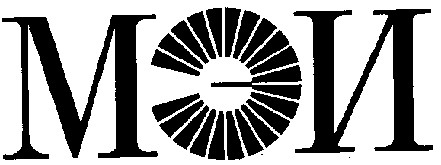
Национальный исследовательский университет «МЭИ»



Кафедра релейной защиты и автоматизации энергосистем

**Лабораторная работа№ 1**

###### *«*МЕТОД К-БЛИЖАЙШИХ СОСЕДЕЙ. ЛОГИСТИЧЕСКАЯ РЕГРЕССИЯ »

|  |  |
| --- | --- |
| Выполнил: | Кондрашов А.В. |
| Группа: | Э-13м-21 |
| Проверил: |  |

Москва 2022

Оглавление

[1. Анализ и предварительная обработка данных 3](#_Toc114067909)

[2. Поиск оптимального значения К для метрического метода и С для линейного 3](#_Toc114067910)

[3. Листинг кода 6](#_Toc114067911)

# Анализ и предварительная обработка данных

В качестве исходных данных предоставлен CSV файл, в котором содержится 32 столбца. В этом файле представлена информация о болезни и измеренные параметры пациентов. Проанализировав, таблицу было принято решение удалить столбец Id так как никакой полезной информации он не несет.

# Поиск оптимального значения К для метрического метода и С для линейного

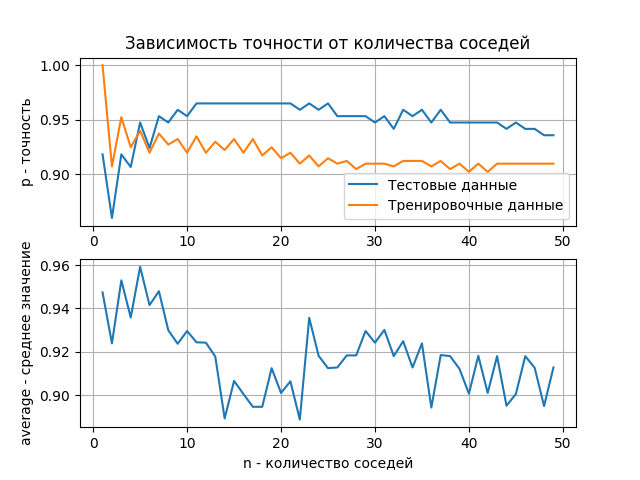


Рисунок . ­­­­­– Метод К-ближайших соседей без нормализации

Average ­­­­­– среднее значение результатов кросс-валидации. Оптимальное количество соседей составляет 11, значение точности при этом составляет 0,965. Из графиков можно заметить, что в целом переобучение имеет место, но в малой степени.

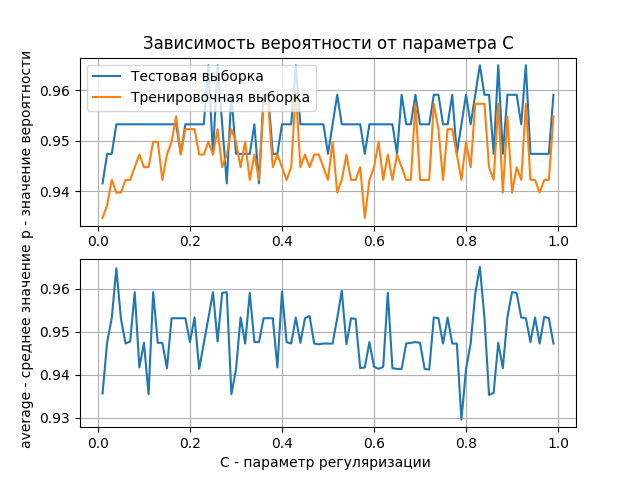


Рисунок . – Логистическая регрессия без нормализации

Оптимальное значение параметра С составляет 0,24 ,значение точности при этом составляет 0,965. Из графиков можно заметить, что в целом переобучение мало.

