Государственное бюджетное общеобразовательное учреждение города Москвы «Школа №1359 им. Авиаконструктора М.Л. Миля»

**Внедрение компьютерных методов для анализа больших данных в процесс обучения в инженерных классах и в группах дополнительного образования**

Выполнила:

Шиншинова Ирина Олеговна,

Преподаватель информатики

Москва, 2024

Оглавление

[Введение 2](#_Toc158587178)

[Анализ текущей ситуации 5](#_Toc158587179)

[Изучение основ Python и базового функционала 6](#_Toc158587180)

[Теория вероятностей и математическая статистика 13](#_Toc158587181)

[Изучение продвинутых библиотек для анализа данных 15](#_Toc158587182)

[Анализ предложенного плана 20](#_Toc158587183)

[Выводы 21](#_Toc158587184)

[Список используемой литературы 22](#_Toc158587185)

# Введение

Анализ данных - это одна из наиболее перспективных и быстро развивающихся областей в современном мире. Она включает в себя сбор, обработку, анализ и интерпретацию данных для получения ценной информации, которая может быть использована в различных сферах жизни. В связи с этим заблаговременное знакомство школьников с анализом данных может быть очень полезным для их будущего.

Во-первых, анализ данных помогает принимать обоснованные решения в различных областях. Например, в экономике данные могут быть использованы для прогнозирования тенденций и принятия решений о том, как лучше управлять ресурсами. В медицине анализ данных может помочь в диагностике и лечении заболеваний. В маркетинге данные могут использоваться для определения наиболее эффективных методов привлечения и удержания клиентов.

Во-вторых, анализ данных способствует развитию критического мышления. Школьники, которые знакомы с анализом данных, будут способны критически оценивать информацию и понимать, как она может быть использована. Это поможет им принимать более взвешенные решения в своей жизни и карьере.

В-третьих, знакомство с анализом данных поможет школьникам развить навыки, которые будут полезны в любой профессии. Например, умение работать с большими объемами информации, анализировать и интерпретировать данные, а также создавать наглядные отчеты. Эти навыки будут востребованы в любой сфере деятельности, от бизнеса до науки и искусства.

Наиболее популярным и широко используемым инструментом для анализа данных на текущий момент является язык программирования Python. Это обусловлено рядом причин:

1. Простота использования: Python обладает простым и понятным синтаксисом, что делает его доступным для широкого круга пользователей, включая начинающих программистов и ученых.
2. Богатая стандартная библиотека: В Python имеется множество готовых функций и модулей для работы с данными, что позволяет сэкономить время на разработке собственного кода.
3. Большое количество библиотек и фреймворков: Для Python существует огромное количество специализированных библиотек, позволяющих решать различные задачи анализа данных, такие как машинное обучение, визуализация, обработка естественного языка и многие другие.
4. Открытый исходный код: Python является бесплатным и открытым программным обеспечением, что позволяет пользователям свободно использовать, изучать и модифицировать его исходный код.
5. Высокая производительность: Несмотря на свою простоту и понятность, Python является достаточно эффективным языком программирования, который может обрабатывать большие объемы данных и выполнять сложные вычисления.
6. Широкое распространение: Python используется во многих отраслях, включая науку, финансы, IT и другие, что обеспечивает его постоянное развитие и поддержку со стороны сообщества.

На текущий момент в большом количестве школьных заведения язык программирования Python является основным для обучения программированию на уроках информатики. Этот язык приоритетно предлагается для первого знакомства школьников с программированием в целом, активно используется для сдачи ОГЭ и ЕГЭ. Таким образом, текущий учебный процесс в образовательных учреждениях уже предполагает изучение Python как основного языка для большинства школьников.

Из вышеперечисленного следует, что внедрение тем, связанных с анализом данных в учебный процесс является актуальным, а также возможным в силу текущего спектра изучаемых тем школьной программы. В текущей работе будет предложен план подготовки учеников инженерных классов по этому направлению и предложен спектр необходимых для изучения тем.

**Цель:**

Разработать и внедрить компьютерные методы анализа больших данных для улучшения процесса обучения в инженерных классах и группах дополнительного образования, с целью повышения качества знаний учащихся и их мотивации к изучению технических дисциплин.

