МИНИСТЕРСТВО ТРАНСПОРТА РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ  
ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА (МИИТ)

ИНСТИТУТ ТРАНСПОРТНОЙ ТЕХНИКИ И СИСТЕМ УПРАВЛЕНИЯ

Кафедра «Управление и защита информации»

**ОТЧЁТ**

по лабораторной работе №2

Линейная нелинейная модели

по дисциплине

«Гибкие технологии программирования»

|  |  |
| --- | --- |
| Выполнил: | Кузнецов П.И. |
| Группа:  Вариант: | ТУУ-411 |
| Проверил: | Доц. Зольникова Н.Н. |

Москва 2022

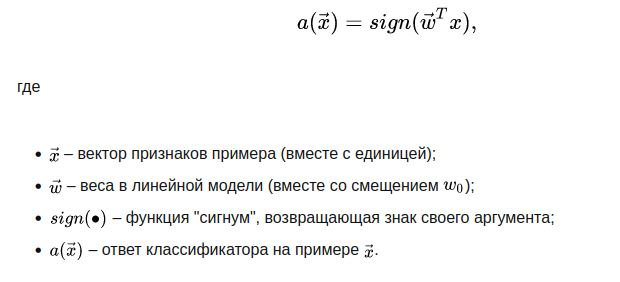
**Формулировка задачи:**

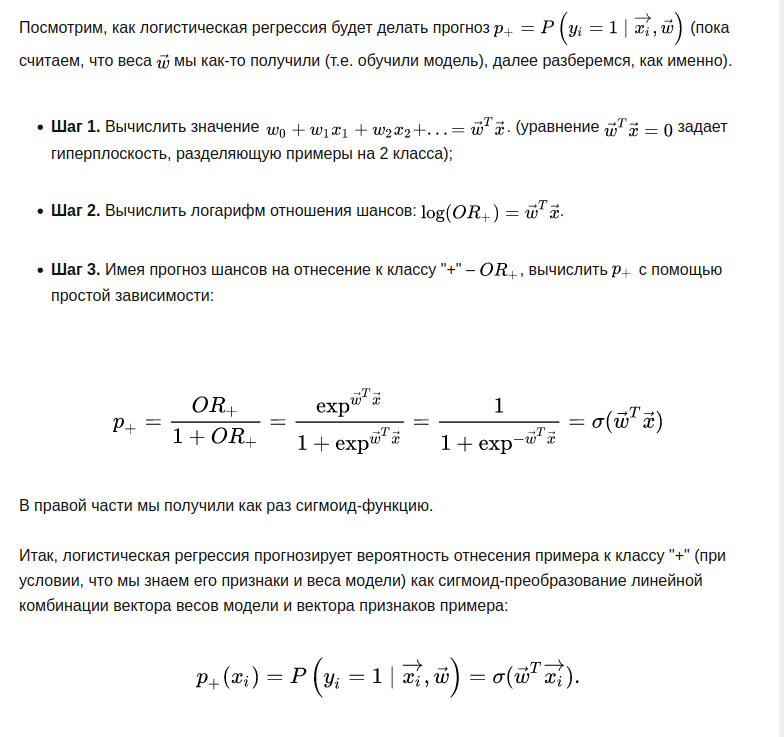
Обучить единичный нейрон методом логистической регрессии без регуляризации. Сравнить результаты для линейной и нелинейной модели. Вычислить точность работы системы на тестовых и обучающих данных. (Использовать файл данных ex2data1\2.txt)

**Теоретическая часть:**

Основная идея линейного классификатора заключается в том, что признаковое пространство может быть разделено гиперплоскостью на два полупространства, в каждом из которых прогнозируется одно из двух значений целевого класса.  
Если это можно сделать без ошибок, то обучающая выборка называется линейно разделимой.

