МОСКОВСКИЙ ИНСТИТУТ ЭЛЕКТРОННОЙ ТЕХНИКИ

Институт системной и программной инженерии   
и информационных технологий (Институт СПИНТех)

Лабораторная работа № 2

по курсу «Интеллектуальные системы»

Предсказание вероятности возникновения события по значениям множества признаков (логистическая регрессия)

Выполнил:

Залесский О.А. гр. ПИН-12М

Проверил преподаватель:

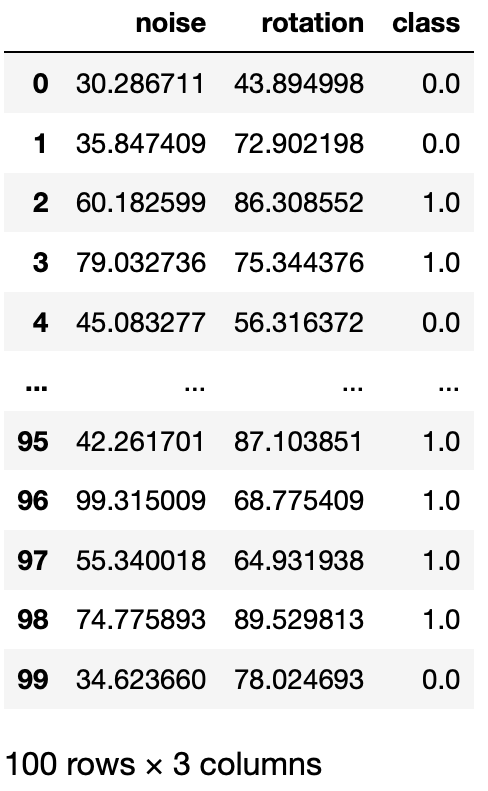
проф., д.ф.-м. н. Рычагов М.Н.

Москва, МИЭТ, 2023

**Инициализация**

df = pd.read\_csv('engine.csv')

df



# Формируем вектор признаков

m = len(df)

x = np.array(df['noise'])

temp = np.array(df['rotation'])

y = np.array(df['class'])

x = x.reshape((m, 1))

temp = temp.reshape((m, 1))

y = y.reshape((m, 1))

X = np.hstack((temp,x))

print(X[:5])

[[43.89499752 30.28671077]

[72.90219803 35.84740877]

[86.3085521 60.18259939]

[75.34437644 79.03273605]

[56.31637178 45.08327748]]

# Добавляем к вектору признаков столбец единиц для theta\_0

t = np.ones((m,1))

X = np.hstack((t,X))

print(X[:5])

[[ 1. 43.89499752 30.28671077]

[ 1. 72.90219803 35.84740877]

[ 1. 86.3085521 60.18259939]

[ 1. 75.34437644 79.03273605]

[ 1. 56.31637178 45.08327748]]

## **Задание 1. Отображение**

def plotData(X, y, size, marker, color, label):

'''

Инструкция: Отобразите на графике исходные обучающие данные, используя

команды "figure", "scatter". Создайте подписи осей графиков,

применяя команды "xlabel" и "ylabel".

'''

# Ваш код здесь

plt.scatter(X, y, marker=marker, c = color, label = label)

plt.xlabel('Шум двигателя')

plt.ylabel('Неравномерность вращения двигателя')

