Московский Авиационный Институт (Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №0 по курсу “Машинное обучение”

Студент: А. Т. Бахарев

Преподаватель: Самир Ахмед

Группа: М8О-306Б

Дата: 27.04.2019

Оценка:

Подпись:

# Лабораторная работа №0

**Тема:** Получение и обработка данных

**Задача:** Требуется сформировать/получить два набора данных соответствующие следующим критериям:

1) Один из датасетов должен представлять собой корпус документов. Язык, источник и тематика произвольна

2) Второй датасет должен содержать категориальные, количественные признаки. Для данного датасета определить предсказываемые признаки (для задачи регрессии и классификации). Если такого признака нет, спроектировать

По каждому датасету построить распределения признаков (в случае корпуса документов – построить распределение слов) и объяснить имеющуюся картину. Вычислить статистические характеристики признаков. Обнаружить и решить возможные проблемы с данными. Если решить данную проблему невозможно, объяснить почему.

**Датасеты:**

1. Оценки школьников по разным предметам
2. Положительные и негативные отзывы о фильмах

# Описание решения проблемы

Данная работа делится на 2 этапа – первый это построение распределений для первого датасета, в котором содержатся категориальные признаки. Второй – построение распределений слов в корпусе документов для наблюдения за частотой встречающихся слов.

Для первого этапа работы я выбрал следующие признаки: Математическое ожидание, Дисперсия, мода, максимальное значение, минимальное значение, среднее значение, размах вариации.

Для работы с документами никаких характеристик строить не нужно: достаточно изобразить частоту каждого слова, встречающегося в данных документах.

Исходный код. Часть 1

#! usr/bin/python

**import** re

**import** plotly.offline **as** py

**import** plotly.graph\_objs **as** go

**import** plotly.io **as** pio

**import** os

**def** AddToDict(d, val):# function froь adding value in dictionary

**if** val == None:

**return**

**if** val **in** d:

d[val] += 1

**else**:

d[val] = 1

**def** AddToMarkList(mark\_list, match):# this function will add marks into list.

group = match.group(1)

mark = match.group(2)

mark\_list[**ord**(group) - **ord**("A")].append(**int**(mark))

**def** DrawDistribution(dist, name):# Visualization of simple Distribution. See html file in work dirrectory

labels = **list**(dist.keys())

values = **list**(dist.values())

trace = go.Pie(labels = labels, values = values)

py.plot([trace], filename = name + ".html")

**def** ReadFile(csv\_file):# read whole file for getting information

count = 0

GroupDist = {}

EduLevelDist = {}

MathMarksList = [[], [], [], [], []]

WritingMarkList = [[], [], [], [], []]

**for** line **in** csv\_file:

**if** count == 0:

count += 1

**continue**

AddToDict(GroupDist, re.match("^[^,]\*,([^,]\*)", line).group(1))

AddToDict(EduLevelDist, re.match("^[^,]\*,[^,]\*,([^,]\*)", line).group(1))

AddToMarkList(MathMarksList, re.match("^[^,]\*,\"[a-z]\*\s([A-Z])\*\",.\*,\"([^,]\*)\",[^,]\*,[^,]\*$", line))

AddToMarkList(WritingMarkList, re.match("^[^,]\*,\"[a-z]\*\s([A-Z])\*\",.\*,\"([^,]\*)\"$", line))

**return** GroupDist, EduLevelDist, MathMarksList, WritingMarkList

**def** GetMarkStatistics(mark\_list):# mark\_list is a list with marks

**def** GetExpectation(mark\_list):#0

**sum** = 0

**for** n **in** mark\_list:

**sum** += **int**(n)

**return** **sum** / **len**(mark\_list)

**def** GetDispersion(mark\_list, E):#1

square\_vals = []

**for** n **in** mark\_list:

square\_vals.append((n - **int**(E)) \*\* 2)

**return** GetExpectation(square\_vals)

**def** GetMedian(mark\_list):#2

**return** mark\_list[**int**(**len**(mark\_list) / 2)]

**def** GetMode(mark\_list):#3

mark\_freq = {}

max\_freq = 0

max\_mark = 0

**for** n **in** mark\_list:

AddToDict(mark\_freq, n)

**if** mark\_freq[n] > max\_freq:

max\_freq = mark\_freq[n]

max\_mark = n

**return** max\_mark

**def** GetMin(mark\_list):

**return** mark\_list[0]

**def** GetMax(mark\_list):

**return** mark\_list[**len**(mark\_list) - 1]

**def** GetRange(mark\_list):

**return** GetMax(mark\_list) - GetMin(mark\_list)

mark\_list.sort()

Stats = []

E = GetExpectation(mark\_list)

Stats.append(E) #0

Stats.append(GetDispersion(mark\_list, E)) #1

Stats.append(GetMedian(mark\_list)) #2

Stats.append(GetMode(mark\_list)) #3

Stats.append(GetMax(mark\_list)) #4

Stats.append(GetMin(mark\_list)) #5

Stats.append(GetRange(mark\_list)) #6

**return** Stats

**def** DrawMarkStatistics(groups\_mark\_list, name): # groups\_mark\_list is a list, which contains list with marks

groups = ['Group A', 'Group B', 'Group C', 'Group D', 'Group E']

**if** **not** os.path.isdir("./" + name):

os.mkdir("./" + name)

os.chdir("./" + name)

**def** DrawExpectation(res\_list):

val = []

**for** gr **in** res\_list: # each group in result list

val.append(gr[0]) # 0 - position of expectation in list

data = [go.Bar(x = groups, y = val)]

layout = go.Layout(title = name + ' expectation among all groups')

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

py.plot(fig, filename = name + "Expectation.html")

