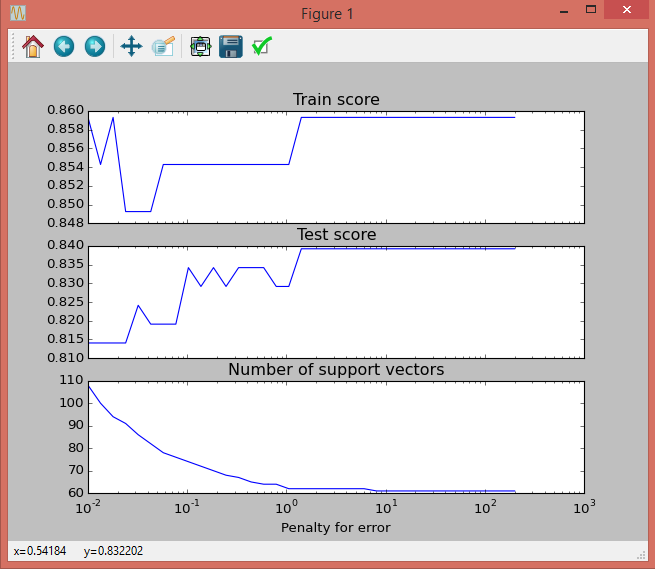
**Текст программы8**

**ТЕОРЕТИЧЕСКАЯ ЧАСТЬ**

Ответ – синяя кривая. Во время кризиса банку важнее кредитовать как можно меньше ненадежных заемщиков. Так как FRP отвечает за вероятность того, что ненадежного определили как надежного (ошибка второго рода), при низком значении FRP нужно высокое значение TPR, которое отвечает за вероятность того, что надежного определили как ненадежного (ошибка первого рода). Именно синий график демонстрирует такую зависимость.

**ПРАКТИЧЕСКАЯ ЧАСТЬ**

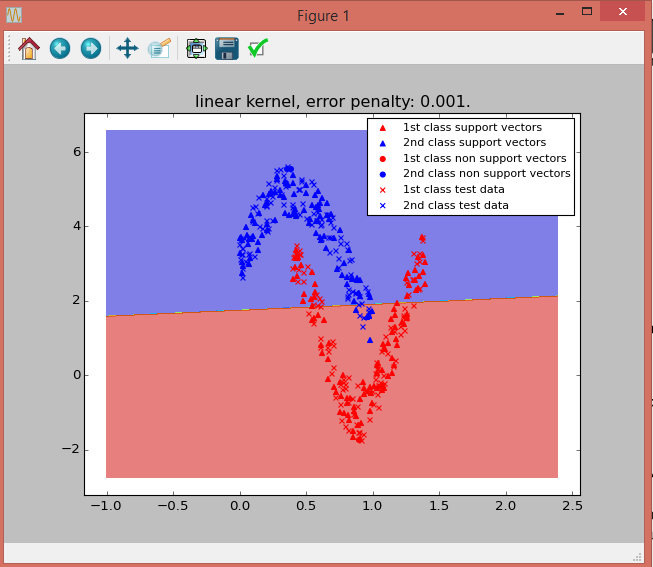
Для решения задания была написана программа, исследующая решающих правил для линейного ядра.

****

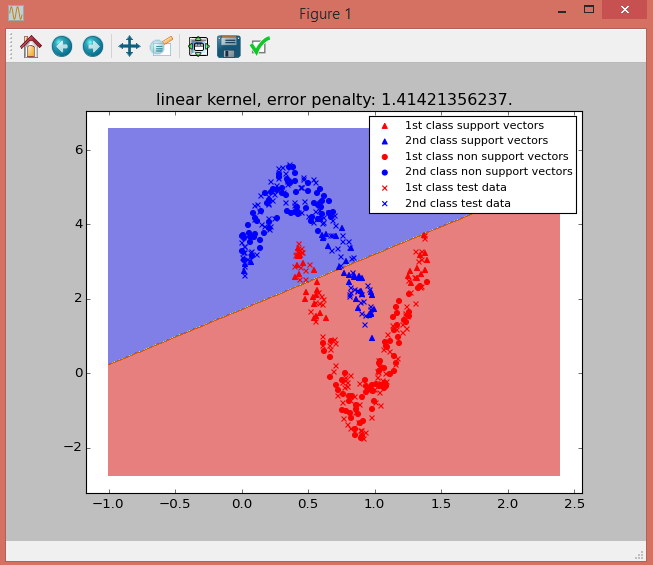
*Рис. 1. Зависимость верно классифицированных объектов на обучающей и тестовой выборках, а также количество опорных векторов от С*

Использовать точность возможно.