#строим набор наших данных

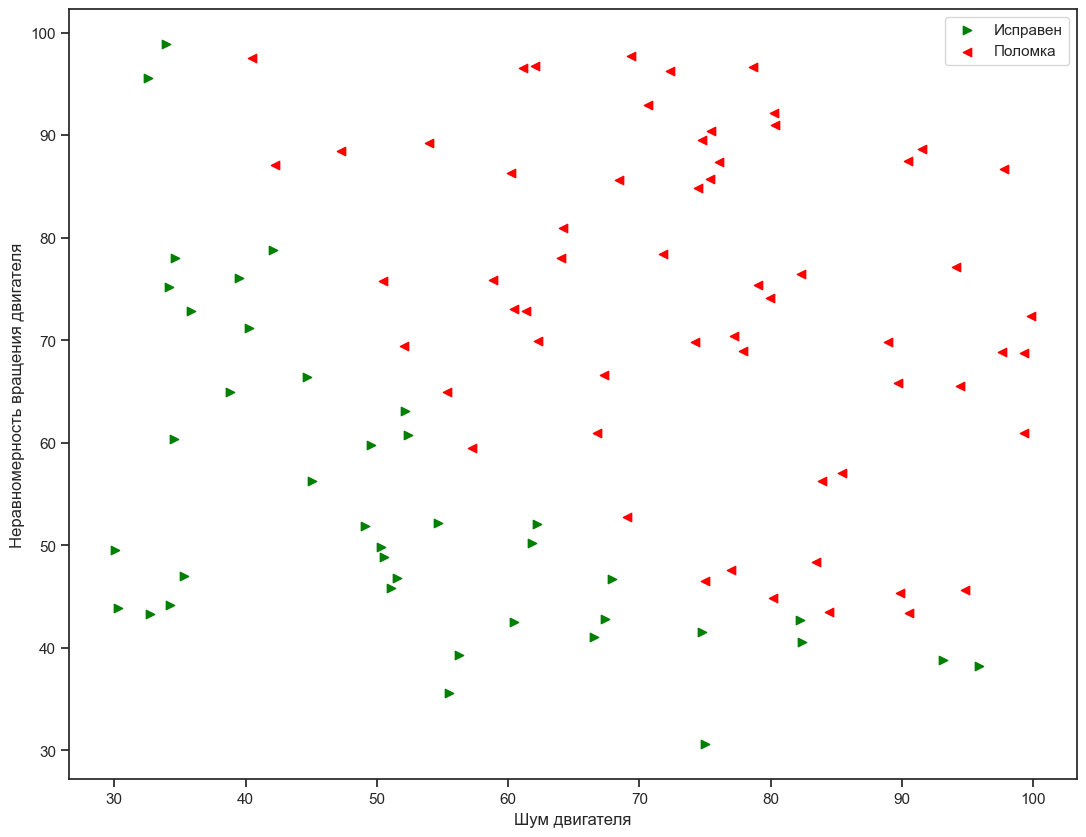
plt.figure(figsize=(13,10))

plotData(df['noise'][df['class'] == 0], df['rotation'][df['class'] == 0], 250, '>', 'green', 'Исправен' )

plotData(df['noise'][df['class'] == 1], df['rotation'][df['class'] == 1], 250, '<', 'red', 'Поломка')

plt.legend()

plt.show()



**Задание 2: Вычисление функции стоимости и градиентов**

initial\_theta = np.zeros((n , 1))

initial\_theta # Инициализируем нулевой вектор theta

array([[0.],

[0.],

[0.]])

def sigmoid(z):

"""

Указание: z может быть матрицей вектором или скаляром.

"""

z = -np.asarray(z)

# Ваш код здесь

g = 1 / (1 + np.exp(z))

return g

def costFunction(theta, X, y, return\_grad=False):

"""

Указание: Градиент должен иметь ту же размерность, что и theta

"""

# Ваш код здесь

m = len(y)

#В процессе выполнения задания, следующие переменные должны быть вычислены правильно

J = 0

grad = np.zeros((theta.shape[0] , 1))

h\_theta = []

first\_part = []

second\_part = []

for i in range (m):

h\_theta.append( sigmoid(np.matmul(np.transpose(theta), X[i])) )

first\_part.append( -(y[i] \* math.log(h\_theta[i])) )

second\_part.append( (1 - y[i]) \* math.log(1 - h\_theta[i]) )

J += first\_part[i] - second\_part[i]

J = J / m

for j in range(X.shape[1]):

summary = 0

for i in range(m):

summary += (h\_theta[i] - y[i]) \* X[i][j]

grad[j] = summary / m

# Примечание: в оригинальном скрипте не было условия вывода переменной grad (выходила ошибка при return\_grad = True), потому я добавил проверку в конце

if return\_grad:

return J, grad

else:

return J

cost, grad = costFunction(initial\_theta, X, y, True)

print('Значение функции стоимости при начальных (нулевых) значениях вектора thetа:', cost, '\n');

print('Значение градиента при начальных (нулевых) значениях вектора thetа:',grad, ' \n');

Значение функции стоимости при начальных

(нулевых) значениях вектора thetа: [0.69314718]

Значение градиента при начальных (нулевых) значениях вектора thetа: [[ -0.1 ]

[-11.26284221]

[-12.00921659]]

**Задание 3: Оптимизация**

Используем для вычисления оптимальных значений параметров theta библиотеку scipy

from scipy.optimize import fmin

myargs = (X, y)

theta = fmin(costFunction, x0=initial\_theta, args=myargs)

Optimization terminated successfully.