**def** DrawMode(res\_list):

val = []

**for** gr **in** res\_list: # each group in result list

val.append(gr[3]) # 3 - position of mode in list

data = [go.Bar(x = groups, y = val)]

layout = go.Layout(title = name + ' mode among all groups')

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

py.plot(fig, filename = name + "Mode.html")

**def** DrawMinMaxRange(res\_list):

min\_val = []

max\_val = []

range\_val = []

median\_val = []

**for** gr **in** res\_list:

min\_val.append(gr[5])

max\_val.append(gr[4])

range\_val.append(gr[6])

median\_val.append(gr[2])

trace1 = go.Bar(x = groups, y = min\_val, name='Min')

trace2 = go.Bar(x = groups, y = max\_val, name = 'Max')

trace3 = go.Bar(x = groups, y = range\_val, name = 'Range')

trace4 = go.Bar(x = groups, y = median\_val, name = 'Median')

data = [trace1, trace2, trace3, trace4]

layout = go.Layout(barmode='group')

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

py.plot(fig, filename = name + 'MinMaxRangeMedian.html')

res\_list = []

**for** m **in** groups\_mark\_list:

res\_list.append(GetMarkStatistics(m))

DrawExpectation(res\_list)

DrawMode(res\_list)

DrawMinMaxRange(res\_list)

os.chdir("./..")

**def** main():

csv\_file = **open**("StudentsPerformance.csv", "r")

GroupDist, EducationDist, MathMarks, WritingMarks = ReadFile(csv\_file)

DrawDistribution(GroupDist, "Groups")

DrawDistribution(EducationDist, "Education")

DrawMarkStatistics(MathMarks, "MathMarks")

DrawMarkStatistics(WritingMarks, "WritingMarks")

**if** \_\_name\_\_ == "\_\_main\_\_":

main()

Исходный код. Часть 2

#! usr/bin/python3

**import** os

**import** plotly.offline **as** py

**import** plotly.graph\_objs **as** go

**import** plotly.io **as** pio

separators = [",", ";", ".", ":", "\"", "\'", "`", "\\", "/", ">", "<", "=", "-", "\_", "@", "#", "(", ")", "\t", "", " "]

**def** AddToDict(d, val):# function fror adding value in dictionary

**if** val == None:

**return**

**if** val **in** d:

d[val] += 1

**else**:

d[val] = 1

**def** ProcessFile(path, word\_freq):# read file and collecting words

**file** = **open**(path, "r", encoding = "latin-1")

**for** line **in** **file**:

words = line.split()

**for** word **in** words:

word = word.strip(**str**(separators))

**if** word **not** **in** separators **and** **not** word.isnumeric():

AddToDict(word\_freq, word)

**file**.close()

**def** DrawWordsDistribution(word\_freq, name):# illustration of distribution

words = []

freq = []

**for** w **in** word\_freq.keys():

**if** word\_freq[w] < 5:# if word frequency less than 5, we don't use them

**continue** # this feature is optional, for more faster computation

**else**:

freq.append(word\_freq[w])

words.append(w)

data = [go.Bar(x = words, y = freq)]

layout = go.Layout(title = name + ' Words Distribution')

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

py.plot(fig, filename = name + ".html")

**def** MakeAnalysis(folder, analysis\_name):# wrapper function for more convinient work

word\_freq = {}

files = os.listdir(folder)

**for** **file** **in** files:

ProcessFile(folder + **file**, word\_freq)

DrawWordsDistribution(word\_freq, analysis\_name)

**def** main():

MakeAnalysis("./neg/", "Negative Reviwes")

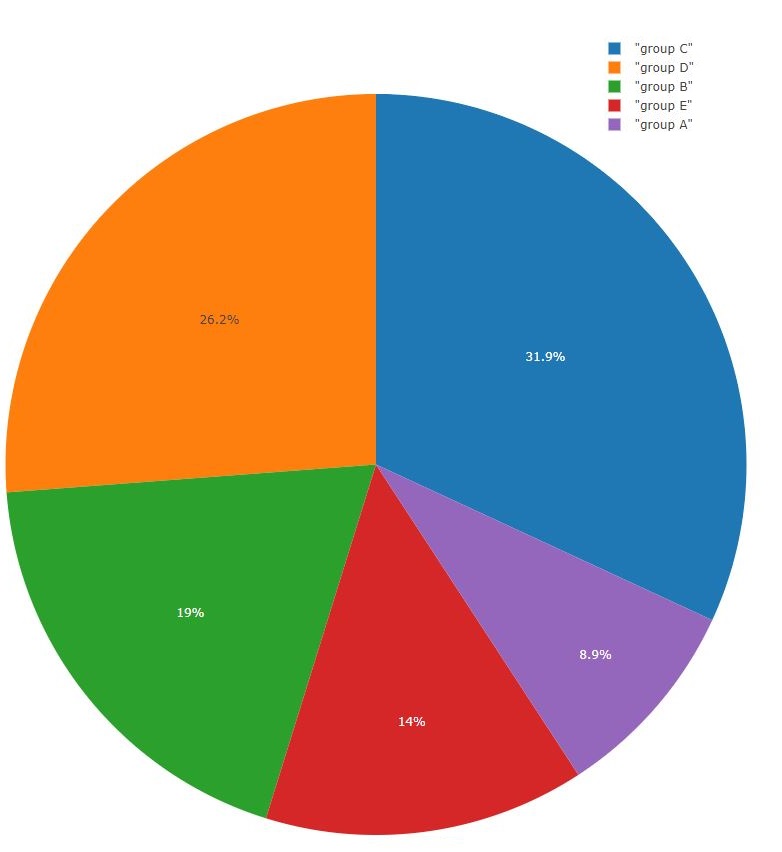
MakeAnalysis("./pos/", "Postive Reviwes")

