**Московский государственный технический**

**университет им. Н.Э. Баумана.**

Факультет «Информатика и управление»

Кафедра ИУ5.

Курс «Введение в анализ данных и машинное обучение»

Отчет по лабораторной работе №4

«Предсказание присутствия людей в помещении»

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-42 |  | преподаватель каф. ИУ5 |
| Егор Хаметов |  | Гапанюк Ю.Е. |
| Подпись и дата: |  | Подпись и дата: |

Москва, 2018 г.

**Задание:**

Необходимо решить задачу предсказания обнаружения присутствия людей в помещении. Задача решается в рамках платформы онлайн-конкурсов по машинному обучению TrainMyData.

**Текст программы:**

import pandas as pd

import numpy as np

from scipy import stats

from sklearn import cross\_validation, metrics

from sklearn.tree import DecisionTreeClassifier

dtrain = pd.read\_csv('./train.csv', sep=',')

dtest = pd.read\_csv('./test.csv', sep=',')

dtrain.corr()['Occupancy'].abs().sort\_values(ascending=False)

dtrain = dtrain.drop(['date'], axis=1)

dtrain = dtrain.drop(['Humidity'], axis=1)

dtest = dtest.drop(['date'], axis=1)

dtest = dtest.drop(['Humidity'], axis=1)

if dtrain.columns[dtrain.isnull().values.any()].tolist() == []:

print('Нет пропущенных')

x = dtrain.drop(['Occupancy'], axis=1) # входные фичи

y = dtrain['Occupancy'] # ответ

x\_train, x\_valid, y\_train, y\_valid = cross\_validation.train\_test\_split(x, y, test\_size= 0.1)

model = DecisionTreeClassifier() # или LogisticRegression()

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

y\_valid\_predict = model.predict(x\_valid)

metrics.accuracy\_score(y\_valid\_predict, y\_valid)

y\_test\_predict = model.predict (dtest)

dtest['value'] = y\_test\_predict

dtest.to\_csv('./solution.csv')

Реализация лог регрессии:

Alpha = 0.1

class LogisticRegression:

def sigmoid(self, x): # значение сигмоиды це и есть предсказание (гипотеза)

return 1.0 / (1 + np.exp(-x))

def gradient\_descent(self, x, y):

w = np.ones(1, np.shape(x)[1]) # начальные значения весов = 1

for i in range(1000):

w = w - Alpha \* x.transpose() \* (self.sigmoid(w.transpose() \* x) - y)

return w

def classification\_result(self, x, w):

probability = self.sigmoid(w.transpose() \* x)

if probability >= 0.5: return 1

else return 0