Current function value: 0.203498

Iterations: 157

Function evaluations: 287

print('Значение theta:',theta)

Значение theta:

[-25.16130062 0.20147143 0.20623142]

**Отображение границы классов**

def plotDecisionBoundary(X, theta, df):

"""

Указание: Ваша задача разобраться, как строиться граница классов

"""

x\_values = [np.min(X[:, 1]), np.max(X[:, 2])]

y\_values = - (theta[0] + np.dot(theta[1], x\_values)) / theta[2]

plt.figure(figsize=(13,10))

plt.plot(x\_values, y\_values, linewidth = 3, color = 'black', label='Граница классов')

plotData(df['noise'][df['class'] == 0], df['rotation'][df['class'] == 0], 250, '>', 'green', 'Исправен' )

plotData(df['noise'][df['class'] == 1], df['rotation'][df['class'] == 1], 250, '<', 'red', 'Поломка')

plt.legend()

plt.show()

***Пояснение вычисления y\_values***:

Пусть уравнение решающей границы равно

, где k – наклон прямой, а c – ее сдвиг.

Также модель с двумя признаками имеет формулу

Граница раздела двух классов проходит там, где , следовательно

Чтобы найти значение сдвига c (сдвиг линии вдоль оси x2) допустим . Тогда

и согласно равенству при , следует что

Чтобы найти наклон, возьмем точку на прямой решающей границы с координатами

Тогда

Также, так как они расположены на прямой с уравнением следует, что

Тогда

Следовательно

**Задание 4: Предсказание и оценка точности**

Определение вероятности работоспособности или отказа двигателя, состояние которого характеризуется следующими параметрами, шум - 45 ед., вибрация - 85 ед.

prob = sigmoid(np.array([1, 45, 85]).dot(theta[:, np.newaxis]))

print('Для двигателя с уровнем шума 45 и вибрацией 85, предсказывается поломка с вероятностью: {:.2%} \n'.format(prob[0]))

Для двигателя с уровнем шума 45 и вибрацией 85,

предсказывается поломка с вероятностью: 80.76%

def predict(x, theta, y, threshold=0.5):

"""

PREDICT Отнесение образца к классам 0 или 1 ("исправен" или "не исправен")

в процессе линейной регрессии на основании оценки theta

PREDICT обеспечивает классификацию X с пороговым

значением 0.5 (т.е., если значение сигмоидной функции

sigmoid(theta'\*x) >= 0.5, то присвоение 1)

"""

# Я дополнил параметры значением theta ибо без нее невозможно # просчитать равенство с классом, заранее описанным в данных df

# Ваш код здесь

accuracy = 0

res = 0

for i in range(x.shape[0]):

if sigmoid(np.matmul(np.transpose(theta), X[i])) >= threshold:

res = 1

else:

res = 0

if res == y[i]:

accuracy += 1

return accuracy

print('Точность обучения: {:.0%} \n'.format(predict(X, theta[:, np.newaxis], y)/X.shape[0]))

# Поменял строгое значение 100 на значение X.shape[0], так как в задании 6 так не сработает из-за разных размеров массива иследований

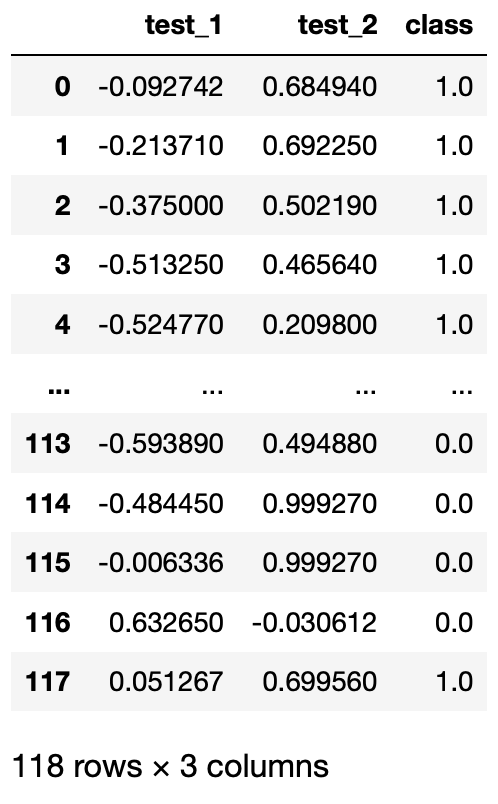
Точность обучения: 89%

## **Задание 5: Регуляризованная логистическая регрессия**

**Инициализация**

data = pd.read\_csv('test.csv')

