**Тестовая Задача. Номинация – Big Data.**

**Основное задание будет открыто на хакатоне тем командам, которые пройдут в очный этап.**

**Первичный анализ данных**

В этом заданиии мы будем использовать данные [SOCR](http://wiki.stat.ucla.edu/socr/index.php/SOCR_Data_Dinov_020108_HeightsWeights) по росту и весу 25 тысяч подростков.

*# Необходимые библиотеки*

from plotly.offline import plot, iplot, init\_notebook\_mode

import plotly.graph\_objs as go

from plotly.graph\_objs import \*

import cufflinks as cf

cf.set\_config\_file(offline=True, world\_readable=True, theme='pearl')

import matplotlib.pyplot as plt

import seaborn as sns

import pandas as pd

import numpy as np

%matplotlib inline

**[1]** Считаем данные по росту и весу (*weights\_heights.csv*, приложенный в задании) в объект DataFrame:

*# Ваш код здесь*

df = pd.read\_excel('weights\_heights.xlsx')

df\_height = df['Height']

df\_weight = df['Weight']

**[2]** Постройте 2 раздельные гистограммы распределения роста и веса подростков из выборки *data*. Выделите параметры разным цветом и подпишите оси.

*# Ваш код здесь*

#Гистограма по росту

df\_height.iplot(kind='histogram',title='Height histogram', xTitle='Height (inches)', yTitle='Probability', barmode='stack', bins=100, histnorm='probability', filename='cufflinks/histogram-binning')

#Гистограма по весу

df\_weight.iplot(kind='histogram', color='Red', title='Weight histogram', xTitle='Weight (pounds)', yTitle='Probability', barmode='stack', bins=100, histnorm='probability', filename='cufflinks/histogram-binning')

**[3.1]** Добавьте третий признак ('BMI'). Создайте признак *Индекс массы тела* ([BMI](https://en.wikipedia.org/wiki/Body_mass_index)). Для этого воспользуйтесь коэффициентами перевода единиц с точностью до 5 знака после запятой.

*# Ваш код здесь*

df['BMI'] = (df['Weight']/(df['Weight']\*df['Height'])\*703).round(5)

**[3.2]** Постройте картинку, на которой будут отображены попарные зависимости признаков , 'Height', 'Weight' и 'BMI' друг от друга.

*# Ваш код здесь*

sns.set()

sns.pairplot(df[['Weight', 'Height', 'BMI']], size = 3.5)

plt.show();

**[4]**. Создайте в DataFrame *data* новый признак *weight\_category*, который будет иметь 3 значения: 1 – если вес меньше 120 фунтов. (~ 54 кг.), 3 - если вес больше или равен 150 фунтов (~68 кг.), 2 – в остальных случаях. Постройте boxplot, демонстрирующий зависимость роста от весовой категории. Подпишите ось *y* меткой «Рост», ось *x* – меткой «Весовая категория».

*# Ваш код здесь*

# Функция весовой категории

def wcat(weight):

if weight<120:

return 1

elif weight>=150:

return 2

else:

return 3

#Добавим в наш df весовую категорию

df['weight\_category'] = df\_weight.apply(wcat)

#Построим boxplot

trace0 = Box(

y=df['Height'],

x=df['weight\_category'],

name='noClassGc',

marker=Marker(

color='green'

)

)

data = Data([trace0])

layout = Layout( title='Boxplot Height/weight category',

yaxis=YAxis(

title='Height (inches)',

zeroline=False

),

xaxis = XAxis(

title = 'weight category'

),

boxmode='group'

)

fig = Figure(data=data, layout=layout)

iplot(fig, filename='box-grouped')

**[5].** Постройте scatter plot зависимости роста от веса. Подпишите картинку.

*# Ваш код здесь*

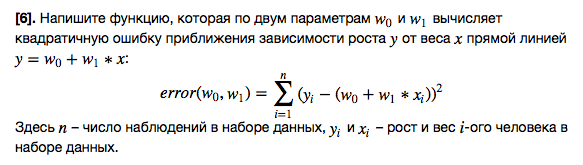
#scatter plot Height/Weight

var = 'Weight'

data = pd.concat([df['Height'], df[var]], axis=1)

data.plot.scatter(x=var, y='Height', title = 'Height dependence of the weight', color='purple')