Один из самых простых линейных классификаторов получается на основе регрессии вот таким образом:



**Как модель обучается?**

### По принципу максимального правдоподобия

Теперь посмотрим, как из принципа максимального правдоподобия получается оптимизационная задача, которую решает логистическая регрессия, а именно, – минимизация логистической функции потерь.  
Только что мы увидели, что логистическая регрессия моделирует вероятность отнесения примера к классу "+" как





Тогда для класса "-" аналогичная вероятность:



Оба этих выражения можно объединить в одно:



**Выражен****ие **

**называется отступом (margin) классификации на объекте** Xi **.  Если Xi неотрицателен, модель не ошибается на объекте , если же отрицателен – значит, класс для  спрогнозирован неправильно.**

Значит, выражение

это своего рода "уверенность" модели в классификации объекта Xi:

* если отступ большой (по модулю) и положительный, это значит, что метка класса поставлена правильно, а объект находится далеко от разделяющей гиперплоскости (такой объект классифицируется уверенно). На рисунке 1 – X3.
* если отступ большой (по модулю) и отрицательный, значит метка класса поставлена неправильно, а объект находится далеко от разделяющей гиперплоскости (скорее всего такой объект – аномалия, например, его метка в обучающей выборке поставлена неправильно). На рисунке 1 – X1.
* если отступ малый (по модулю), то объект находится близко к разделяющей гиперплоскости, а знак отступа определяет, правильно ли объект классифицирован. На рисунке 1 – X2 и X4.

