МИНОБРНАУКИ

Государственное бюджетное образовательное учреждение высшего образования

**«Санкт-Петербургский государственный морской технический университет»**

Кафедра судовой автоматики и измерений

**Курсовая работа**

По дисциплине

«Основы искусственного интеллекта»

На тему: «Создание синтетических данных»

Выполнила

Студентка группы 2450

Налимова Е.А.

Проверил

Доцент

Пшеничная Клавдия Викторовна

Санкт-Петербург, 2020 г.

**Содержание**

[Введение 3](#_Toc70369017)

[Требования к синтетическим данным 4](#_Toc70369018)

[Поле применения синтетических данных вкупе или отдельно с реальными 5](#_Toc70369019)

[Виды синтетических данных 7](#_Toc70369020)

[Часть 1. Генерация данных по заданному распределению 8](#_Toc70369021)

[Часть 2. Создание синтетических данных на основе линейной регрессионной модели 15](#_Toc70369022)

[Часть 3. Создание синтетических данных на основе нелинейной регрессионной модели 18](#_Toc70369023)

[Часть 4. Создание синтетических данных на основе задачи множества классов 24](#_Toc70369024)

[Часть 5. Создание синтетических данных на основе задачи назначения лейблов объектам 30](#_Toc70369025)

[Часть 6. Создание синтетических данных на основе задачи кластеризации 36](#_Toc70369026)

[Часть 7. Создание синтетических данных на основе задачи о спектральной кластеризации 43](#_Toc70369027)

[Часть 8. Пример «зашумления» изображений 47](#_Toc70369028)

[Заключение 49](#_Toc70369029)

[Список использованных источников 50](#_Toc70369030)

Введение

Говоря в нынешних реалиях об искусственном интеллекте, зачастую подразумевается область, которая используется как общее решение в задачах, в которых отсутствует прямая или требуется принципиально новый подход, который бы давал возможность сэкономить деньги.

Об искусственном интеллекте как о рабочем инструменте пишут и говорят на выступлениях конференций, посвященных данной тематике, такие крупные компании как Amazon, Google, Apple и другие титаны IT-индустрии, так как для них отсутствует проблема, встающая ребром для более скромных компаний, занимающихся разработкой AI-решений для задач, это объемы реальных данных.

Реальными данными называются сведения, полученные путем реальных наблюдений за событиями задачи, представленной на решение методами искусственного интеллекта. Бывают случаи, что количество реальных сведений недостаточно для обучающей выборки, возникает необходимость в создании синтетических данных.

Синтетические данные – данные, созданные на основе реальных для увеличения объема обучающей выборки, позволяют дополнить обучающий набор или используются для разделения на «учебник» и «тест адекватности».

Синтетические данные позволяют экономить на сборе реальных данных, который порой бывает слишком дорог, организуя новую нишу в сфере технологий машинного обучения, формируя новые сервисы, открывая новую сторону разработки, облегчают и ускоряют работу специалистов по данным, позволяя проекту не растягиваться по времени и финансам, обеспечивают безопасность процесса проектирования решений, так как предлагают альтернативу разглашению конфиденциальных данных, если суть задачи – создания продукта для определенного круга пользователей.

Поддельные данные – поле творчества, разработчик может добавлять исключения из сформировавшихся правил, менять и/или корректировать логическое направление модели, повышать качество выходного продукта, а с повышением качества можно претендовать на более высокую стоимость работы и успешнее конкурировать на рынке.

В данной работе рассматриваются некоторые способы создания синтетических данных, проводится анализ достоинств и недостатков.

Требования к синтетическим данным

1. Синтетические данные могут быть любого типа, начиная от простых двоичных последовательностей, заканчивая объемными моделями для обучения моделей компьютерного зрения. Основное требование для любого вида данных – их размеченость. Разметка данных – выделение некоторого объекта из нескольких, который используется как основа для обучения необходимой модели.
2. Количество объектов и длина набора данных должны быть произвольны. Ни один из методов генерации не должен ограничиваться размером возможной выборки.
3. В синтетических данных должна прослеживаться некоторая случайность, ширина статистического распределения выборки должна быть достаточной, случайность процесса – настраиваемой.
4. Сложность задачи должна быть контролируемой.
5. В выборке должен присутствовать случайный шум, размер которого можно задать.
6. Сгенерированные данные не могут противоречить реальным данным.
7. Синтетические данные являются дополнением реальных, а не полной заменой, они добавляют разнообразие в обучающую выборку.