**Минимизация квадратичной ошибки**



*# Ваш код здесь*

def func\_y(w0, w1, x):

return w0 + w1\*x

def func\_error(w0, w1):

return sum(map(lambda x, y: (y - func\_y(w0, w1, x)) \*\* 2, df.Weight, df.Height))

/Users/Kate/Desktop/Screen Shot 2017-04-21 at 19.02.48.png

*# Ваш код здесь*

def plot\_model(w0\_w1\_values):

plot\_scatter = df.plot(x ='Weight',y = 'Height', kind = 'scatter', color='purple', title='Height dependence of the weight')

plot\_scatter.set(xlabel = 'Weight', ylabel = 'Height')

plt.xlim(50, 200)

plt.ylim(50, 80)

x\_range = np.linspace(50, 200, 100)

for w0, w1 in w0\_w1\_values:

y = map(lambda p: func\_y(w0, w1, p), x\_range)

plt.plot(x\_range, y, lw = 2)

plot\_model([(55, 0.04), (50, 0.22)])

/Users/Kate/Desktop/Screen Shot 2017-04-21 at 19.02.54.png

*# Ваш код здесь*

w0 = 50

w1 = np.linspace(-1, 1, 100)

error = map(lambda p: func\_error(w0, p), w1)

# Слой с описанием

fig = plt.figure()

ax = fig.add\_subplot(111)

ax.set\_title('Error dependence of the w1')

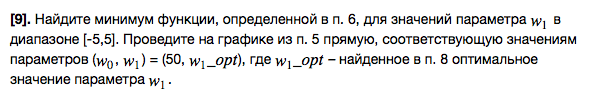
ax.set\_xlabel('w1')

ax.set\_ylabel('Error')

# Слой с нашими данными

plt.plot(w1, error)

plt.show()



*# Ваш код здесь*

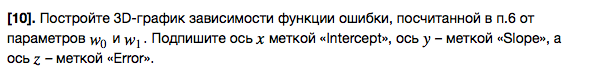
from scipy import optimize

w0 = 50

w1\_opt = optimize.minimize\_scalar(lambda x: func\_error(w0, x), bounds = [-5, 5]).x

print 'w1\_opt = %.4f' % w1\_opt

plot\_model([(w0, w1\_opt)])



*# Ваш код здесь*

from mpl\_toolkits.mplot3d import Axes3D

fig = plt.figure()

fig.set\_size\_inches(13.5, 7.5)

ax = fig.gca(projection='3d') # get current axis

# Создаем массивы NumPy с координатами точек по осям x и y.

# Используем метод meshgrid, при котором по векторам координат

# создается матрица координат. Задаем нужную функцию Z(x, y).

W0 = np.arange(-100, 101, 1)

W1 = np.arange(-5, 5, 0.25)

W0, W1 = np.meshgrid(W0, W1)

ERR = func\_error(W0, W1)

# # Наконец, используем метод \*plot\_surface\* объекта

# # типа Axes3DSubplot. Также подписываем оси.

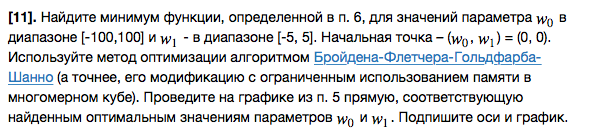
surf = ax.plot\_surface(W0, W1, ERR)

ax.set\_xlabel('Intercept')

ax.set\_ylabel('Slope')

ax.set\_zlabel('Error')

plt.show()



*# Ваш код здесь*

def func\_error\_opt(w):

return func\_error(w[0], w[1])

# Оптимизация функции

result = optimize.minimize(func\_error\_opt, (0,0), bounds = [(-100, 100), (-5, 5)], method = 'L-BFGS-B')

#Присвоим w0\_opt, w1\_opt значения из массива result.x

w0\_opt, w1\_opt = result.x

optimize.minimize(func\_error\_opt, (0,0), bounds = [(-100, 100), (-5, 5)], method = 'L-BFGS-B')

print 'w0\_opt = %.4f, w1\_opt = %.4f' % (w0\_opt, w1\_opt)

plot\_model([(w0\_opt, w1\_opt)])