data



m = len(data)

x = np.array(data['test\_1'])

temp = np.array(data['test\_2'])

y = np.array(data['class'])

x = x.reshape((m, 1))

temp = temp.reshape((m, 1))

y = y.reshape((m, 1))

X = np.hstack((temp,x))

print(X[:5])

[[ 0.68494 -0.092742]

[ 0.69225 -0.21371 ]

[ 0.50219 -0.375 ]

[ 0.46564 -0.51325 ]

[ 0.2098 -0.52477 ]]

plt.figure(figsize=(13,10))

plotData(data['test\_1'][data['class'] == 0], data['test\_2'][data['class'] == 0], 200, 'o', 'blue', 'Class 1' )

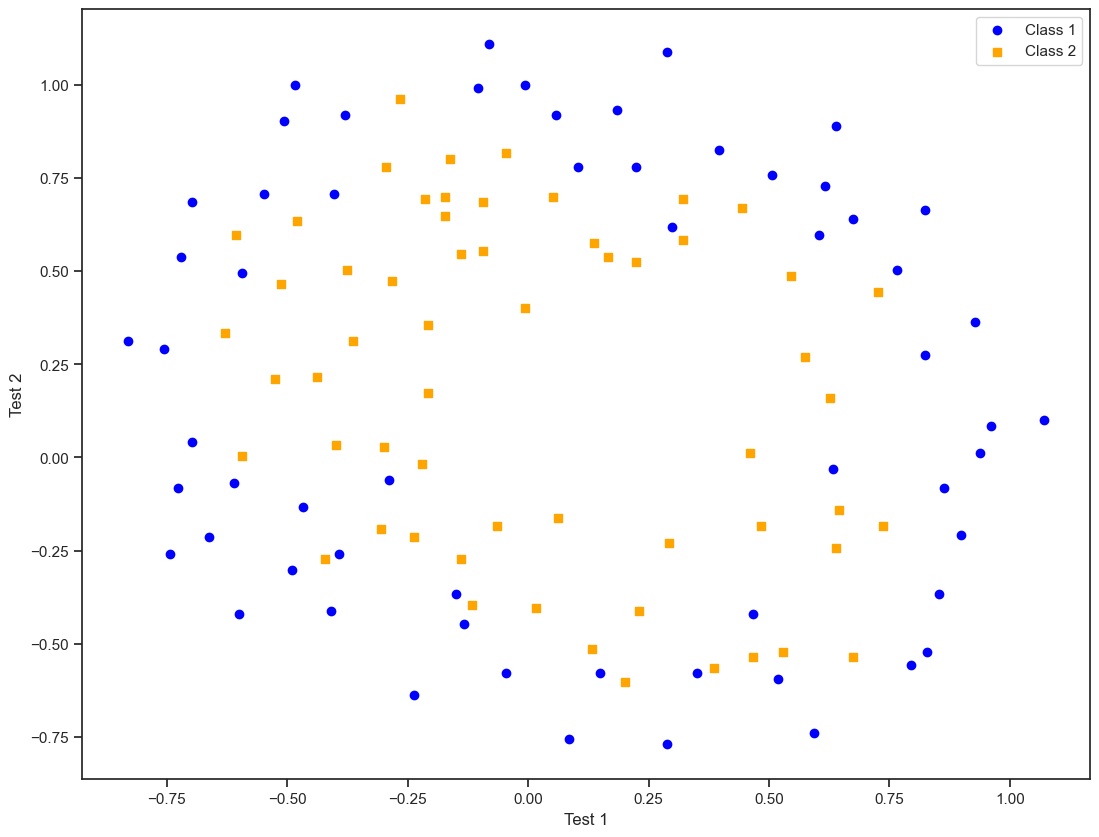
plotData(data['test\_1'][data['class'] == 1], data['test\_2'][data['class'] == 1], 200, 's', 'orange', 'Class 2')

plt.xlabel('Test 1')

plt.ylabel('Test 2')

plt.legend()

plt.show()



def mapFeature(X1, X2):

degree = 6

out = np.ones(( X1.shape[0], sum(range(degree + 2)) ))

curr\_column = 1

for i in range(1, degree + 1):

for j in range(i+1):

out[:,curr\_column] = np.power(X1,i-j) \* np.power(X2,j)

curr\_column += 1

return out

X = mapFeature(X[:,0], X[:,1])