**print**("OK")

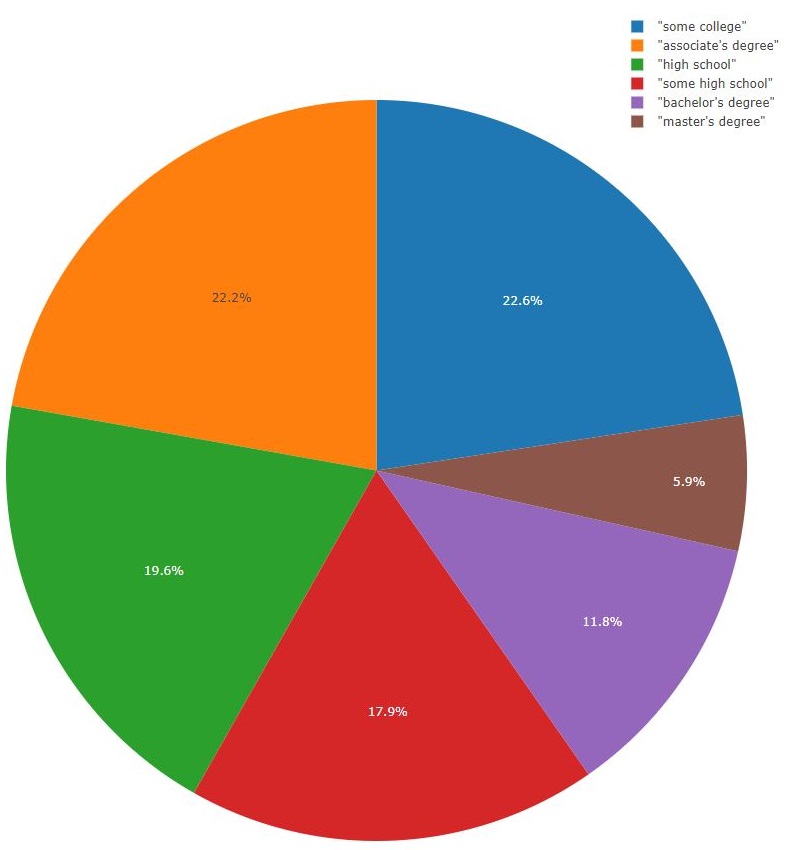
**if** \_\_name\_\_ == "\_\_main\_\_":

main()