**Задачи:**

1. Разработать набор образовательных модулей по программированию, необходимых для изучения учениками направление «анализ данных» с указанием количества академических часов и примерами заданий.
2. Разработать набор образовательных модулей по математике, необходимых для изучения учениками направление «анализ данных» с указанием количества академических часов и примерами заданий.
3. Оценить возможность внедрения предложенного учебного плана в текущий школьный курс, оценить возможные сложности.

# Анализ текущей ситуации

Сейчас на уроках школьной информатики основной упор делается на изучение следующих тем:

1. Основы работы с компьютером и операционной системой.
2. Работа с текстовыми редакторами и электронными таблицами.
3. Создание презентаций.
4. Основы программирования.
5. Работа с интернетом и поиск информации.
6. Безопасность в интернете.
7. Основы цифровой грамотности и защиты персональных данных.

Текущий список является сжатым и отражает общую картину подготовки по предмету «информатика» в общеобразовательных учреждениях. Как можно заметить, основной спектр тем направлен на увеличение компьютерной грамотности, а не на формирование у школьников навыков для практической работы с областями, где информатика и программирование используется активно в наши дни. Исходя из того, что большая часть навыков, необходимая для изучения анализа данных, уже есть у школьников, внедрение этого направления является не только актуальным, но и возможным.

Сформулируем и опишем, какие навыки должны получить школьники для дальнейшего изучения текущей области:

1. Изучение основ Python и базового функционала
2. Изучение основ теории вероятностей и математической статистики.
3. Изучение продвинутых библиотек для анализа данных: Pandas и др.

Перейдем к описанию соответствующих модулей.

# Изучение основ Python и базового функционала

Сформулируем основные темы, которые должен освоить ученик, и подкрепим их примерами конкретных заданий и их решениями для формирования более точного понимания каждой темы:

1. Ввод-вывод данных
2. Условия
3. Вычисления
4. Цикл for
5. Цикл while
6. Строки
7. Списки
8. Функции
9. Двумерные списки
10. Множества
11. Словари

**Ввод-вывод данных**

*Предполагаемое время на разбор: 2 академических часа*

Тема “ввод-вывод данных” является одной из ключевых в программировании на Python. В этой теме рассматриваются различные способы получения данных от пользователя, а также сохранения и отображения результатов работы программы.

Пример практического задания:

Напишите программу, которая считывает три числа и выводит их сумму. Каждое число записано в отдельной строке.

**Ввод:**

2

3

6

**Вывод:**

11

Возможное решение:

a = int(input())  
b = int(input())  
c = int(input())  
s = a + b + c  
print(s)

**Условия**

*Предполагаемое время на разбор: 2 академических часа*

Условная конструкция if в Python используется для выполнения определенного блока кода в зависимости от выполнения заданного условия.

Пример практического задания:

Даны три целых числа. Определите, сколько среди них совпадающих. Программа должна вывести одно из чисел: 3 (если все совпадают), 2 (если два совпадает) или 0 (если все числа различны).

**Ввод:**

10

5

10

**Вывод:**

2

Возможное решение:

a = int(input())  
b = int(input())  
c = int(input())  
if a == b == c:  
 print(3)  
elif a == b or a == c or b == c:  
 print(2)  
else:  
 print(0)

**Вычисления**

*Предполагаемое время на разбор: 2 академических часа*

Встроенные и дополнительные арифметические операции являются одной из причин применения языков программирования, в том числе для анализа данных. Благодаря программированию мы получаем возможность производить вычисления быстро и эффективно.

Пример практического задания:

Дано два числа a и b. Найдите гипотенузу треугольника с заданными катетами. В двух строках вводятся два числа, нужно вывести ответ на задачу.

**Ввод:**

3

4

**Вывод:**

5.0

Возможное решение:

from math import sqrt  
a = float(input())  
b = float(input())  
print(sqrt(a \* a + b \* b))

**Цикл For**

*Предполагаемое время на разбор: 3 академических часа*

Цикл for в Python используется для перебора элементов в коллекции, такой как список, кортеж, строка или диапазон чисел.

Пример практического задания:

Даны два целых числа A и B (при этом A ≤ B). Выведите все числа от A до B включительно.