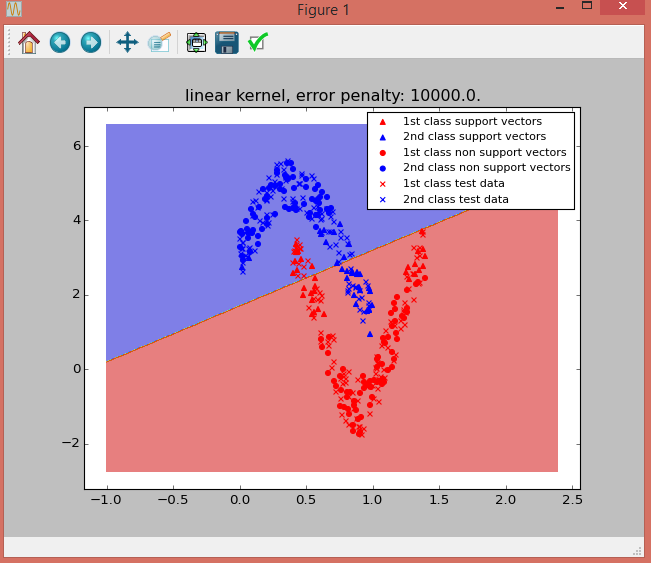
Без доступа к тестовой выборке можно воспользоваться число опорных векторов.



*Рис. 2. Оптимальное С*



*Рис. 1. Неадекватно маленькое С*

****

*Рис. 1. Неадекватно большое С*

**СПИСОК ЛИТЕРАТУРЫ**

1. **Анализ данных (Программная инженерия)** – http://wiki.cs.hse.ru/%D0%90%D0%BD%D0%B0%D0%BB%D0%B8%D0%B7\_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85\_%28%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%B0%D1%8F\_%D0%B8%D0%BD%D0%B6%D0%B5%D0%BD%D0%B5%D1%80%D0%B8%D1%8F%29#.D0.9E.D1.84.D0.BE.D1.80.D0.BC.D0.BB.D0.B5.D0.BD.D0.B8.D0.B5\_.D0.BF.D0.B8.D1.81.D0.B5.D0.BC

**ТЕКСТ ПРОГРАММЫ**

\_\_author\_\_ = 'Lev Osipov'  
  
**import** pandas **as** pd  
**import** numpy **as** np  
**import** matplotlib.pyplot **as** plt  
**from** sklearn.svm **import** SVC  
**from** math **import** log10  
  
  
**def learn\_stat**(train\_data, train\_labels, test\_data, test\_labels, kernel, C):  
 clf = SVC(C=C, kernel=kernel)  
 clf.fit(train\_data, train\_labels)  
 train\_score = clf.score(train\_data, train\_labels)  
 test\_score = clf.score(test\_data, test\_labels)  
 support = clf.support\_.size  
 **return** train\_score, test\_score, support  
  
  
**def plot\_stat**(train\_data, train\_labels, test\_data, test\_labels, kernel, minC, maxC, steps):  
  
 train\_score = np.empty(steps)  
 test\_score = np.empty(steps)  
 support = np.empty(steps)  
  
 c\_interval = np.logspace(minC, maxC, steps)  
 **for** i, C **in** enumerate(c\_interval):  
 train\_score[i], test\_score[i], support[i] = \  
 learn\_stat(train\_data, train\_labels, test\_data, test\_labels, kernel, C)  
 f, ax = plt.subplots(3, sharex=True)  
 i = 0  
  
 ax[i].plot(c\_interval, train\_score)  
 ax[i].set\_title("Train score")  
 ax[i].set\_xscale('log', basex=10)  
 i += 1  
  
 ax[i].plot(c\_interval, test\_score)  
 ax[i].set\_title("Test score")  
 ax[i].set\_xscale('log', basex=10)  
 i += 1  
  
 ax[i].plot(c\_interval, support)  
 ax[i].set\_title("Number of support vectors")  
 ax[i].set\_xlabel("Penalty for error")  
  
 ax[i].set\_xscale('log', basex=10)  
  
 plt.show()  
  
