МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

кафедра «Вычислительная техника».

**Лабораторная работа №3 (вариант 5)**

по дисциплине: «Системы искусственного интеллекта»

Выполнил:

студент 4 курса, гр. ИВТВМбд-41

Захарычев Никита Алексеевич.

Проверил:

кандидат технических наук

Святов Кирилл Валерьевич.

г. Ульяновск, 2017

**Задание**

1. Написать программу, которая разделяет исходную выборку на обучающую и тестовую (training set, validation set, test set), если такое разделение не предусмотрено предложенным набором данных.
2. С использованием библиотеки scikit-learn обучить 2 модели нейронной сети (Perceptron и MLPClassifier) по обучающей выборке. Перед обучением необходимо осуществить масштабирование признаков. Проверить точность модели по тестовой выборке.
3. Проверить точность модели по тестовой выборке.
4. Провести эксперименты и определить наилучшие параметры коэффициента обучения, параметра регуляризации, функции оптимизации. Данные экспериментов необходимо представить в отчете (графики, ход проведения эксперимента, выводы).

**Вариант №5** – Набор данных: EEG Eye State Data Set. Набор данных представляет определить состоянии глаз (открыты или закрыты) при помощи различных параметров.

**Ход работы**

**Подключение библиотек**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.metrics import mean\_squared\_error, r2\_score

from sklearn.metrics import f1\_score

from sklearn.utils import shuffle

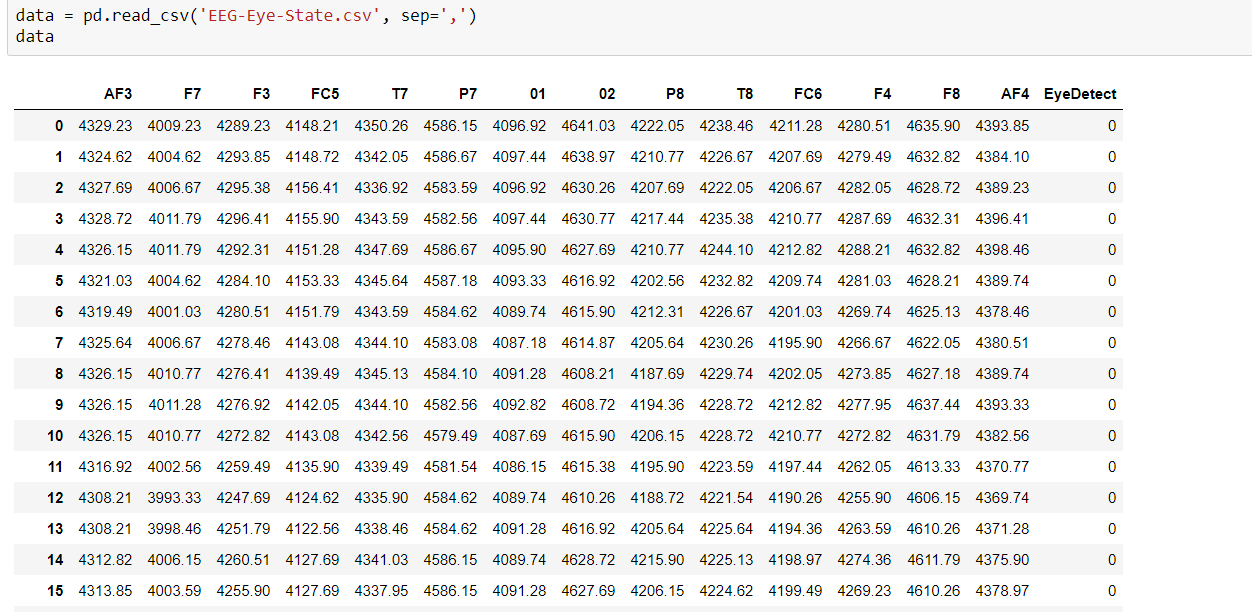
from sklearn.neural\_network import MLPClassifier

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import Perceptron

from sklearn import preprocessing

**Подготовка данных**



Разделяем наши данные для обучения и тестирования:

train\_data = int(math.floor(all\_rows \* 0.8))

test\_data = int(math.floor(all\_rows \* 0.2))

target\_data = int(math.floor(all\_rows \* 0))

Формирование вектора Х для обучения и тестирования

x\_train = x[:train\_data]

x\_test = x[train\_data:(train\_data + test\_data)]

Формирование вектора Y для обучения и тестирования

y\_train = y.iloc[:train\_data]

y\_test = y.iloc[train\_data:(train\_data + test\_data)]

Нормализуем Х

x\_train = preprocessing.scale(x\_train)

x\_test = preprocessing.scale(x\_test)

**Построение модели MLPClassifier**

alphas = [0.09, 0.06, 0.01, 0.009, 0.006, 0.001, 0.0009, 0.0006, 0.0001, 0.00009, 0.00006, 0.00001, 0.000009, 0.000006, 0.000001, 0.0000009, 0.0000006, 0.0000001]

varScr = np.empty(18)

for i in range(len(alphas)):

clas = MLPClassifier(alpha=alphas[i], max\_iter=1500, learning\_rate\_init = 0.003, shuffle=False)

clas.fit(x\_train, y\_train)

y\_pred = clas.predict(x\_test)

varScr[i] = f1\_score(y\_test, y\_pred, average='micro')

print("alpha: " , alphas[i])

print(" score: %.2f" % clas.score(x\_test, y\_test))

print(' Variance score: %.2f' % f1\_score(y\_test, y\_pred, average='micro'))