**Ввод:**

1

10

**Вывод:**

1 2 3 4 5 6 7 8 9 10

Возможное решение:

A = int(input())  
B = int(input())  
for x in range(A, B + 1):  
 print(x, end = ' ')

**Цикл While**

*Предполагаемое время на разбор: 3 академических часа*

Цикл while в Python используется, когда нужно выполнить блок кода несколько раз, пока выполняется определенное условие.

Пример практического задания:

По данному целому числу N распечатайте все квадраты натуральных чисел, не превосходящие N, в порядке возрастания.

**Ввод:**

50

**Вывод:**

1 4 9 16 25 36 49

Возможное решение:

N = int(input())  
i = 1   
while i \* i <= N:  
 print(i \* i, end = ' ')  
 i += 1

**Строки**

*Предполагаемое время на разбор: 4 академических часа*

Тема “Строки в Python” включает в себя изучение основных операций со строками, таких как создание, изменение, сравнение и обработка строк.

Пример практического задания:

Дана строка. Разрежьте ее на две равные части (если длина строки — четная, а если длина строки нечетная, то длина первой части должна быть на один символ больше). Переставьте эти две части местами, результат запишите в новую строку и выведите на экран.

**Ввод:**

Hi

**Вывод:**

iH

Возможное решение:

s = input()  
n = len(s)  
ind = (n + 1) // 2  
s1 = s[:ind]  
s2 = s[ind:]  
print(s2 + s1)

**Списки**

*Предполагаемое время на разбор: 4 академических часа*

Тема “Списки в Python” посвящена изучению основных операций со списками, таких как создание списка, добавление и удаление элементов, сортировка и поиск элементов. Списки в Python представляют собой упорядоченные коллекции элементов одного типа.

Пример практического задания:

Дан список чисел. Выведите все элементы списка, которые больше предыдущего элемента.

**Ввод:**

1 5 2 4 3

**Вывод:**

5 4

Возможное решение:

a = [int(x) for x in input().split()]  
  
for i in range(1, len(a)):  
 if a[i] > a[i - 1]:  
 print(a[i], end = ' ')

**Функции**

*Предполагаемое время на разбор: 2 академических часа*

Тема “Функции в Python” охватывает изучение принципов работы с функциями, их создание, вызов и возврат значений. Функции в Python позволяют выполнять определенные операции над данными и возвращать результат этих операций. Функции могут принимать аргументы и выполнять различные операции в зависимости от этих аргументов.

Пример практического задания:

Даны четыре действительных числа. Напишите функцию distance(x1, y1, x2, y2), вычисляющая расстояние между двумя точками в прямоугольной системе координат. Считайте четыре действительных числа и выведите результат работы этой функции.

**Ввод:**

0

0

1

1

**Вывод:**

1.4142135623730951

Возможное решение:

from math import sqrt  
def distance(x1, y1, x2, y2):  
 dx = x2 - x1  
 dy = y2 - y1  
 return sqrt(dx \* dx + dy \* dy)  
  
x1 = float(input())  
y1 = float(input())  
x2 = float(input())  
y2 = float(input())  
print(distance(x1, y1, x2, y2))

**Двумерные списки**

*Предполагаемое время на разбор: 4 академических часа*

Двумерные списки, также известные как матрицы, представляют собой список элементов, каждый из которых также является списком. Они используются для представления табличных данных или матриц.

Пример практического задания:

Найдите индексы первого вхождения максимального элемента. Выведите два числа: номер строки и номер столбца, в которых стоит наибольший элемент в двумерном массиве. Если таких элементов несколько, то выводится тот, у которого меньше номер строки, а если номера строк равны то тот, у которого меньше номер столбца.

**Ввод:**

3 4

0 3 2 4

2 3 5 5

5 1 2 3

**Вывод:**

1 2

Возможное решение:

n, m = [int(i) for i in input().split()]  
a = [[int(j) for j in input().split()] for i in range(n)]  
best\_i, best\_j = 0, 0  
curr\_max = a[0][0]  
for i in range(n):  
 for j in range(m):  
 if a[i][j] > curr\_max:  
 curr\_max = a[i][j]  
 best\_i, best\_j = i, j  
print(best\_i, best\_j)

**Множества**

*Предполагаемое время на разбор: 4 академических часа*

Тема “Множества (set) в Python” рассматривает основные операции над множествами, такие как создание, проверка на принадлежность, пересечение, объединение и разность множеств.