Поле применения синтетических данных вкупе или отдельно с реальными

Создание синтетических данных выглядит как универсальное решение для разработчиков крупных по размеру моделей с малым объемом обучающих выборок. Однако не стоит забывать, что генерация синтетических данных должно являться запасным способом расширения объема данных для обучения.

Исходя из вышесказанного, перечисляем положительные моменты использования синтетических данных:

* *Дешевизна*. Стоимость реальных баз данных идет в начале списка, так как зачастую крупную долю стоимости проекта занимает сбор реальных данных, разметка и подготовка к дальнейшей работе. Сама разработка не так дорога, как сбор материала, так как при возможности безвредно сэкономить – лучше экономить, становится целесообразно создавать «поддельные» данные.
* *Упрощение и стандартизация*. В случаях баз данных, собранных на основе информации пользователей (часто применяется при разработке бизнес-решений), выборка может получиться достаточно огромной, что, казалось бы, является плюсом, однако на этапе обработки данных и выделении не похожих меж собой единиц выборки, оказывается, что многие данные сходны и могут быть объединены группы или классы (возникновение задачи кластеризации), что приводит к дистилляции данных[[1]](#footnote-1), после которой существенно сокращается размер выборки и, как следствие, время обработки и обучения модели.
* *Комбинаторика и покрытие тестами*. Несмотря на размер выборки, в ней могут отсутствовать возможные случаи работы модели. Наличие «исключений из правил» является одним из факторов формирования «адекватности поведения»[[2]](#footnote-2) продукта.
* *Автоматизация.* При тестировании модели автоматически, нужно создать условия, в которых будет производится тест по определенным настройкам, для этого синтетические данные – наилучший вариант.
* *Повышение управляемости.* Зачастую после обучения модели, встает вопрос о том, каким образом она научилась и какова её реакция на другие данные. Например, принцип «черного ящика» у скрытых слоев ИНС не позволяет на 100% понимать, что происходит внутри, оставляя возможность манипуляций лишь на входном и выходном слоях. Синтетические данные являются эффективным средством проверки для моделей, ввиду способности оценить реальный результат на соответствие ожидаемому.

Вышеизложенные положительные стороны могут ввести в заблуждение относительно применимости синтетических данных в любом проекте, однако существует ряд минусов в использовании данной технологии:

* *Действительно большие данные.* Порой при дистилляции важные примеры случаев, делающие модель реальной, отбрасываются, вследствие чего выборка «идеализируется» и становится далека от действительного условия задачи. В итоге на сокращенной базе данных модель работает идеально, но в реальных условиях не проходит ни один тест. Об этом необходимо всегда помнить.
* *Разнообразный формат данных.* При вариативности и отсутствии возможности предугадать какой именно вид данных будет подгружаться в определенное поле базы данных использование синтетических данных может только усложнить задачу и потребовать больше времени на решение.
* *Непредсказуемый пользователь.* Если база данных для выборки собирается на основе действий пользователей (человека), то угадать поведение человека невозможно, поэтому если в задаче есть человеческий фактор, стоит отдать предпочтение реальным данным.
* *Обновления.* Когда модель тренируется на какой-то системе, все может пойти не так после обновления данной системы, поэтому стоит учитывать и этот фактор при построении работы модели.

С учетом приведенных достоинств и недостатков синтетических данных, можно сделать вывод, что их использование целесообразно, когда данные предсказуемы, и появление «случайностей» маловероятно. Однако синтетические проигрывают реальным, когда существует человеческий фактор.

Виды синтетических данных

Говоря о видах синтетических данных, стоит обратить внимание, что создание искусственных данных практически не имеет ограничений относительно того, текстовая, графическая, звуковая или иная форма информации представлена в выборке. Однако стоит помнить, что сгенерировать какое-либо число намного проще, чем объемную модель, поэтому вопрос упирается зачастую в количество времени и затрачиваемых ресурсов. Вследствие этого при генерации искусственных данных нужно постоянно обращать внимание на структурную сложность, возможно, что набрать реальные примеры получится быстрее, чем создать искусственные.