  
**def plot\_support**(train\_data, train\_labels, test\_data, test\_labels, kernel, C):  
  
 clf = SVC(C=C, kernel=kernel)  
 clf.fit(train\_data, train\_labels)  
  
 support\_i = clf.support\_  
  
 train\_sup\_vec = train\_data[support\_i]  
 train\_sup\_lab = train\_labels[support\_i]  
  
 train\_non\_sup\_vec = np.delete(train\_data, support\_i, axis=0)  
 train\_non\_sup\_lab = np.delete(train\_labels, support\_i, axis=0)  
  
 train\_sup\_vec\_1 = train\_sup\_vec[np.where(train\_sup\_lab == 1)]  
 train\_sup\_vec\_2 = train\_sup\_vec[np.where(train\_sup\_lab != 1)]  
  
 train\_non\_sup\_vec\_1 = train\_non\_sup\_vec[np.where(train\_non\_sup\_lab == 1)]  
 train\_non\_sup\_vec\_2 = train\_non\_sup\_vec[np.where(train\_non\_sup\_lab != 1)]  
  
 test\_data\_1 = test\_data[np.where(test\_labels == 1)]  
 test\_data\_2 = test\_data[np.where(test\_labels != 1)]  
  
 all = np.concatenate((test\_data, train\_data), axis=0)  
 h = .01  
  
 x\_min, x\_max = all[:, 0].min() - 1, all[:, 0].max() + 1  
 y\_min, y\_max = all[:, 1].min() - 1, all[:, 1].max() + 1  
 xx, yy = np.meshgrid(np.arange(x\_min, x\_max, h),  
 np.arange(y\_min, y\_max, h))  
  
 z = clf.predict(np.c\_[xx.ravel(), yy.ravel()])  
 z = z.reshape(xx.shape)  
 plt.contourf(xx, yy, z, alpha=0.5)  
  
 plt.scatter(train\_sup\_vec\_1[:, 0], train\_sup\_vec\_1[:, 1], color='r', marker='^',  
 label='1st class support vectors')  
 plt.scatter(train\_sup\_vec\_2[:, 0], train\_sup\_vec\_2[:, 1], color='b', marker='^',  
 label='2nd class support vectors')  
  
 plt.scatter(train\_non\_sup\_vec\_1[:, 0], train\_non\_sup\_vec\_1[:, 1], color='r', marker='o',  
 label='1st class non support vectors')  
 plt.scatter(train\_non\_sup\_vec\_2[:, 0], train\_non\_sup\_vec\_2[:, 1], color='b', marker='o',  
 label='2nd class non support vectors')  
  
 plt.scatter(test\_data\_1[:, 0], test\_data\_1[:, 1], color='r', marker='x', label='1st class test data')  
 plt.scatter(test\_data\_2[:, 0], test\_data\_2[:, 1], color='b', marker='x', label='2nd class test data')  
  
 plt.legend(scatterpoints=1, fontsize=10)  
 plt.title("{0} kernel, error penalty: {1}.".format(kernel, C))  
  
 plt.show()  
 plt.clf()  
  
  
**def find\_optimal\_c**(train\_data, train\_labels, test\_data, test\_labels, kernel, minC, maxC, steps):  
 c\_interval = np.logspace(minC, maxC, steps)  
 max\_score = 0  
 c = minC  
 **for** i, C **in** enumerate(c\_interval):  
 test\_score = learn\_stat(train\_data, train\_labels, test\_data, test\_labels, kernel, C)[1]  
 **if** test\_score > max\_score:  
 max\_score = test\_score  
 c = C  
 **return** c  
  
# Task 1  
tr\_data = pd.read\_csv('synth\_train.csv').as\_matrix()  
tr\_labels = tr\_data[:, 0]  
tr\_data = np.delete(tr\_data, 0, axis=1)  
  
te\_data = pd.read\_csv('synth\_test.csv').as\_matrix()  
te\_labels = te\_data[:, 0]  
te\_data = np.delete(te\_data, 0, axis=1)  
  
# Task 2  
minimC = log10(1e-2)  
maximC = log10(200)  
steps\_count = 35  
plot\_stat(tr\_data, tr\_labels, te\_data, te\_labels, 'linear', minimC, maximC, steps\_count)  
  
# Task 3  
c = find\_optimal\_c(tr\_data, tr\_labels, te\_data, te\_labels, 'linear', minimC, maximC, steps\_count)  
plot\_support(tr\_data, tr\_labels, te\_data, te\_labels, 'linear', 1e-3)  
plot\_support(tr\_data, tr\_labels, te\_data, te\_labels, 'linear', c)  
plot\_support(tr\_data, tr\_labels, te\_data, te\_labels, 'linear', 1e4)