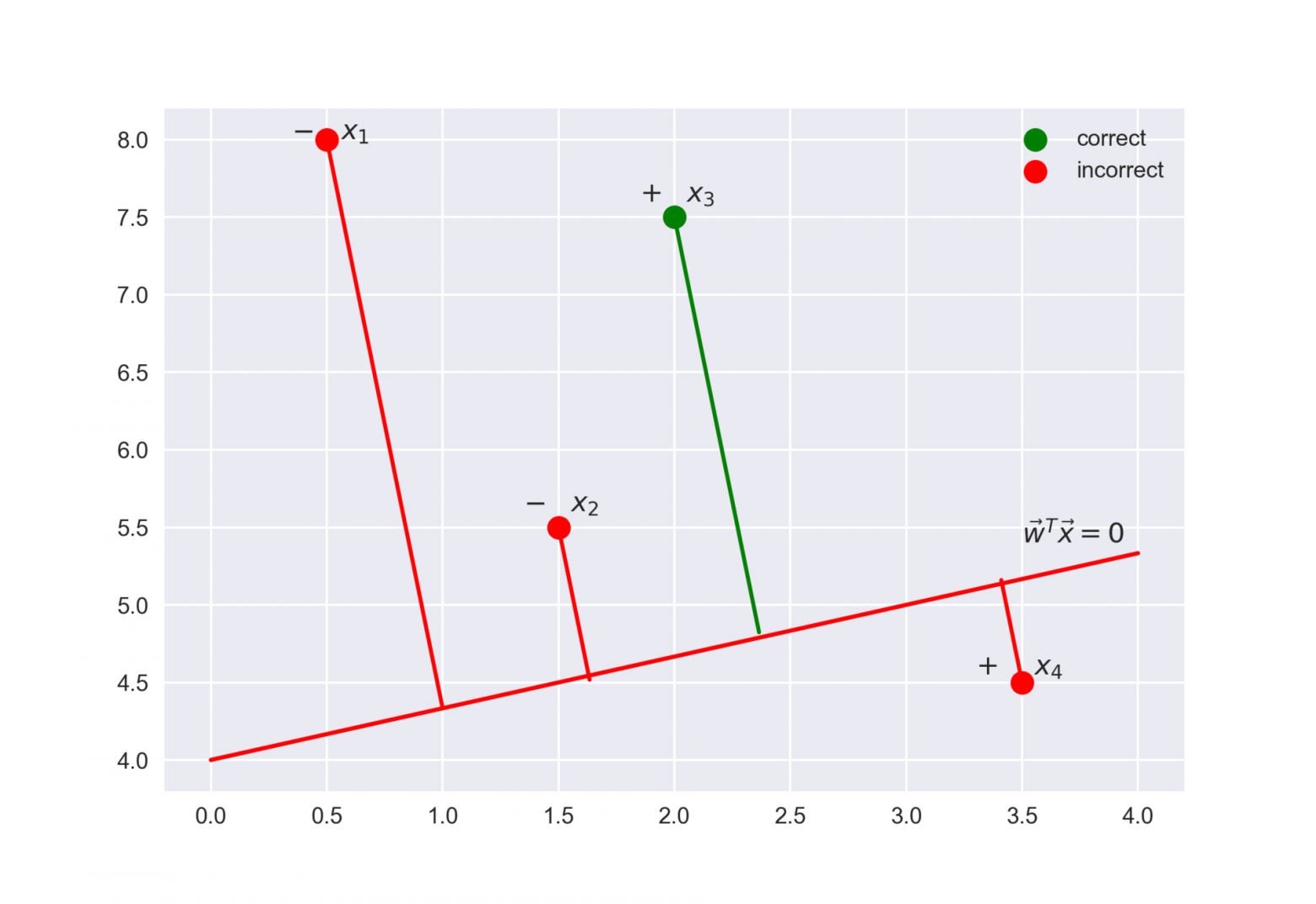


Рисунок-1. Классификация

## **Полиномиальные признаки**

Полиномиальные признаки - это те признаки, которые создаются путем возведения существующих признаков в степень.

Например, если в наборе данных есть одна входная характеристика X, то полиномиальная функция будет добавлением новой функции (столбца), где значения вычисляются путем возведения в квадрат значений в X, например X ^ 2.

Полиномиальными признаками до степени  для двух переменных  и  мы называем следующие:



Например, для  d=3 это будут следующие признаки:



Этот процесс можно повторить для каждой входной переменной в наборе данных, создав преобразованную версию каждой.

По существу, полиномиальные признаки представляют собой тип разработки признаков, например создание новых входных признаков на основе существующих признаков.

Значок «степень”Полинома используется для управления количеством добавленных функций, например, степень 3 добавит две новые переменные для каждой входной переменной. Обычно используется небольшая степень, например 2 или 3.

Вообще говоря, использование d больше 3 или 4 необычно, потому что при больших значениях d полиномиальная кривая может стать слишком гибкой и принимать очень странные формы.

Также часто добавляются новые переменные, которые представляют взаимодействие между функциями, например, новый столбец, который представляет одну переменную, умноженную на другую. Это тоже можно повторить для каждой входной переменной, там самым повысив корреляцию признаков.

Возведенная в квадрат входная переменная изменит распределение вероятностей, разделив маленькие и большие значения.

Это разделение может помочь некоторым алгоритмам машинного обучения(в нашем случае логистическая регрессия) делать более точные прогнозы.

**Выполнение задачи:**

Создадим модель используя tensorflow.keras

Cначала используем нелинейную регуляризация без добавления полиномиальных признаков.

x, y = load\_data('ex2data1.txt')  
 #полиномиальные признаки  
 #poly = PolynomialFeatures(degree=7)  
 #x = poly.fit\_transform(x)  
 #разбиваем датасет  
 train\_X, test\_X, test\_Y, train\_Y = train\_test\_split(x, y, random\_state=15, test\_size=0.5)  
  
 from keras.regularizers import l1\_l2  
  
 #создаем модель с бинарной классификацией  
 model = tf.keras.models.Sequential()  
 #1 слой Выравнивает вход. Не влияет на размер партии.  
 model.add(tf.keras.layers.Flatten())  
 #2 слой Dense реализует операцию: output = activation(dot(input, kernel) + bias), где активация  
 # — это функция активации по элементам,  
 # переданная в качестве аргумента активации, кернел — это матрица весов, созданная слоем,  
 # а смещение — это вектор смещения, созданный слоем (применимо только в случае, если use\_bias — True).  
 model.add(tf.keras.layers.Dense(1, activation=tf.nn.sigmoid, input\_dim=x.shape[1]))  
 #Компилируем модель оптимизатор= rmsprop  
 #функция потерь бинарная энтропия  
 model.compile(optimizer='rmsprop', loss='binary\_crossentropy')  
 #тренируем модель 100 эпох  
 model.fit(train\_X, train\_Y, epochs=100, verbose=1)  
 #выводим предсказания  
 ans = model.predict(test\_X)  
 print(ans)  
 fig, ax = plot\_decision\_boundary(x, y, model)  
 fig.savefig("output.png")