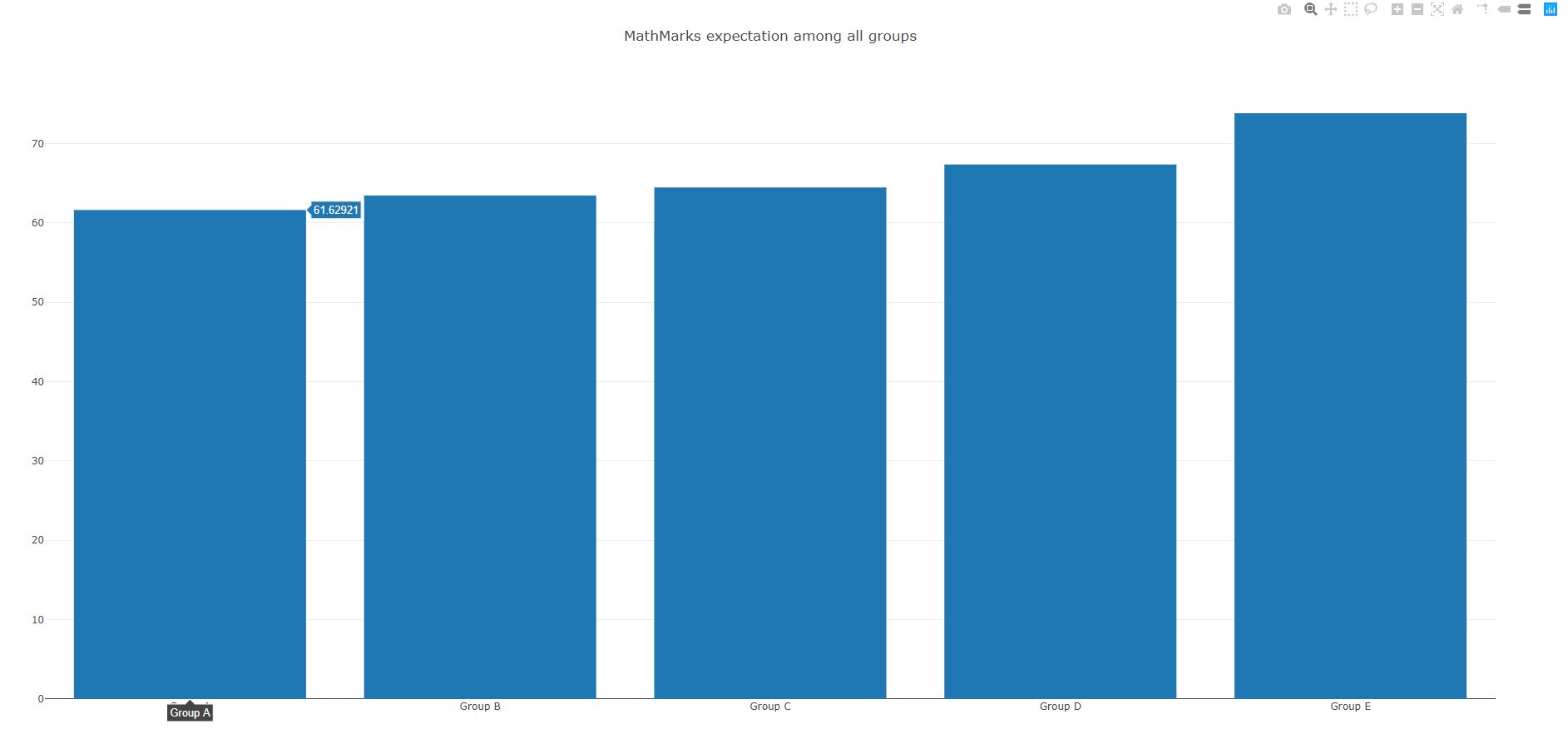
Распределения. Часть 1



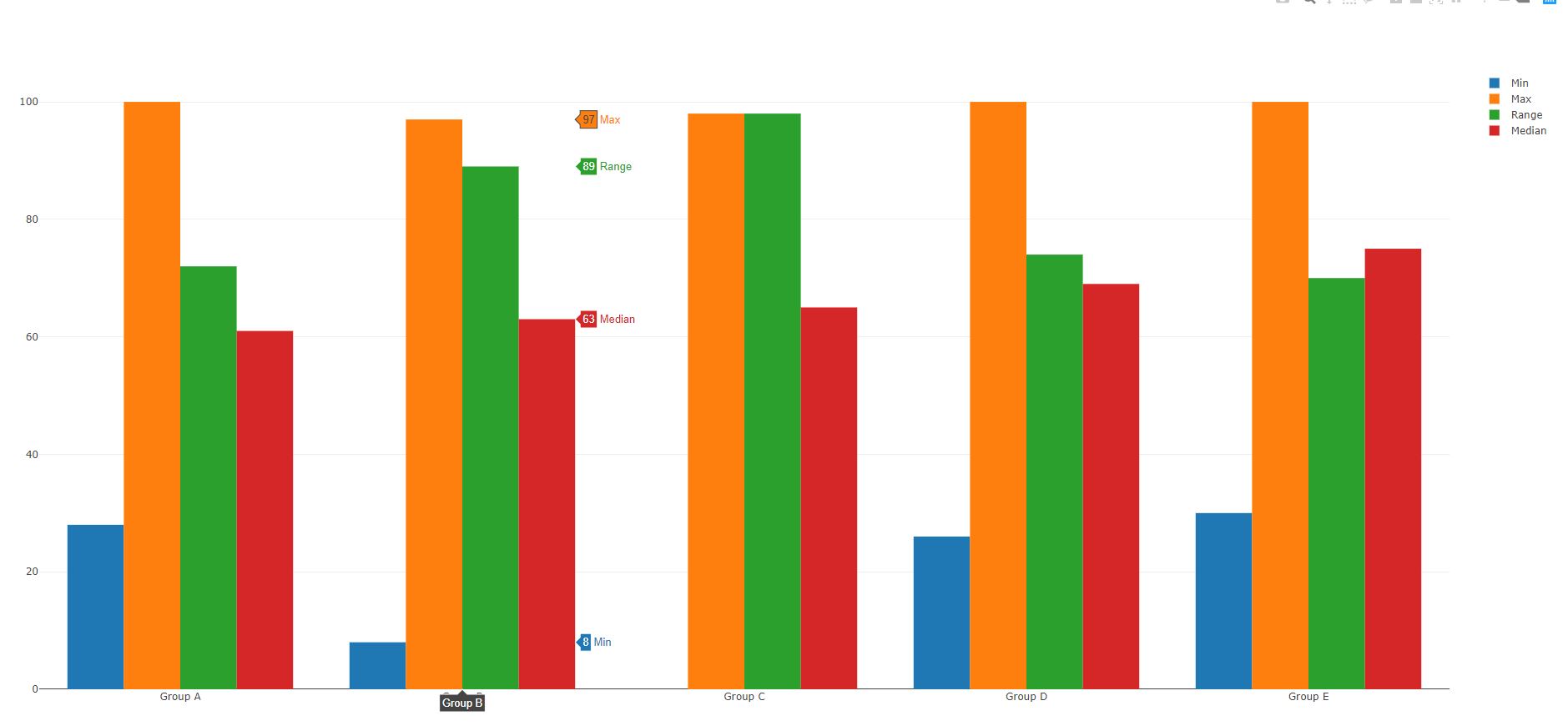
Распределение учеников по группам



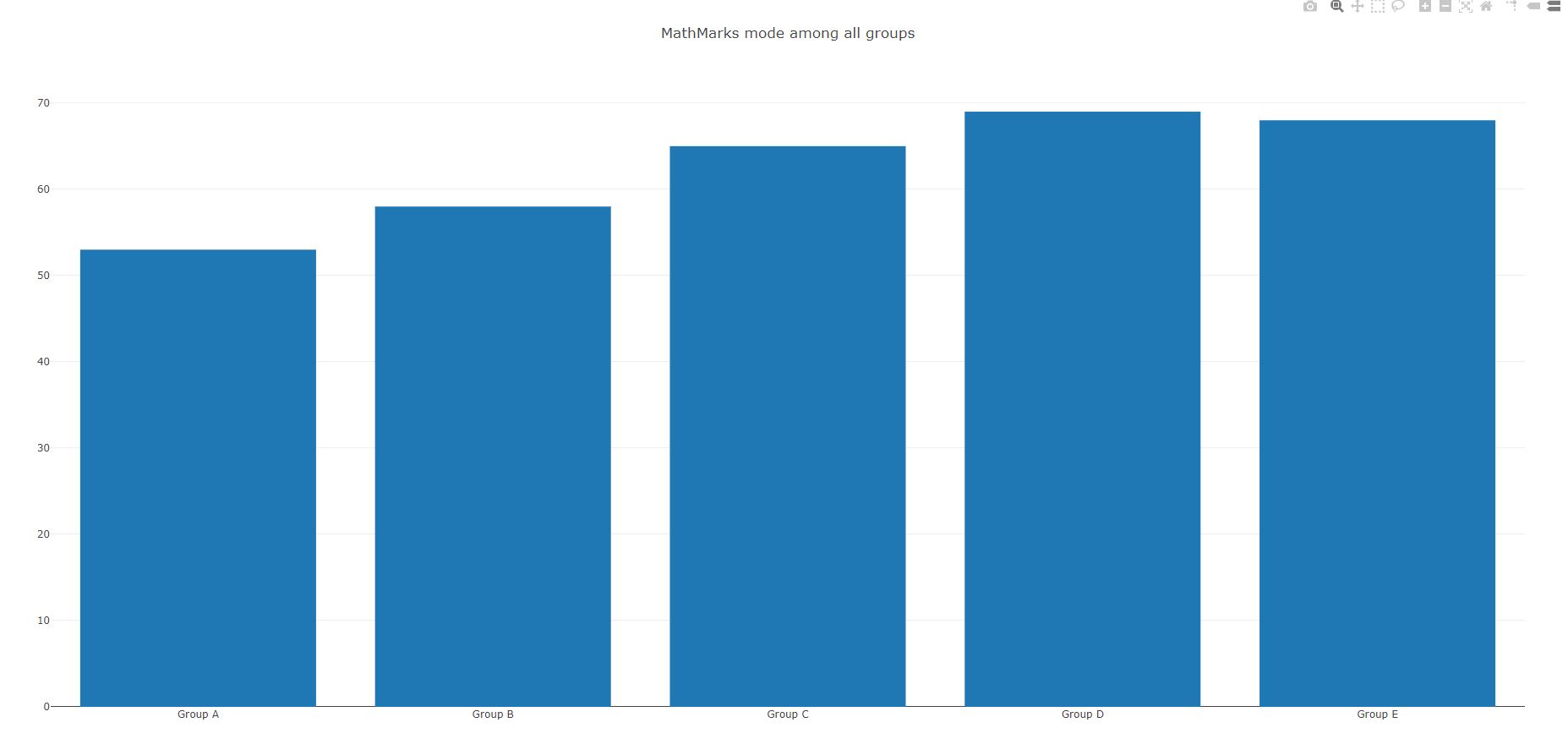