X.shape

(118, 28)

initial\_theta = np.zeros((X.shape[1], 1))

initial\_theta

array([[0.],

[0.],

[0.],

[0.],

[0.],

[0.],

[0.],

[0.],

[0.],

[0.],

[0.],

[0.],

[0.],

[0.],

[0.],

[0.],

[0.],

[0.],

[0.],

[0.],

[0.],

[0.],

[0.],

[0.],

[0.],

[0.],

[0.],

[0.]])

def costFunctionReg(theta, X, y, lambda\_reg):

"""

Вычисление функции стоимости и значения градиента(ов)для

задачи логистической регрессии с регуляризацией

costFunctionReg(theta, X, y, lambda\_reg) вычисляет функцию стоимости, используя

theta в качестве параметра логистической регрессии, а также значение(я)

градиентов

"""

# Ваш код здесь

m = len(y)

J = 0

h\_theta = []

first\_part = []

second\_part = []

theta\_sum = 0

for i in range (m):

h\_theta.append( sigmoid(np.matmul(np.transpose(theta), X[i])) )

first\_part.append( -(y[i] \* math.log(h\_theta[i])) )

second\_part.append( (1 - y[i]) \* math.log(1 - h\_theta[i]) )

J += first\_part[i] - second\_part[i]

for i in range (theta.shape[0]):

theta\_sum += math.pow(theta[i], 2)

J = J / m + ( (lambda\_reg \* theta\_sum) / (2 \* m) )

return J

cost = costFunctionReg(initial\_theta, X, y, 1)

print('Значение функции стоимости при начальном значении theta (нулевых):', cost)

Значение функции стоимости при начальном значении theta (нулевых): [0.69314718]

def gradFunctionReg(theta, X, y, lambda\_reg):

grad = np.zeros(theta.shape)

# изменил функцию так как для для theta\_0 градиент вычисляется без суммирования с lambda

# (на нулевом векторе theta действие функции не видно, но смысл в изменении есть)

# и я убрал транспонирование sigmoid( np.dot(X,theta) ) так как при вычитании с y он даст неверную матрицу

# вместо этого поставил транспонирование на sigmoid( np.dot(X,theta) ) - y чтобы получить матрицу размером

# (1, 118) и при умножении на X получить значение размерностью (1,28) и транспонируется в (28,1)

grad = (1./m) \* np.dot( (sigmoid( np.dot(X,theta) ) - y).T , X).T + ( float(lambda\_reg) / m )\*theta

grad[0] = (1./m) \* np.dot( (sigmoid( np.dot(X,theta) ) - y).T, X[:, 0])

return grad

gradFunctionReg(initial\_theta, X, y, 1)

array([[8.47457627e-03],

[7.77711864e-05],

[1.87880932e-02],

[3.76648474e-02],

[1.15013308e-02],

[5.03446395e-02],

[2.34764889e-02],

[8.19244468e-03],

[7.32393391e-03],

[1.83559872e-02],

[3.93028171e-02],

[3.09593720e-03],

[1.28600503e-02],

[2.23923907e-03],

[3.93486234e-02],

[3.10079849e-02],

[4.47629067e-03],

[5.83822078e-03],

[3.38643902e-03],

[4.32983232e-03],

[1.99707467e-02],

[3.87936363e-02],

[1.37646175e-03],

[7.26504316e-03],

[4.08503006e-04],

[6.31570797e-03],

[1.09740238e-03],

[3.10312442e-02]])

**Задание 6: Регуляризация и точность**

Дополнительное задание: В этом задании, варьируя значения lambda, выяснить влияние регуляризации на поведение границы разделения классов

Указание: Продемонстрировать эффект для lambda (0, 1, 10, 100).

