**Московский авиационный институт**

**(Национальный исследовательский университет)**

Факультет: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

Дисциплина: «Искусственный интеллект»

**Лабораторная работа № 2**

Тема: Алгоритмы машинного обучения

Студент: Чекушкин Д.И

Группа: М80-304Б

Преподаватель: Ахмед Самир Халид

**Москва, 2019**

1. Постановка задачи

Требуется реализовать класс на выбранном языке программирования, который реализует один из алгоритмов машинного обучения. Обязательным является наличия в классе двух методов fit, predict. Необходимо проверить работу вашего алгоритма на ваших данных (на таблице и на текстовых данных), произведя необходимую подготовку данных. Также необходимо реализовать алгоритм полиномиальной регрессии, для предсказания значений в таблице. Сравнить результаты с стандартной реализацией sklearn, определить в чем сходство и различие ваших алгоритмов. Замерить время работы алгоритмов.

Вариант: 12 % 6 + 1 = 1 (Логистическая регрессия)

Решения задачи

ЯП: python 3.5.2

Библиотеки:

math - библиотека математических функций

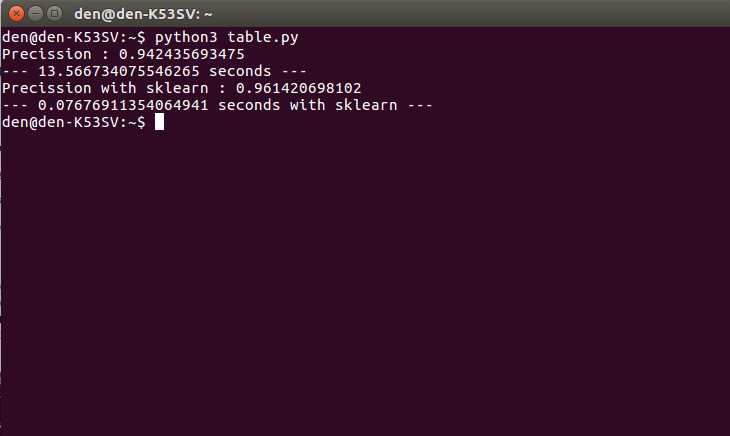
numpy - высокоуровневые математические функции

pandas - обработка и анализ данных

sklearn - статистические функции

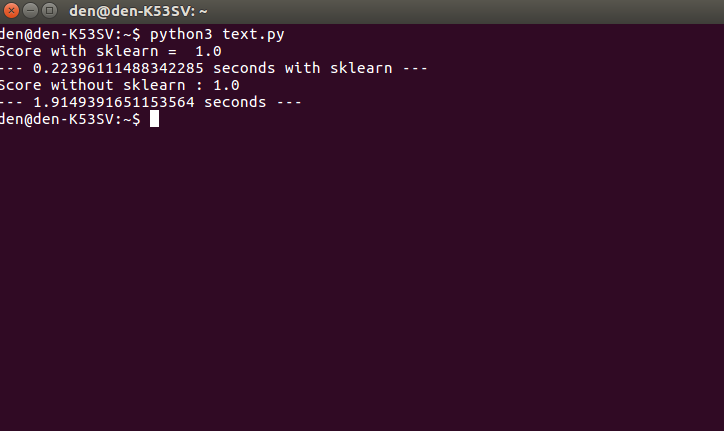
Руководство по использованию программы

Файл table.py содержит код программы, которая обрабатывает табличные данные из файла in.csv, реализцию класса LogisticRegression2 и его сравнение со страндартной реализацией из библиотеки sklearn.

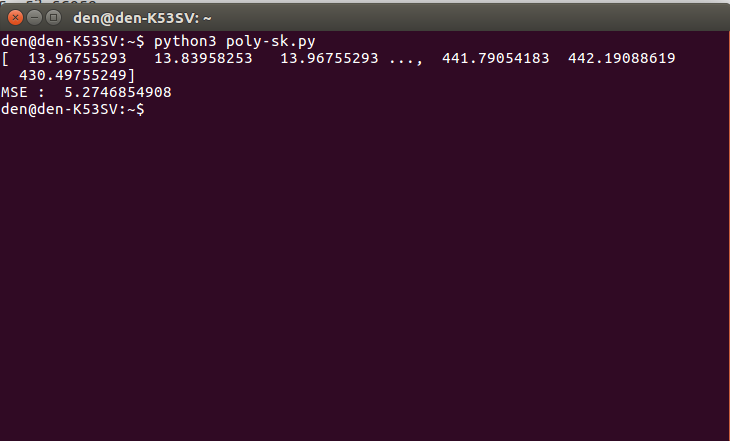


На табличных данных стандартная реализация оказалась более точной и намного более быстрой.