Распределение учеников по ученой степени



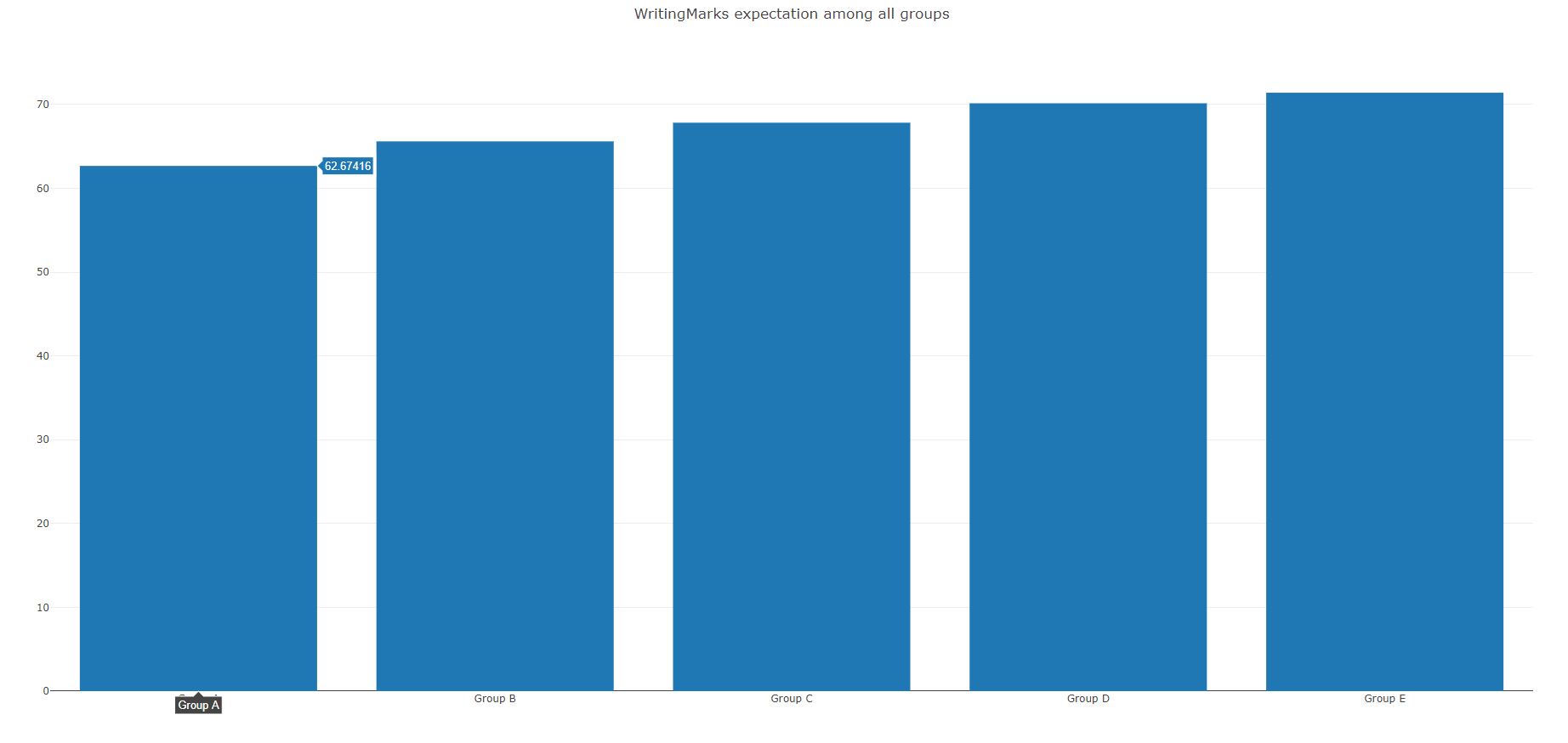
Математическое ожидание оценок по математике среди всех групп