def plotDecisionBoundary(theta, lambda\_reg):

plt.figure(figsize=(10,10))

plotData(data['test\_1'][data['class'] == 0], data['test\_2'][data['class'] == 0], 200, 'o', 'blue', 'Class 1' )

plotData(data['test\_1'][data['class'] == 1], data['test\_2'][data['class'] == 1], 200, 's', 'orange', 'Class 2')

u = np.linspace(-1, 1.5, 50)

v = np.linspace(-1, 1.5, 50)

uu, vv = np.meshgrid(u, v)

z = np.zeros(( len(u), len(v) ))

for i in range(len(u)):

for j in range(len(v)):

z[i,j] = np.dot(mapFeature(np.array([u[i]]), np.array([v[j]])),theta)

z = np.transpose(z)

plt.contour(u, v, z, levels=[0], colors='black', linewidths=3)

plt.title('\n lambda = {:d}, Точность обучения: {:.2%} \n'.format(lambda\_reg, predict(X, theta, y)/X.shape[0]),\

fontsize=20)

plt.xlabel('Test 1', fontsize=20)

plt.ylabel('Test 2', fontsize=20)

plt.yticks(size = 15)

plt.xticks(size = 15)

plt.yticks(size = 15)

plt.xticks(size = 15)

plt.legend(fontsize=15)

plt.show()

from scipy.optimize import fmin\_bfgs

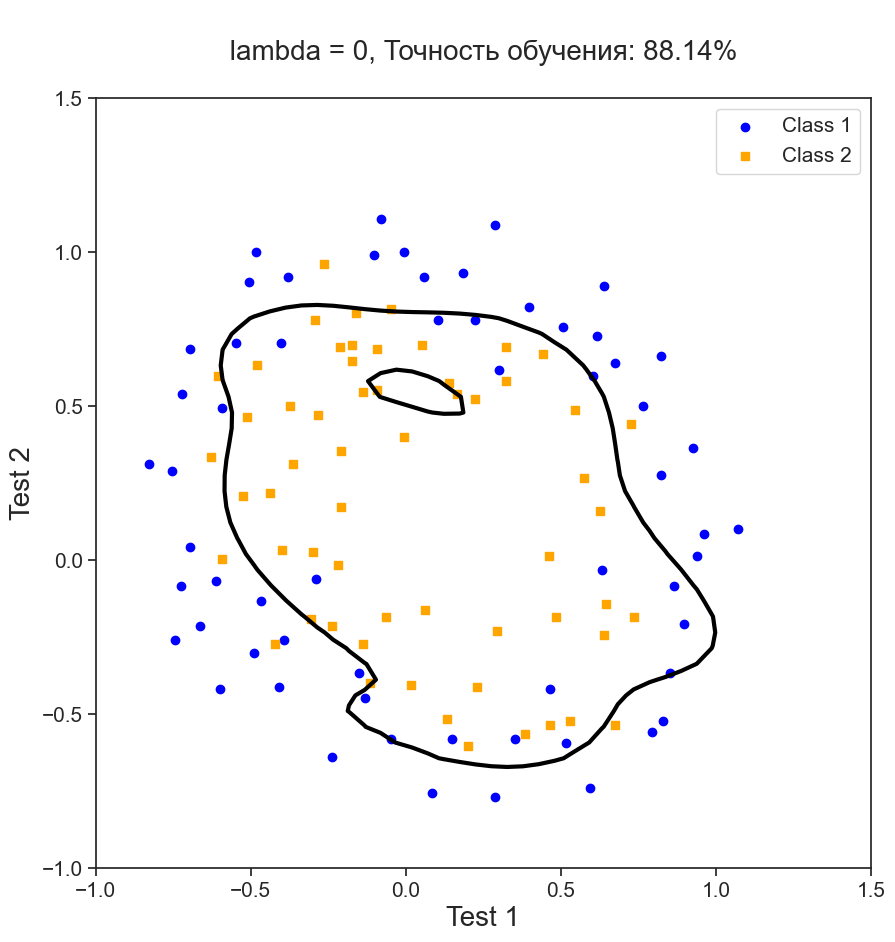
lambda\_reg = [0, 1, 10, 100]