Пример практического задания:

Во входной строке записана последовательность чисел через пробел. Для каждого числа выведите слово YES (в отдельной строке), если это число ранее встречалось в последовательности или NO, если не встречалось.

**Ввод:**

1 2 3 2 3 4

**Sample Output:**

NO

NO

NO

YES

YES

NO

Возможное решение:

numbers = [int(s) for s in input().split()]  
occur\_before = set()  
for num in numbers:  
 if num in occur\_before:  
 print('YES')  
 else:  
 print('NO')  
 occur\_before.add(num)

**Словари**

*Предполагаемое время на разбор: 4 академических часа*

Тема “Словари (dict) в Python” изучает основы работы со словарями, их создание и изменение, а также доступ к значениям по ключу.

Словари в Python используются для хранения пар ключ-значение. Они обеспечивают быстрый доступ к данным по ключу, в отличие от списков, где доступ осуществляется по индексу.

Пример практического задания:

В единственной строке записан текст. Для каждого слова из данного текста подсчитайте, сколько раз оно встречалось в этом тексте ранее.

Словом считается последовательность непробельных символов идущих подряд, слова разделены одним или большим числом пробелов или символами конца строки.

**Ввод:**

one two one tho three

**Вывод:**

0 0 1 0 0

Возможное решение:

freq = {}  
words = input().split()  
for w in words:  
 freq[w] = freq.get(w, 0) + 1  
 print(freq[w] - 1, end = ' ')

**Вывод:**

За **34 академических часа** есть возможность освоить весь спектр основных тем на языке программирования Python. Важно, что текущие темы можно изучать и для других языков программирования, так как это фундаментальная основа, которая присутствует в большинстве актуальных языков программирования.

# Теория вероятностей и математическая статистика

Для развития в сфере анализа данных важно знать математическую статистику и теорию вероятностей по следующим причинам:

1. Основы вероятности и статистики лежат в основе анализа данных. Без понимания основных понятий и методов невозможно эффективно анализировать и интерпретировать данные.
2. Математическая статистика предоставляет инструменты для обработки и интерпретации данных, позволяет делать выводы о характеристиках генеральной совокупности на основе выборочных данных.
3. Теория вероятностей помогает понять и оценить риски, связанные с принятием решений на основе анализа данных, а также разрабатывать и оценивать статистические модели и методы.
4. Знание математической статистики и теории вероятностей позволяет аналитикам данных лучше понимать ограничения и предположения, лежащие в основе используемых методов, а также возможные альтернативные подходы к решению задач.
5. В целом, глубокое понимание этих областей помогает аналитикам данных разрабатывать более эффективные и обоснованные решения на основе данных, что является ключевым навыком в отрасли.

В некоторых образовательных учреждениях уже внедрены отдельные уроки по теории вероятностей и статистике, что положительно будет отражаться на прогрессе учеников по этому направлению.

Для корректного анализа данных школьникам необходимо знание следующих тем по теории вероятностей и математической статистике:

1. Вероятность: понятие вероятности, теоремы сложения и умножения вероятностей, условная вероятность. *Предполагаемое время на разбор: 2 академических часа*
2. Случайные величины: дискретные и непрерывные случайные величины, законы распределения, характеристики распределения (математическое ожидание, дисперсия, стандартное отклонение). *Предполагаемое время на разбор: 4 академических часа*
3. Введение в математическую статистику: выборка, генеральная совокупность, точечные и интервальные оценки параметров распределения, проверка гипотез. *Предполагаемое время на разбор: 4 академических часа*
4. Теория случайных процессов: основы *Предполагаемое время на разбор: 4 академических часа*
5. Основы теории графов: основные определения, связность, вершинная связность, связные компоненты, связность вершин и ребер, пути, циклы, деревья. *Предполагаемое время на разбор: 2 академических часа*

**Вывод:**

За **16 академических часов** можно дать необходимую базу по теории вероятностей и математической статистике.