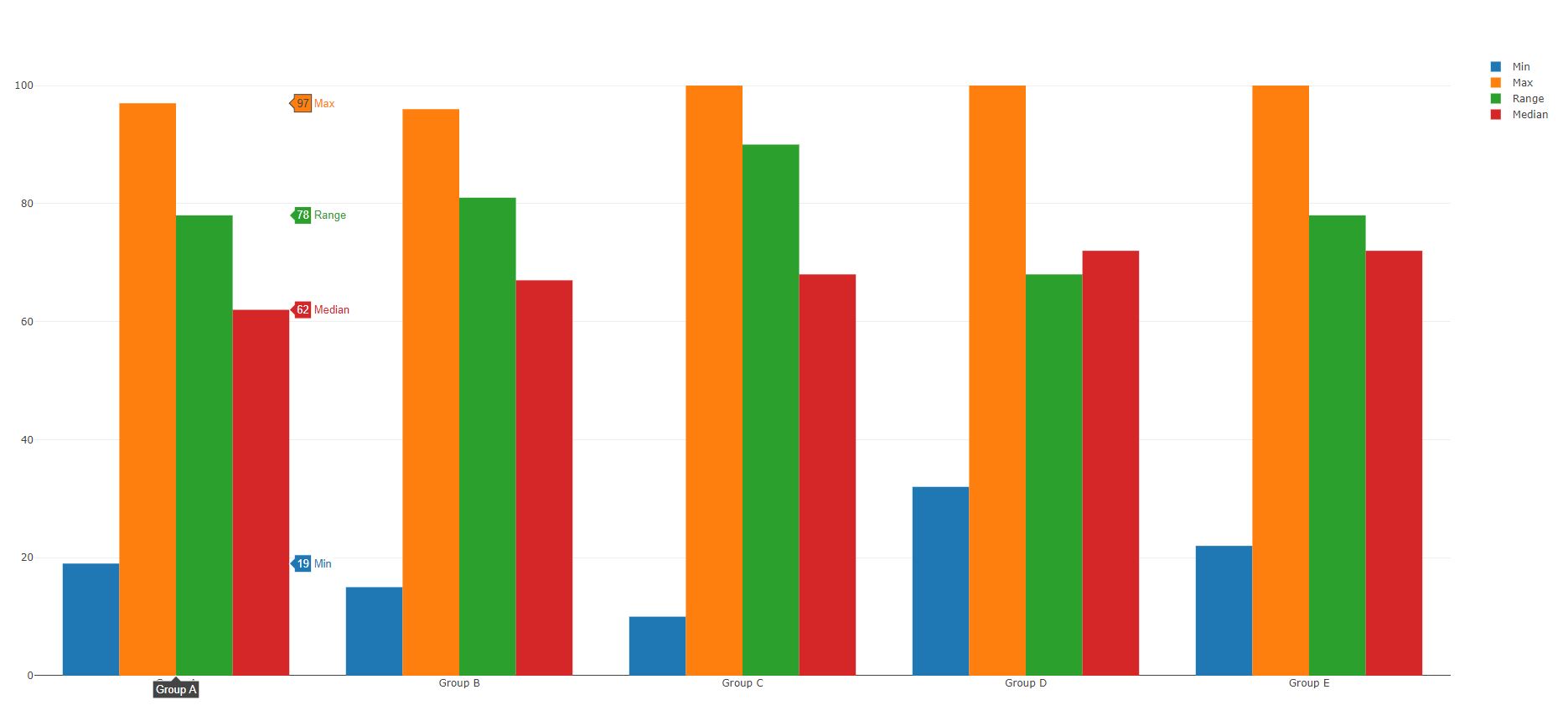
Минимальное, максимальное, средние значения, а также медиана оценок по математике среди всех групп