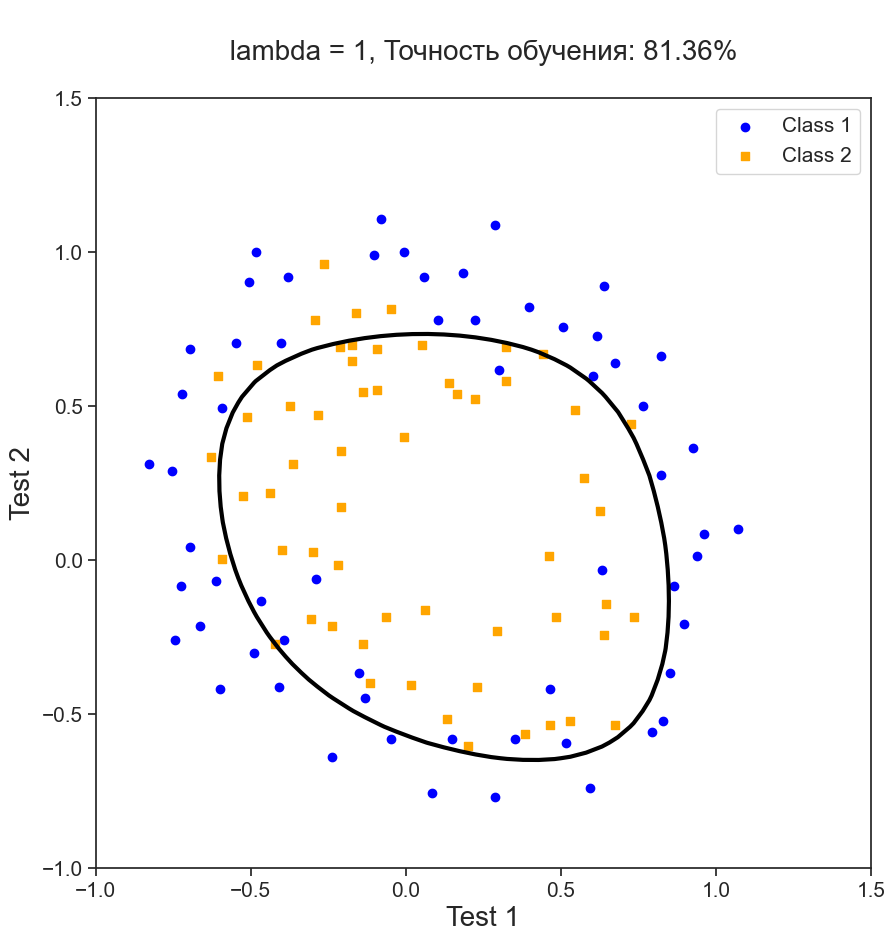
for lambda\_i in lambda\_reg:

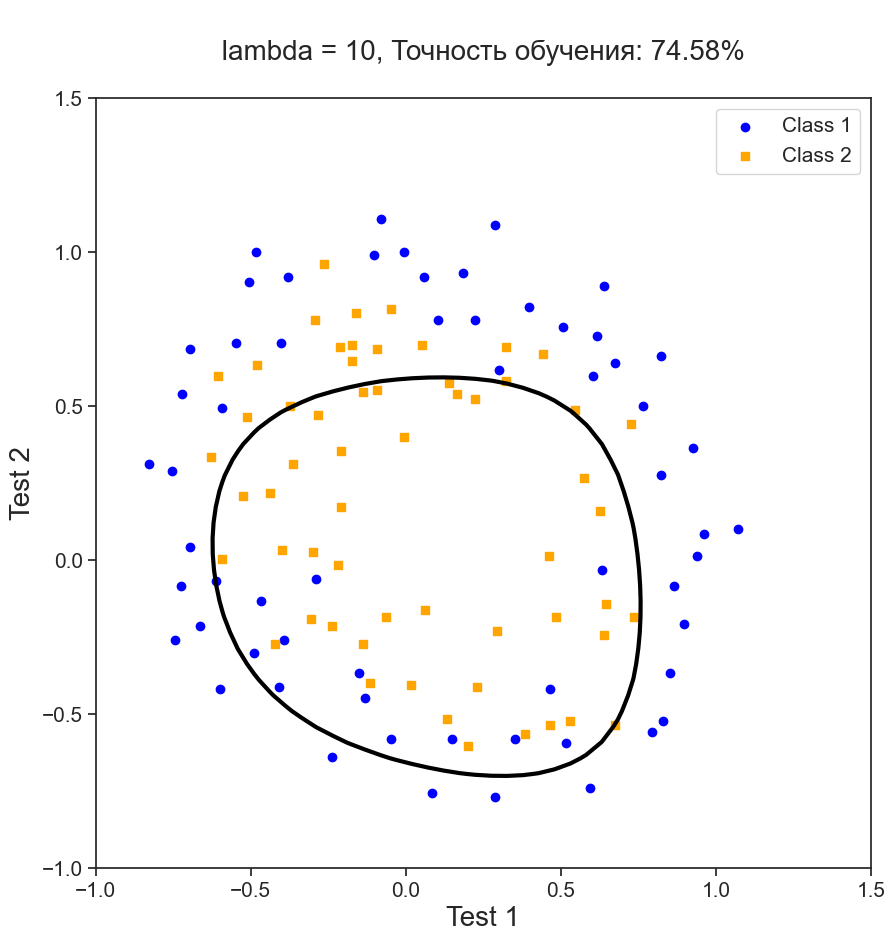
myargs=(X, y, lambda\_i)