Часть 1. Генерация данных по заданному распределению

В случаях, когда синтетические данные генерируются относительно процесса, который можно описать некоторым существующим математическим распределением, описанным в теории вероятности, можно пользоваться следующими способами:

***Равномерное распределение вероятности (uniform):*** вид вероятности, в котором события имеют одинаковую возможность реализации. Наиболее часто приводящийся пример в теории вероятности– это график прихода автобуса. При ожидании автобуса, интервал которого составляет 10 минут вероятность подойти на 1, 4 или 8 минуте интервала будет равен 1/10. При генерации синтетических данных мы задаем одинаковый вес (вероятность появления) нужного нам числа (или иного вида данных).

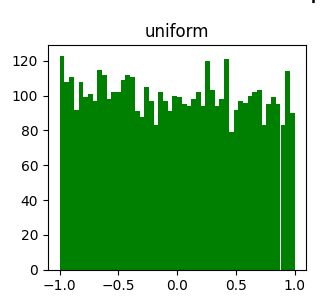
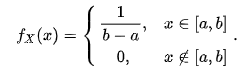
 

Рисунок 1 – равномерное распределение

Где a, b – конечные точки.

***Нормальное распределение или Гауссовское (normal***): наиболее распространенный вид вероятности, который описывает большинство процессов нашей жизни, так как в них существует принятая норма и некоторые отклонения от нее.

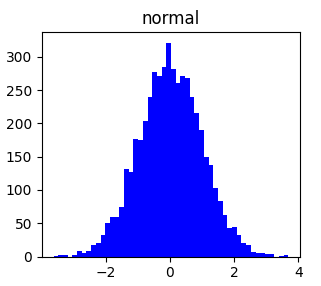
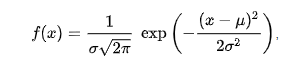
 

Рисунок 2 – нормальное (гауссовское) распределение

Где μ- мат.ожидание, σ- среднеквадратичное отклонение.

***Экспоненциальное распределение (exponental):*** экспоненциальное распределение широко применимо в законах физики, например, описывается вероятность распада ядра радиоактивного вещества. Или проще – вероятность появления покупателя в магазине после предыдущего.

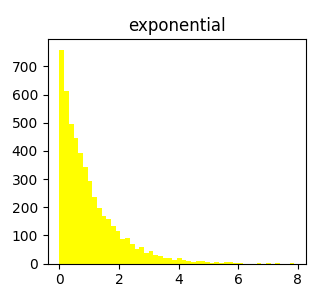
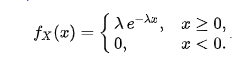
 

Рисунок 3 – экспоненциальное распределение

Где λ- интенсивность (вероятность появления события)

***Логарифмически нормальное распределение (lognormal):*** логнормальное распределение описывает процесс дробления, а именно с какой скоростью будет уменьшаться размер объекта при его разбивании, или при моделировании доходов, или при рассчете возможности появления в продуктах ядовитых веществ.

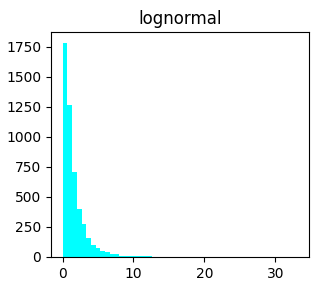
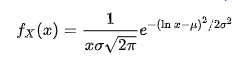
 

Рисунок 4 – логнормальное распределение

***Распределение хи-квадрат (chisquare):*** распределение, позволяющее проверить некоторые статистические гипотезы применимые в генетике, психологии и других науках, в которым присутствует статистика.

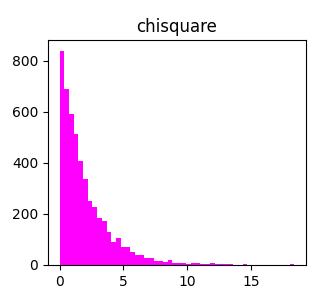
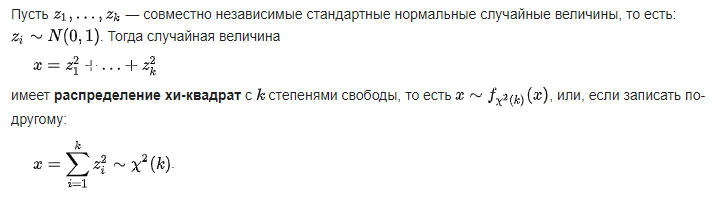


Рисунок 5 – распределение хи-квадрат



***Бета распределение (beta):*** это распределение вероятностей по вероятностям, например, посещаемость сайта, конверсия клиентов, активность пользователей, прогноз жизни больных раком.