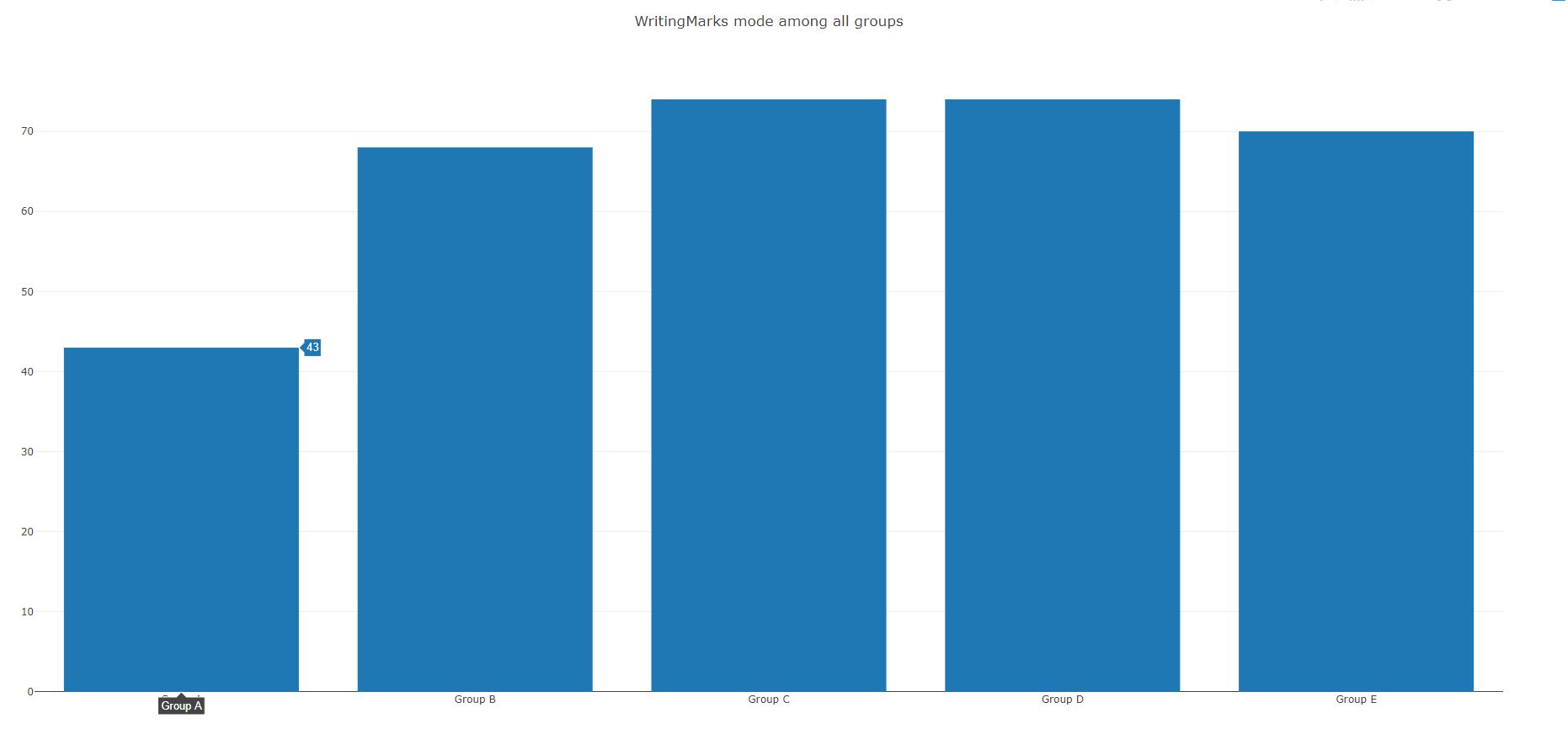
Мода оценок по математике среди всех групп