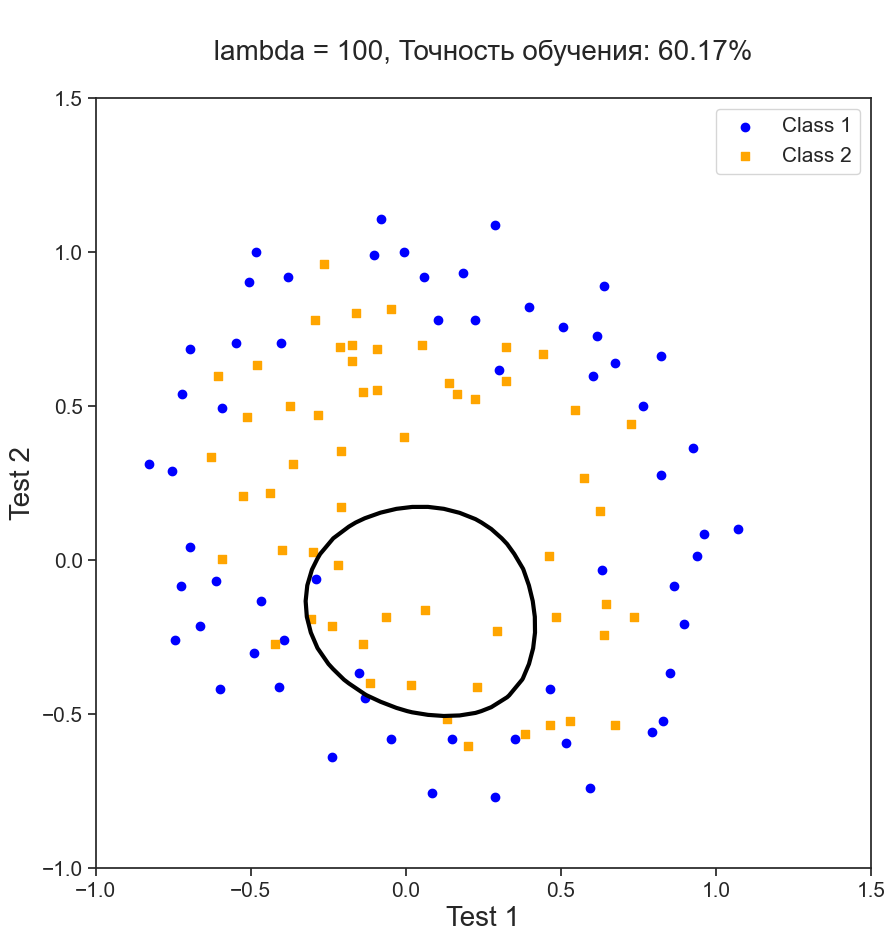
theta = fmin\_bfgs(costFunctionReg, x0 = initial\_theta, args = myargs)

plotDecisionBoundary(theta, lambda\_i)









Чем больше значение , тем менее точно обучение классификатора (так как при больших значениях мы пренебрегаем малыми значениями )

lambda\_reg = 0

myargs=(X, y, lambda\_reg)

theta = fmin\_bfgs(costFunctionReg, x0 = initial\_theta, args = myargs)

print('Максимальная точность обучения: {:.2%} \n'.format(predict(X, theta, y)/X.shape[0]))

Максимальная точность обучения: 88.14%