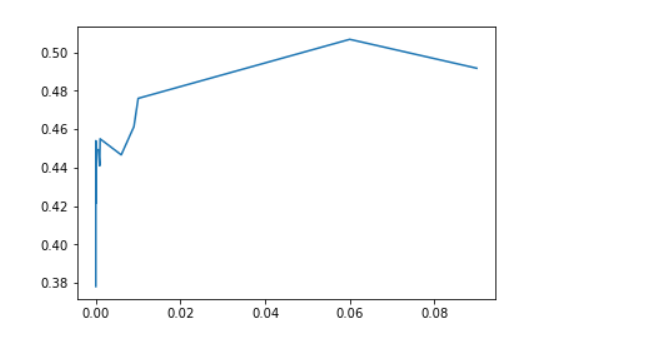
fig = plt.figure()

plt.plot(alphas,varScr)

plt.show()

f1-score представляет собой гармоническое среднее между точностью и полнотой алгоритма. Она стремится к нулю, если точность или полнота стремится к нулю.

График зависимости f1-score от alpha



**Построение модели Perceptron**

for i in range(len(alphas)):

pers = Perceptron(alpha=alphas[i])

pers.fit(x\_train, y\_train)

y\_pred = pers.predict(x\_test)

varScr[i] = f1\_score(y\_test, y\_pred, average='micro')

print("alpha: " , alphas[i])

print('Variance score: %.2f' % f1\_score(y\_test, y\_pred, average='micro'))

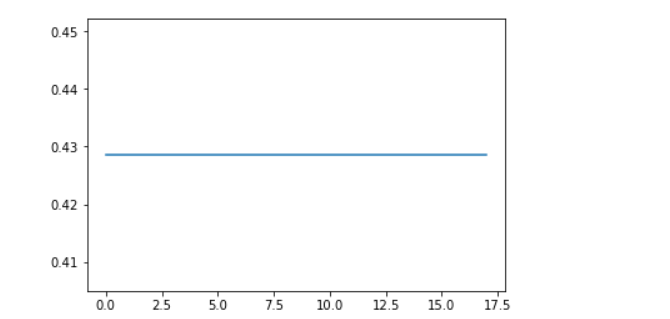
print("score: %.2f" % pers.score(x\_test, y\_test))

fig = plt.figure()

plt.plot(varScr)

plt.show()

График зависимости f1-score от alpha.



В результате проделанной работы можно сделать вывод, что в данной модели оценка f1-score не зависит от коэффициента регуляризации alpha.

**Вывод**

Выполнив данную лабораторную работу, были улучшены навыки в программировании на языке Python. Были изучены библиотеки Python для построения Perceptron и MLPClassifier моделей.

**Исходный код**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.metrics import mean\_squared\_error, r2\_score

from sklearn.metrics import f1\_score

from sklearn.utils import shuffle

from sklearn.neural\_network import MLPClassifier

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import Perceptron

from sklearn import preprocessing

data = pd.read\_csv('EEG-Eye-State.csv', sep=',')

data

y = data.iloc[:, -1:]

y

y = data.iloc[:, -1:]

y

import math

all\_rows = 14980

train\_data = int(math.floor(all\_rows \* 0.8))

test\_data = int(math.floor(all\_rows \* 0.2))

target\_data = int(math.floor(all\_rows \* 0))

x\_train = x[:train\_data]

x\_test = x[train\_data:(train\_data + test\_data)]

y\_train = y.iloc[:train\_data]

y\_test = y.iloc[train\_data:(train\_data + test\_data)]

x\_train = preprocessing.scale(x\_train)

x\_test = preprocessing.scale(x\_test)

alphas = [0.09, 0.06, 0.01, 0.009, 0.006, 0.001, 0.0009, 0.0006, 0.0001, 0.00009, 0.00006, 0.00001, 0.000009, 0.000006, 0.000001, 0.0000009, 0.0000006, 0.0000001]

varScr = np.empty(18)

# average = micro --- Calculate metrics globally by counting the total true positives, false negatives and false positives.

# alpha : --- L2 penalty (regularization term) parameter.

# solver --- adam’ refers to a stochastic gradient-based optimizer proposed by Kingma, Diederik, and Jimmy Ba

# max\_iter --- Maximum number of iterations.

# learning\_rate\_init --- The initial learning rate used.

#It controls the step-size in updating the weights. Only used when solver=’sgd’ or ‘adam’.

for i in range(len(alphas)):

clas = MLPClassifier(alpha=alphas[i], max\_iter=1500, learning\_rate\_init = 0.003, shuffle=False)

clas.fit(x\_train, y\_train)

y\_pred = clas.predict(x\_test)

varScr[i] = f1\_score(y\_test, y\_pred, average='micro')

print("alpha: " , alphas[i])

print(" score: %.2f" % clas.score(x\_test, y\_test))

print(' Variance score: %.2f' % f1\_score(y\_test, y\_pred, average='micro'))

fig = plt.figure()

plt.plot(alphas,varScr)

plt.show()

for i in range(len(alphas)):

pers = Perceptron(alpha=alphas[i])

pers.fit(x\_train, y\_train)

y\_pred = pers.predict(x\_test)

varScr[i] = f1\_score(y\_test, y\_pred, average='micro')

print("alpha: " , alphas[i])

print('Variance score: %.2f' % f1\_score(y\_test, y\_pred, average='micro'))

print("score: %.2f" % pers.score(x\_test, y\_test))

fig = plt.figure()

plt.plot(varScr)

plt.show()