Важно отметить, что ученики на текущий момент проходят темы, где используют вероятность случайных событий. Например, в ЕГЭ по математике сейчас на это выделено 2 задания короткой части. Тема с графами изучается школьниками в том числе при подготовке к сдаче ОГЭ по информатике. На текущий момент для этого отведено 9 задание. В ЕГЭ графы также встречаются. Например, первое задание в ЕГЭ по информатике в текущем году отведено графам: школьникам предлагается проанализировать список смежности (весов) графа и ответить на вопрос задачами.

Таким образом, внедрение дополнительных тем будет соответствовать актуальным темам школьной программы.

Перейдем к описанию необходимых навыков и знаний по продвинутым библиотекам для анализа данных: Pandas и др.

# Изучение продвинутых библиотек для анализа данных

Приведем библиотеки на Python, необходимые для первых шагов в анализе данных:

1. Numpy: для работы с многомерными массивами и векторами, а также для выполнения математических операций над ними.
2. Pandas: для работы с данными в табличном формате, включая чтение и запись файлов, группировку, объединение и многое другое.
3. Matplotlib: для визуализации данных с помощью различных типов графиков и диаграмм.
4. Seaborn: расширение библиотеки Matplotlib, которое упрощает создание красивых и информативных графиков.
5. Scikit-Learn: библиотека для машинного обучения, содержащая множество алгоритмов классификации, регрессии и кластеризации.

Опишем основные навыки, которые должен получить ученик, работая с этими библиотеками.

**Numpy**

*Предполагаемое время на разбор: 6 академических часов*

Наиболее часто используемые методы и функции из библиотеки Numpy включают в себя:

1. Создание массивов: np.array(), np.zeros(), np.ones(), np.empty().
2. Доступ к элементам массива: arr[i], arr[:, i], arr[i, :].
3. Изменение формы и размера массива: reshape(), resize(), flatten().
4. Умножение и деление массивов: dot(), divide().
5. Математические операции: add(), subtract(), multiply(), power().
6. Генерирование случайных чисел: random(), random\_sample().
7. Вычисление суммы, среднего значения и стандартного отклонения: sum(), mean(), std().
8. Генерировка специальных функций: linspace(), logspace(), meshgrid().
9. Свёртка (convolution) и корреляция (correlation): convolve(), correlate().

**Pandas**

*Предполагаемое время на разбор: 6 академических часов*

Наиболее часто используемые методы и функции из библиотеки Pandas включают в себя следующие:

**Создание датафрейма**:

1. pd.DataFrame() - создание датафрейма из двумерного массива или списка словарей.
2. pd.read\_csv(), pd.read\_excel(), pd.read\_json(), pd.read\_sql() - чтение CSV, Excel, JSON и SQL файлов соответственно.

**Изменение структуры датафрейма:**

1. df.rename() - изменение имен столбцов.
2. df.set\_index() - установка индекса.
3. df.pivot\_table() - создание сводной таблицы.

**Фильтрация и сортировка данных:**

1. df[column\_name] - выбор столбца.
2. df[(condition)] - фильтрация данных по условию.
3. df.sort\_values(by=[‘column1’, ‘column2’, …]) - сортировка значений по нескольким столбцам.

**Группировка и агрегирование данных:**

1. groupby() - группировка данных.

**Matplotlib**

*Предполагаемое время на разбор: 6 академических часов*

Библиотека Matplotlib предлагает множество функций для создания визуализаций данных. Вот некоторые из наиболее часто используемых функций и методов:

1. Создание фигур и осей: plt.figure(), plt.subplot(), plt.axis(), plt.gca()
2. Настройка стилей осей: plt.style.use(), plt.tick\_params(), plt.ylim()
3. Добавление меток и подписей: plt.title(), plt.xlabel(), plt.ylabel(), plt.text()
4. Рисование графиков: plt.plot(), plt.scatter(), plt.bar(), plt.hist()
5. Управление масштабом и форматом графика: plt.tight\_layout(), plt.autoscale(), plt.rcParams
6. Сохранение и отображение графиков: plt.savefig(), plt.show()

Приведем примеры графиков, которые можно построить при помощи текущей библиотеки:

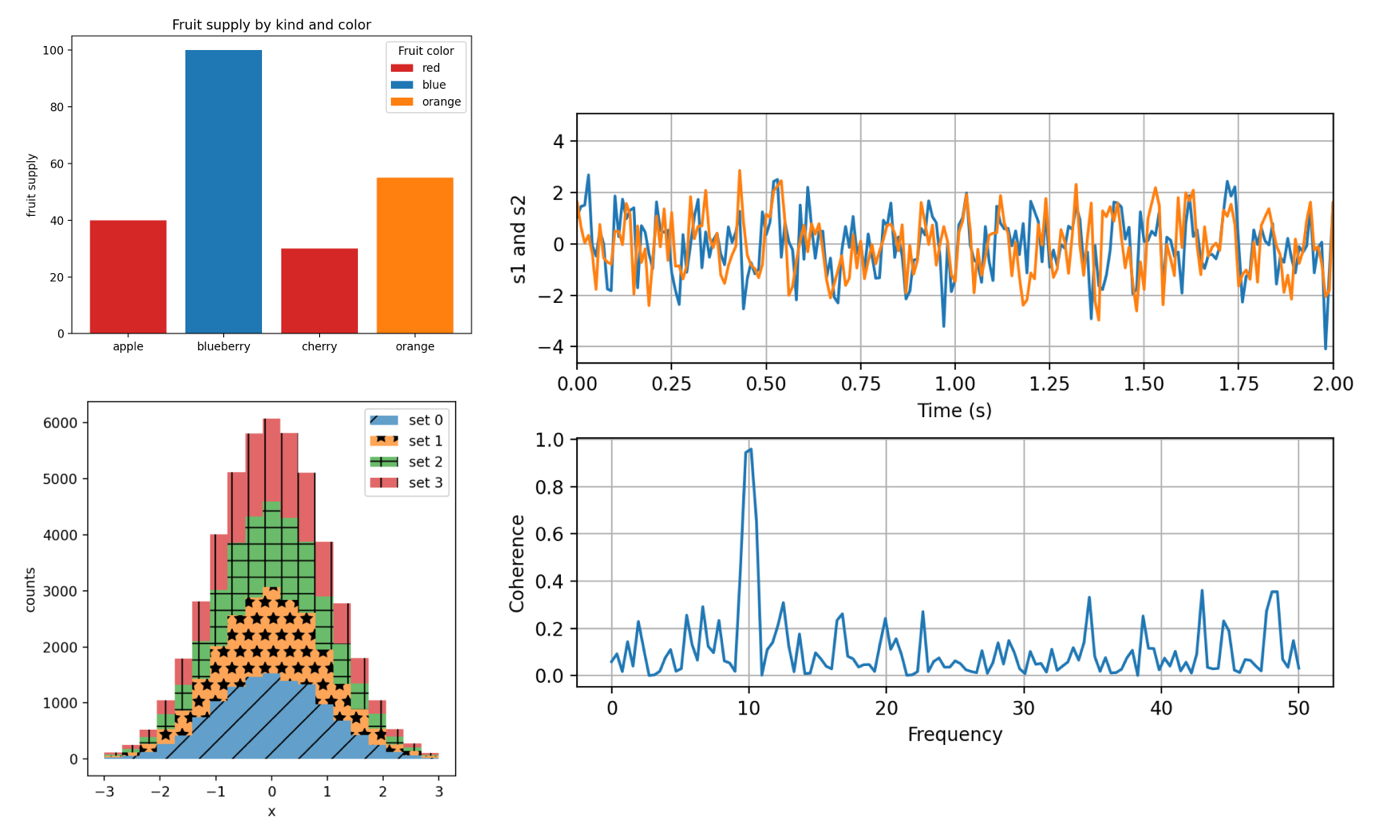


Рисунок 1

**Seaborn**

*Предполагаемое время на разбор: 6 академических часов*

Библиотека Seaborn предоставляет набор функций для создания красивых и информативных визуализаций. Вот некоторые из самых популярных функций:

1. pairplot(): создает точечную диаграмму для всех пар переменных в наборе данных.
2. displot(): генерирует различные типы графиков (например, гистограммы, точечные диаграммы и т. д.) для одной переменной.
3. lineplot(): строит линейный график для одной переменной против индекса.
4. joinplot(): объединяет точечную диаграмму и линейный график в одном графике.
5. heatmap(): создает тепловую карту для матрицы данных.