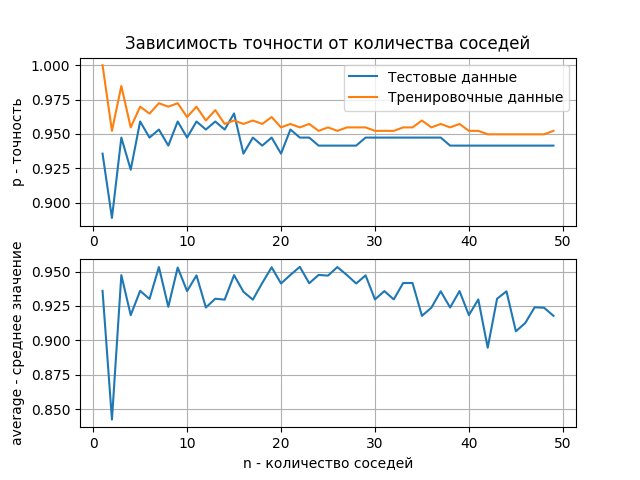


Рисунок . – Метод К-ближайших соседей с нормализацией

Оптимальное количество соседей составляет 15, значение точности при этом составляет 0,965. Можно заметить, что в целом точность не изменилась при нормализации, однако изменилось оптимальное количество соседей. В данном случае переобучение невелико, поэтому можно заключить, что в целом модель работает корректно.

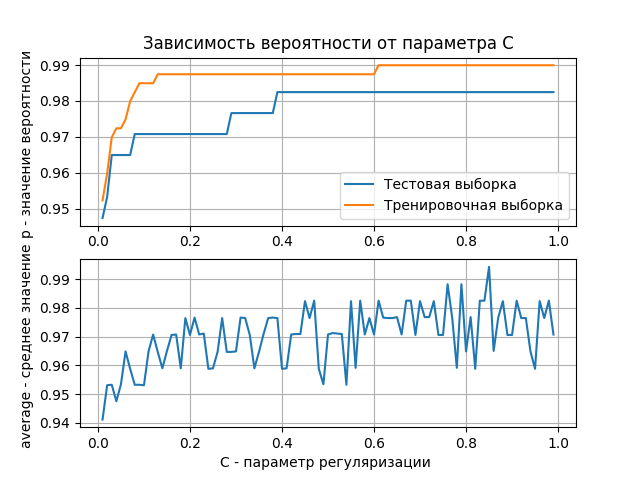


Рисунок . – Логистическая регрессия с нормализацией

Оптимальное значение параметра С составляет 0,39 ,значение точности при этом составляет 0,982. В данном случае результат вероятности улучшился, а также увеличилось значение параметра регуляризации. В целом есть некоторое переобучение, но оно не велико и можно заключить, что модель обучена корректно .

# Листинг кода

import numpy as np  
import pandas as pd  
from sklearn.model\_selection import train\_test\_split  
from sklearn.neighbors import KNeighborsClassifier  
from sklearn.metrics import accuracy\_score  
import matplotlib.pyplot as plt  
from sklearn.model\_selection import KFold  
from sklearn.model\_selection import cross\_val\_score  
from sklearn.linear\_model import LogisticRegression  
from sklearn.preprocessing import StandardScaler  
  
  
def draw\_graph(x\_M, y\_M, x\_B, y\_B, labelX, labelY):  
 plt.figure(figsize=(7, 8))  
 plt.title(" Распределение видов \n по двум признакам ")  
 plt.scatter(x\_M, y\_M, marker='^', label="M")  
 plt.scatter(x\_B, y\_B, marker='o', label="B")  
 plt.xlabel(labelX)  
 plt.ylabel(labelY)  
 plt.legend()  
 plt.grid()  
  
  
data\_frame = pd.get\_dummies(pd.read\_csv("breast\_cancer.csv"), columns=["diagnosis"])  
tags\_knn = data\_frame.get(["diagnosis\_B", "diagnosis\_M"]).to\_numpy()  
tags\_lr = data\_frame.get("diagnosis\_B").to\_numpy()  
  
# Построение графиков по двум признакам  
# data\_M = data\_frame[data\_frame.diagnosis\_M == 1]  
# data\_B = data\_frame[data\_frame.diagnosis\_B == 1]  
# field1 = "texture\_mean"  
# field2 = "area\_mean"  
#  
# draw\_graph(data\_M.get([field1]), data\_M.get([field2]),  
# data\_B.get([field1]), data\_B.get([field2]),  
# field1, field2)  
  