**На первом шаге загружаем, датасет ex1data1:**

Датасет имеет два признака и два класса

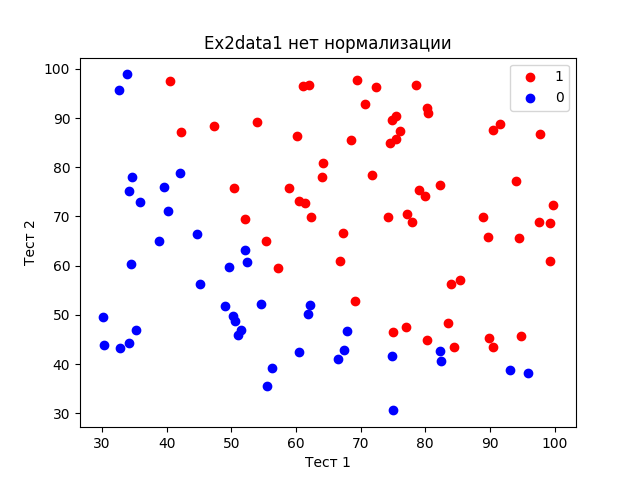


Рисунок-1. Исходные данные

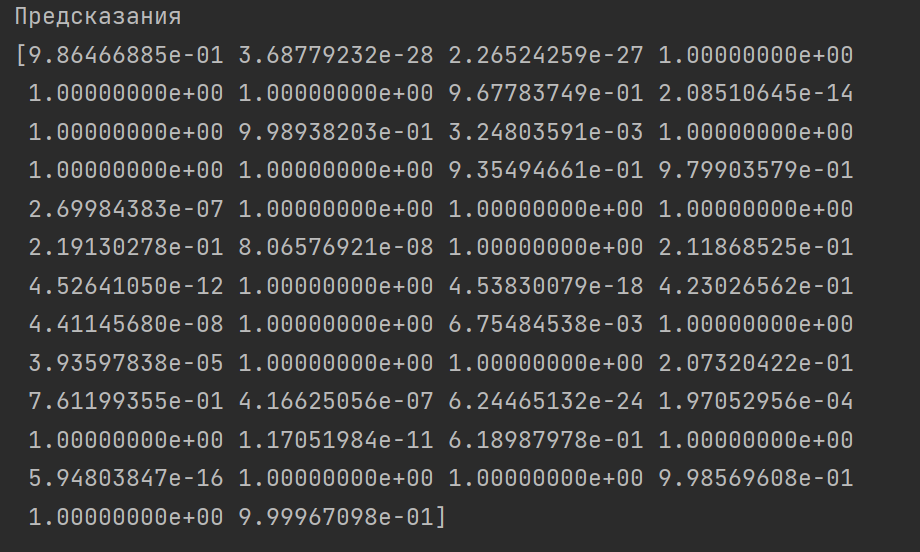


Рисунок-2. Признаки

Разделяем датасет на 30% обучающих даных.

**Обучаем модель,высчитываем точность и смотрим полученные предсказания:**

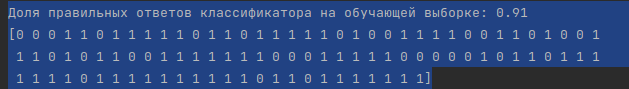
****

Рисунок-3. Полученные предсказания

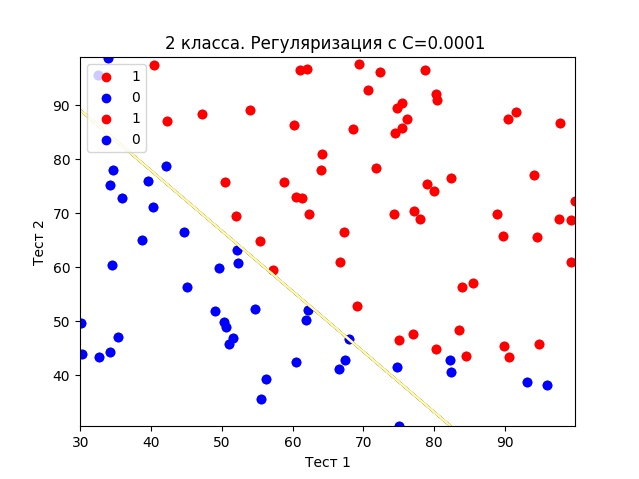


Рисунок-4. Разделяющая прямая

**Создадим нелинейную модель(большей корреляции между признакми) добавим полиномиальные признаки 3 степени**

poly = PolynomialFeatures(degree=3)  
x = poly.fit\_transform(x)