Математическое ожидание оценок по письму среди всех групп

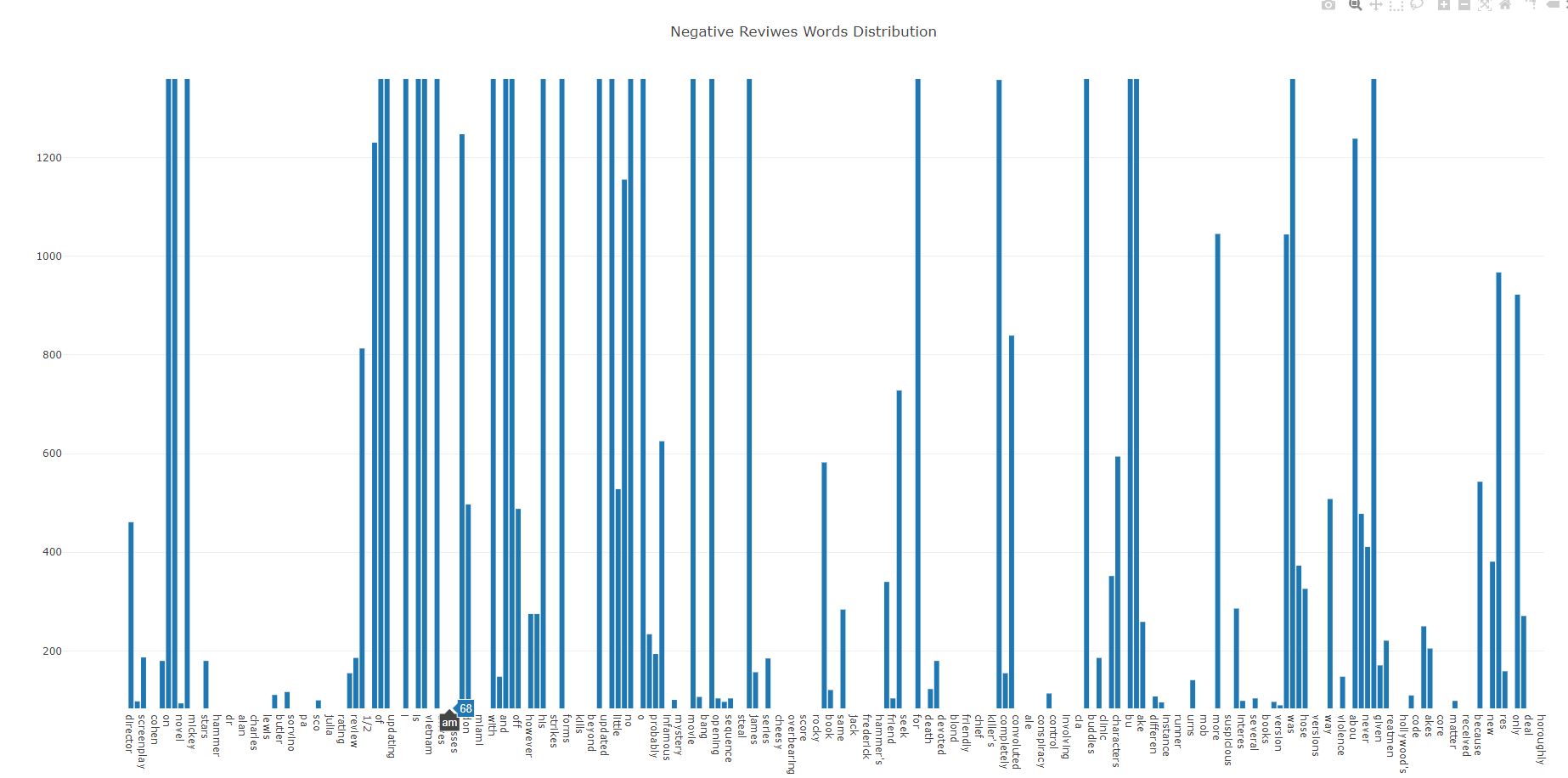


Минимальное, максимальное, средние значения, а также медиана оценок по письму среди всех групп

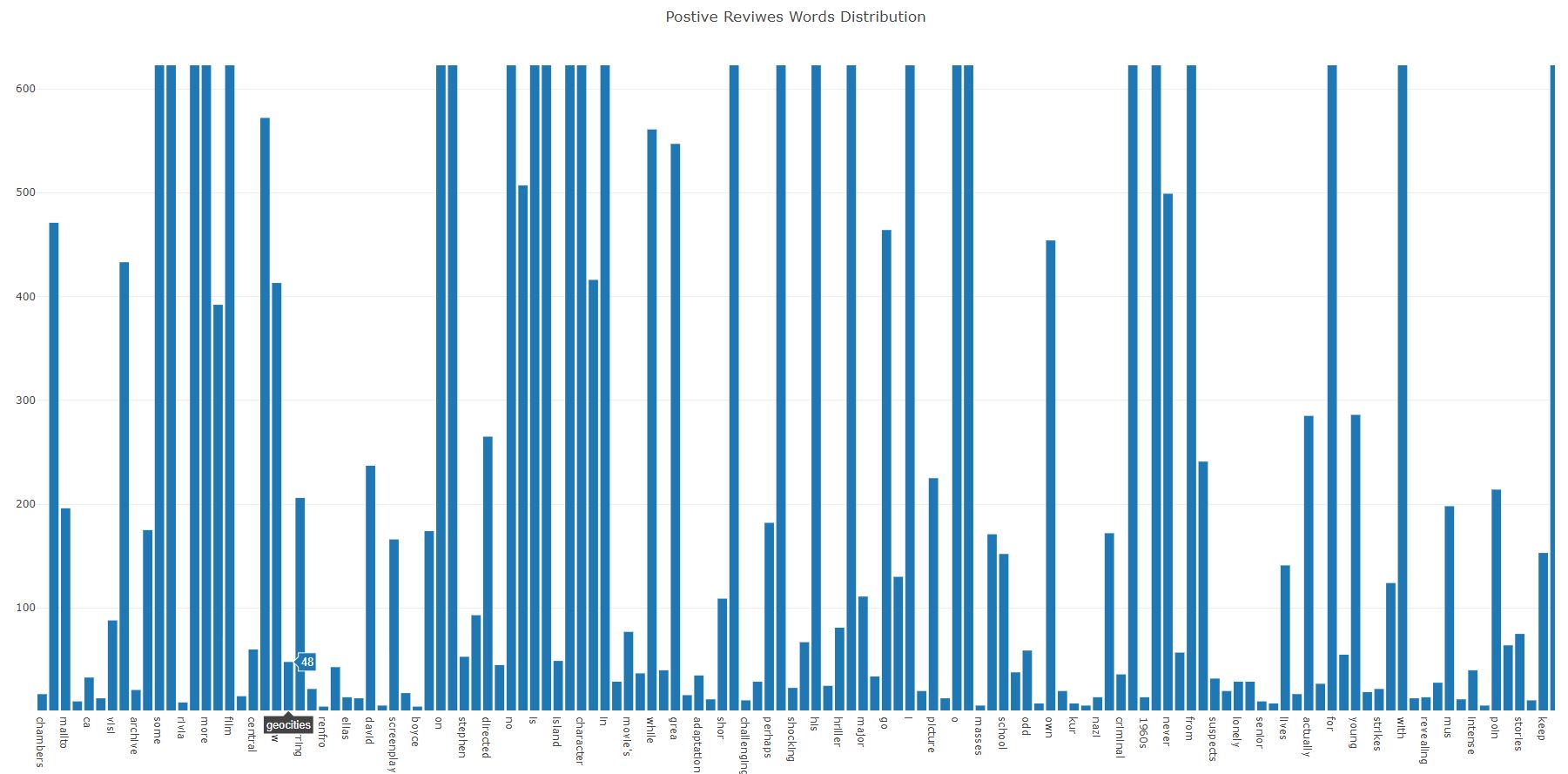


Мода оценок по письму среди всех групп

Распределения. Часть 2



Фрагмент распределения слов в отрицательных отзывах к фильмам



Фрагмент распределения слов в положительных отзывах к фильмам

Из распределений слов в документах видно, что наиболее часто встречаемые слова – это предлоги, числительные, местоимения и междометия. Это логично, так как большинство предложений строится из этих синтаксических частей. Так же, не редко встречаются самые популярные слова в английском языке (cat, house, girl, boy). Еще отмечу наличие положительных и отрицательных слов в распределениях слов.

Выводы

Данная лабораторная работа познакомила меня с принципами и приемами сбора и анализа данных, а также их визуализации. Я повторил основные статистические распределения и характеристики данных.

Я считаю, что получил хорошие начальные знания для дальнейшего изучения машинного обучения и искусственного интеллекта. Надеюсь, что эти знания мне пригодятся в дальнейшем.