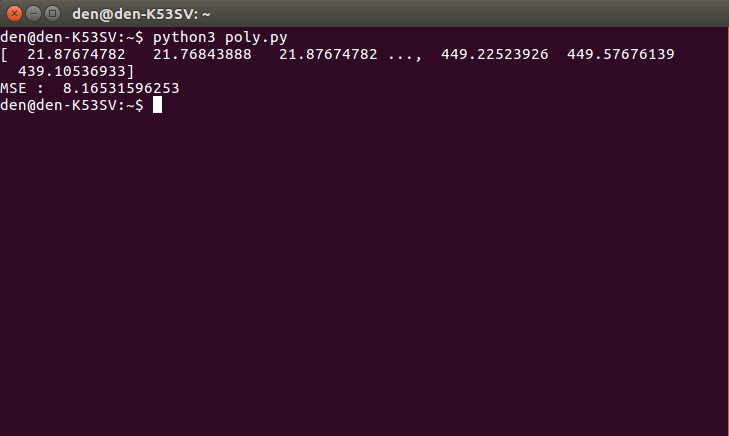
Файл text.py обрабатывает текстовые данные из файла text.csv, также сожержит реализцию класса LogisticRegression2 и его сравнение со страндартной реализацией из библиотеки sklearn

На этот раз точность алгоритмов совпала, но время работы стандартной реализации в разы меньше.

Файл poly-sk.py — построение полиномиальной регрессии с помощью библиотеки sklearn и и вывод предсказанных данных, также вывод коэффициента среднеквадратической ошибки (mean squared error = MSE)



Файл poly.py — построение полиномиальной регрессии и вывод предсказанных данных, также коэффициента среднеквадратической ошибке (mean squared error = MSE)



У полиномиальной регрессии из стандартной библиотеки коэффициент среднеквадратической ошибки ощутимо меньше, можно сделать вывод, что этот алгоритм точнее предсказал результаты.

Листинг класса, реализующего алгоритм логистической регрессия:

class LogisticRegression2:

def \_\_init\_\_(self, lr=0.01, num\_iter=10000, fit\_intercept=True, theta=0, verbose=False):

self.lr = lr

self.num\_iter = num\_iter

self.fit\_intercept = fit\_intercept

self.theta = theta

self.verbose = verbose

def \_\_add\_intercept(self, X):

intercept = np.ones((X.shape[0], 1))

return np.concatenate((intercept, X), axis=1)

def \_\_sigmoid(self, z):

return 1 / (1 + np.exp(-z))

def \_\_loss(self, h, y):

return (-y \* np.log(h) - (1 - y) \* np.log(1 - h)).mean()

def fit(self, X, y):

if self.fit\_intercept:

X = self.\_\_add\_intercept(X)

# инициализируем веса

self.theta = np.zeros(X.shape[1])

for i in range(self.num\_iter):

z = np.dot(X, self.theta)

h = self.\_\_sigmoid(z)

gradient = np.dot(X.T, (h - y)) / y.size

self.theta -= self.lr \* gradient

if(self.verbose == True and i % 10000 == 0):

z = np.dot(X, self.theta)

h = self.\_\_sigmoid(z)

print('loss: ', self.\_\_loss(h, y))

def predict\_prob(self, X):

if self.fit\_intercept:

X = self.\_\_add\_intercept(X)

return self.\_\_sigmoid(np.dot(X, self.theta))

def predict(self, X, threshold=0.5):

return self.predict\_prob(X) >= threshold

Вывод: благодаря проделанной работе, я научился реализовывать алгоритмы машинного обучения, пользоваться стандатной библиотекой sklearn.

Ссылка на репозиторий github: https://github.com/israelcode/mai\_ai\_2019/tree/master/lw2