**Получаем предсказния:**

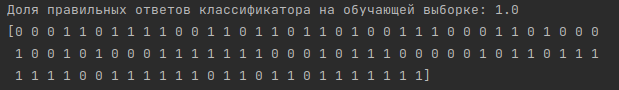


Рисунок-5. Предсказания

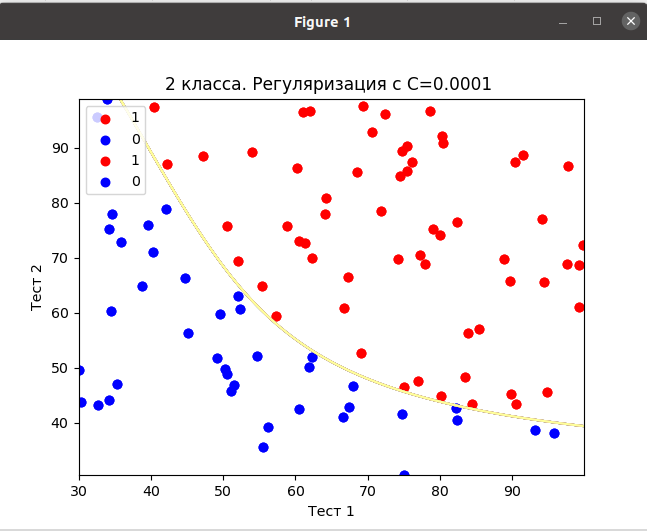


Рисунок-6. Разделяющая прямая на тестовых данных c добавлением полиномиальных признаков

**Построение разделяющей прямой на датасете ex1data2:**

Для начала обучим модель без использования полниномиальных признаков

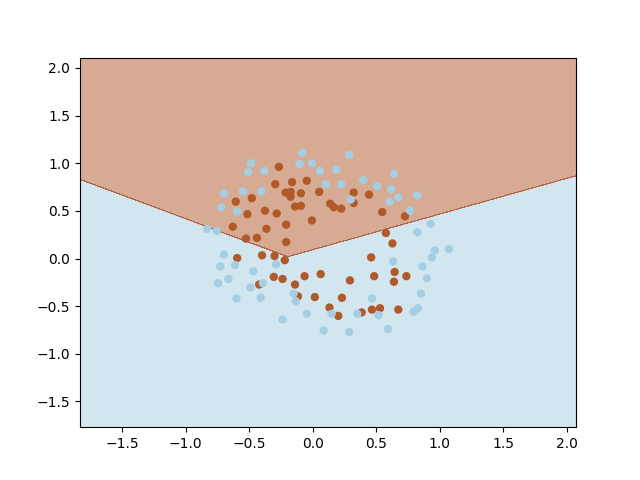


Рисунок-7. Ex1data2 без полиномиальных признаков

Как видно точность модели неудовлетворительная.

Test fraction correct (NN-Accuracy) keras = 0.56

**Добавим признаки 7 степени:**

Test fraction correct (NN-Accuracy) keras = 0.86

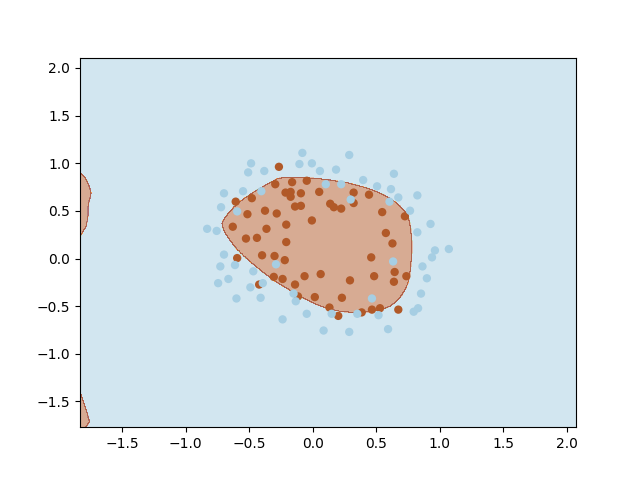


Рисунок-8. Ex1data2 с полиномиальными признаками 7 степени

Вывод: для создания данной программы, были использованы библиотеки scikit , tensorflow