# Неплохое сочетание признаков  
# field1 = "radius\_worst"  
# field2 = "texture\_worst"  
# draw\_graph(data\_M.get([field1]), data\_M.get([field2]),  
# data\_B.get([field1]), data\_B.get([field2]),  
# field1, field2)  
# plt.show()  
  
data\_frame.drop(["Unnamed: 32", "diagnosis\_B", "diagnosis\_M", "id"], axis=1, inplace=True)  
  
  
def knNeighbors(data\_set, answers, isScaler):  
 number\_of\_neighbors = list()  
 predict\_test\_set = list()  
 predict\_train\_set = list()  
 average\_value = list()  
  
 if isScaler:  
 scaler = StandardScaler()  
 data = scaler.fit\_transform(data\_set)  
 else:  
 data = data\_set.to\_numpy()  
  
 x\_train, x\_test, y\_train, y\_test = train\_test\_split(data, answers, test\_size=0.3, random\_state=0)  
 kf = KFold(n\_splits=5, shuffle=True)  
  
 for i in range(1, 50):  
 knn = KNeighborsClassifier(i)  
 # for train\_index, test\_index in kf.split(data):  
 # knn.fit(data[train\_index], answers[train\_index])  
 knn.fit(x\_train, y\_train)  
 number\_of\_neighbors.append(i)  
 predict\_test\_set.append(accuracy\_score(y\_test, knn.predict(x\_test)))  
 predict\_train\_set.append(accuracy\_score(y\_train, knn.predict(x\_train)))  
 array = cross\_val\_score(knn, x\_test, y\_test, cv=kf, scoring='accuracy')  
 average = sum(array) / len(array)  
 average\_value.append(average)  
 print(f"Оптимальное значение количества соседей = {predict\_test\_set.index(max(predict\_test\_set)) + 1}")  
 print(f"Значение точности классификации при этом {max(predict\_test\_set)}")  
  
 plt.subplot(2, 1, 1)  
 plt.plot(number\_of\_neighbors, predict\_test\_set, label="Тестовые данные")  
 plt.plot(number\_of\_neighbors, predict\_train\_set, label="Тренировочные данные")  
 plt.title("Зависимость точности от количества соседей")  
 plt.ylabel("p - точность")  
 plt.legend()  
 plt.grid()  
 plt.subplot(2, 1, 2)  
 plt.plot(number\_of\_neighbors, average\_value)  
 plt.grid()  
 plt.ylabel("average - среднее значение")  
 plt.xlabel("n - количество соседей")  
 plt.show()  
  
  
def logisticRegression(data\_set, answers, isScaler):  
 list\_of\_c = list()  
 result\_test = list()  
 result\_train = list()  
 average\_value = list()  
  
 if isScaler:  
 scaler = StandardScaler()  
 data = scaler.fit\_transform(data\_set)  
 else:  
 data = data\_set.to\_numpy()  
 x\_train, x\_test, y\_train, y\_test = train\_test\_split(data, answers, test\_size=0.3, random\_state=0)  
  
 kf = KFold(n\_splits=5, shuffle=True)  
  
 for i in np.arange(0.01, 1, 0.01):  
 lr = LogisticRegression(C=i)  
 # for train\_index, test\_index in kf.split(data):  
 # lr.fit(data[train\_index], answers[train\_index])  
 lr.fit(x\_train, y\_train)  
 list\_of\_c.append(i)  
 result\_test.append(accuracy\_score(y\_test, lr.predict(x\_test)))  
 result\_train.append(accuracy\_score(y\_train, lr.predict(x\_train)))  
 array = cross\_val\_score(lr, x\_test, y\_test, cv=kf, scoring='accuracy')  
 average = sum(array) / len(array)  
 average\_value.append(average)  
 print(f"Оптимальное значение параметра С = {list\_of\_c[result\_test.index(max(result\_test))]:1.2f}")  
 print(f"Значение точности классификации при этом {max(result\_test)}:1.2f")  
 plt.subplot(2, 1, 1)  
 plt.plot(list\_of\_c, result\_test, label="Тестовая выборка")  
 plt.plot(list\_of\_c, result\_train, label="Тренировочная выборка")  
 plt.title("Зависимость вероятности от параметра C")  
 plt.ylabel("p - значение вероятности ")  
 plt.legend()  
 plt.grid()  
 plt.subplot(2, 1, 2)  
 plt.plot(list\_of\_c, average\_value)  
 plt.ylabel("average - среднее значение ")  
 plt.xlabel("С - параметр регуляризации ")  
 plt.grid()  
 plt.show()  
  
knNeighbors(data\_frame, tags\_knn, False)  
logisticRegression(data\_frame, tags\_lr, False)  
knNeighbors(data\_frame, tags\_knn, True)  
logisticRegression(data\_frame, tags\_lr, True)