Приведем примеры графиков, которые можно построить при помощи текущей библиотеки:

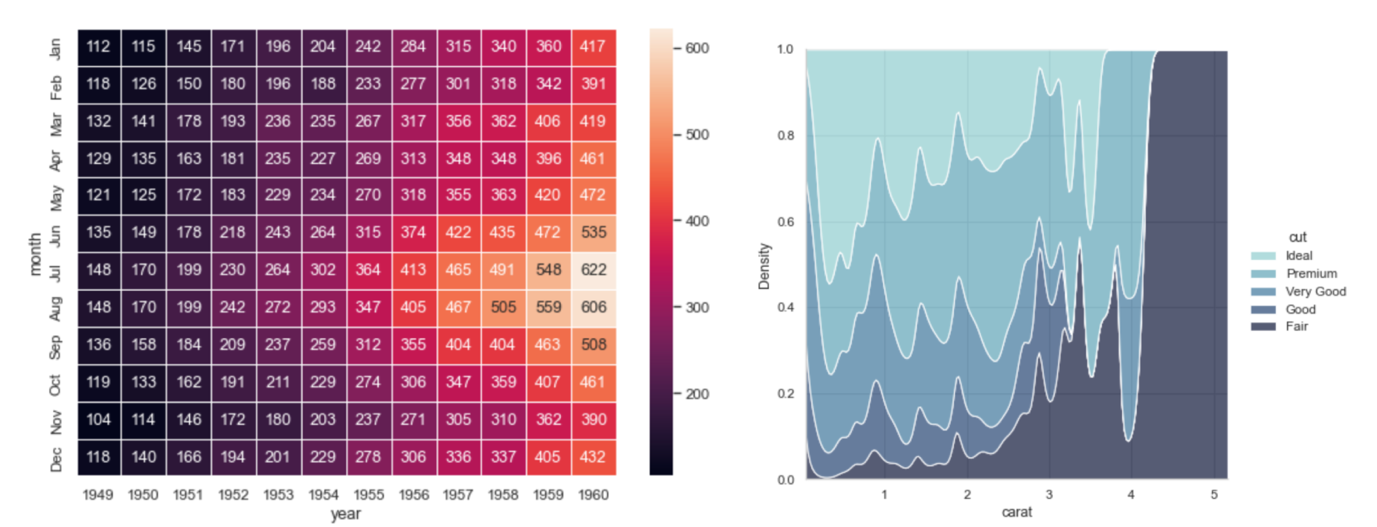


Рисунок 2

**Scikit-Learn**

*Предполагаемое время на разбор: 6 академических часов*

Scikit-Learn - это библиотека машинного обучения для Python, которая предоставляет множество алгоритмов для задач классификации, регрессии и кластеризации. Приведем основные алгоритмы, которые необходимы для разбора:

1. Линейная регрессия и логистическая регрессии
2. Деревья решений
3. Наивный Байесовский классификатор
4. Метод опорных векторов

Помимо этих функций, Scikit-Learn также предоставляет множество других инструментов для предварительной обработки данных, выбора признаков и визуализации результатов. Отметим, что в силу сложной математической структуры данных методов, знакомство с ними производится на базовом уровне. Ученик должен понять, для каких задач используется тот или иной алгоритм, а таже уметь оценивать точность предсказания модели и понимать, как влиять на точность ее результата. Приведем примеры задач, которые можно будет решить с помощью вышеописанных методов.

**Линейная регрессия**

Примеры задач:

1. Прогнозирование цен на товары или услуги
2. Определение оптимальной цены для продажи товара
3. Анализ взаимосвязи между различными факторами и показателями (например, между уровнем дохода и потреблением)
4. Определение факторов, влияющих на уровень продаж
5. Оценка эффективности маркетинговых кампаний
6. Прогнозирование спроса на товары или услуги

**Логистическая регрессия**

Примеры задач:

1. Классификация текстов: определение категории текста, такого как спам или не спам, на основе его содержания.
2. Предсказание отклика: предсказание вероятности того, что пользователь совершит определенное действие, например, купит продукт или зарегистрируется на сайте.
3. Распознавание образов: классификация изображений на основе их содержания, например, определение того, содержит ли изображение лицо или нет.
4. Прогнозирование временных рядов: прогнозирование будущих значений временного ряда на основе предыдущих значений.
5. Обнаружение аномалий: обнаружение необычных или неожиданных событий на основе исторических данных.

**Наивный Байесовский классификатор**

Примеры задач:

1. Классификация текстов: разделение текстов на различные категории, например, спам и не спам.
2. Определение тональности текста: классификация текста на положительный, отрицательный или нейтральный тон.
3. Обнаружение мошенничества: идентификация мошеннических транзакций на основе финансовых данных.
4. Прогнозирование оттока клиентов: определение вероятности того, что клиент прекратит пользоваться услугой.
5. Классификация изображений: разделение изображений на различные категории, например, кошки и собаки.
6. Распознавание речи: классификация аудиозаписей на различные языки или диалекты.

Метод опорных векторов

1. Классификация документов: разделение документов на различные категории на основе их содержимого.
2. Определение тональности текста: разделение текста на положительный и отрицательный тон.
3. Обнаружение спама: идентификация нежелательных электронных писем.
4. Прогнозирование продаж: предсказание будущих продаж на основе исторических данных.
5. Классификация изображений: разделение изображений на разные категории, такие как кошки и собаки.

Таким образом, после освоения текущих моделей, ученики смогут решать широкий спектр задач как прикладного, так и научного содержания.

**Вывод:**

За **30 академических часов** можно изучить и закрепить базовые инструменты для анализа данных на языке программирования Python.

# Анализ предложенного плана

Мы разобрали три основных блока, которые необходимы для подготовки и дальнейшего обучения по направлению «анализ данных». Еще раз напомним, что это:

1. Изучение основ Python и базового функционала (34 академических часа)
2. Изучение основ теории вероятностей и математической статистики (16 академических часов)
3. Изучение продвинутых библиотек для анализа данных: Pandas и др. (30 академических часов)

Итого, за 80 академических часов, текущий материал может быть освоен на базовом уровне и закреплен учениками. Текущую программу рекомендуется вводить в 10 и 11 классах и распределять на 1,5 года, чтобы вторую половину 11 класса ученики могли сфокусировано готовиться к ЕГЭ.

# Выводы

Подводя итог текущей работе, удалось разработать план подготовки по направлению компьютерные методы анализа больших данных в инженерных классах и группах дополнительного образования, с целью повышения качества знаний учащихся и их мотивации к изучению технических дисциплин.

Плюсы подхода:

1. Текущий план благоприятно повлияет на знание основ Python для учеников старших классов.
2. Помимо теоретических знаний, ученики получат инструменты для практического исследования, которые впоследствии будут полезны для обучения в ВУЗе и устройстве на работу.
3. Ученики смогут осмысленно самостоятельно заниматься проектной деятельностью.
4. Улучшаться в том числе теоретические знания разделов математики и информатики.

Трудности:

1. Общее количество часов вносит сложности в составление расписания.
2. Важным критерием является вопрос мотивации учеников. В отличие от ЕГЭ, текущие знания важны в большей степени для быстрого развития после школы.

В любом случае, внедрение подготовки по теме анализ данных в школах является важным и необходимым шагом для развития навыков критического мышления у школьников. Обучение анализу данных помогает учащимся развивать навыки, которые будут полезны не только в их будущей профессии, но и в повседневной жизни. Аналитическое мышление позволяет принимать обоснованные решения и делать выводы на основе имеющихся данных. Кроме того, анализ данных является ключевым навыком для многих профессий, таких как бизнес аналитика, data science, машинное обучение и другие. Таким образом, внедрение подготовки по анализу данных в школах может существенно повлиять на выбор будущей профессии школьниками и помочь им стать успешными специалистами в своей области.

# Список используемой литературы

1. Документация по python: <https://www.python.org>
2. Библиотека Numpy: <https://numpy.org>
3. Библиотека Pandas: <https://pandas.pydata.org>
4. Библиотека Matplotlib: <https://matplotlib.org>
5. Библиотека Seaborn: <https://seaborn.pydata.org>
6. Библиотек Scikit-Learn: <https://scikit-learn.org/stable/>