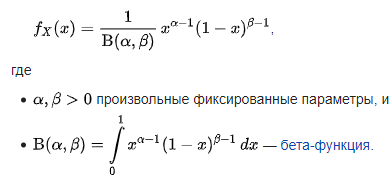
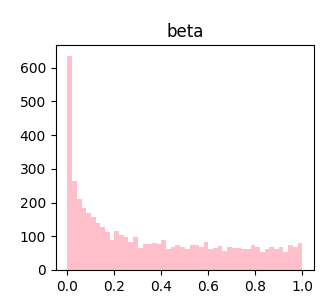


Рисунок 6 – бета-распределение

Ниже приведен код на языке Python 3.8.3 позволяющий генерировать данные по заданным вероятностям.

import numpy as np  
  
import matplotlib.colors  
import matplotlib.pyplot as plt  
from openpyxl.workbook import Workbook  
from mpl\_toolkits.mplot3d import axes3d  
  
import sklearn.datasets as dt  
  
from sklearn.neighbors import KernelDensity  
from sklearn.model\_selection import GridSearchCV  
  
seed = 10  
rand\_state = 10  
  
color\_map = plt.cm.get\_cmap('RdYlBu')  
color\_map\_discrete = matplotlib.colors.LinearSegmentedColormap.from\_list("", ["red","cyan","magenta","blue"])  
  
rand = np.random.RandomState(seed)  
  
dist\_list = ['uniform','normal','exponential','lognormal','chisquare','beta']  
param\_list = ['-1,1','0,1','1','0,1','2','0.5,0.9']  
colors\_list = ['green','blue','yellow','cyan','magenta','pink']  
  
fig, ax = plt.subplots(nrows=2, ncols=3, figsize=(12, 7))  
plt\_ind\_list = np.arange(6) + 231  
  
filename = input('Введи путь файла: ')  
datas = []  
for dist, plt\_ind, param, colors in zip(dist\_list, plt\_ind\_list, param\_list, colors\_list):  
 x = eval('rand.' + dist + '(' + param + ',5000)')  
 a = np.ndarray.tolist(x)  
 datas.append(a)  
 plt.subplot(plt\_ind)  
 plt.hist(x, bins=50, color=colors)  
 plt.title(dist)  
# Вывод в визуал  
import pandas as pd  
frame = pd.DataFrame(datas, index=dist\_list)  
with pd.ExcelWriter(filename) as writer:  
 frame.to\_excel(writer)  
fig.subplots\_adjust(hspace=0.4, wspace=.3)  
plt.suptitle('Sampling from Various Distributions', fontsize=20)  
plt.show()

Библиотека numpy – средство для работы с математической частью генерации искусственных данных, основная использующаяся функция – random, делающая создаваемые данные «случайными» относительно заданного распределения.

Библиотека sklearn – наиболее распространенный инструмент в машинном обучении для формирования данных в обучающую выборку по определенному закону, которые будут рассмотрены далее.

Библиотека matplotlib – визуализатор в Python, иногда необходим для отслеживания поведения данных в виде графиков, схем, диаграмм и других способов визуального представления информации.

Библиотека pandas – набор инструментов для чтения и записи данными в файлы, пригодные для восприятия человеком. С помощью данной библиотеки возможно составить детальный анализ по данным в виде диаграмм, таблиц, графиков. Работает с различными средствами построения баз данных: 1С, Microsoft Excel, Microsoft Access, а также с URL-ссылками.

В результате исполнения данного кода был получен следующий набор данных, размер данной выборки 6х5000:

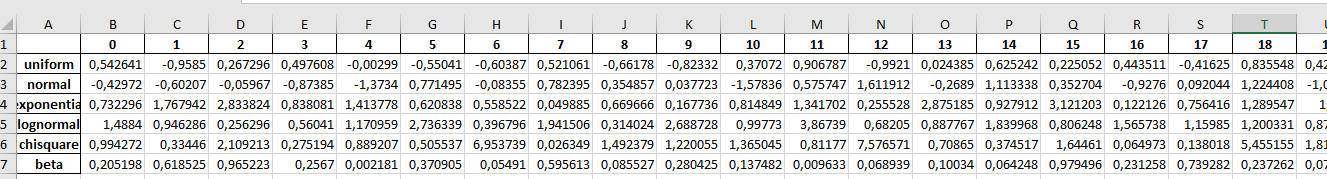


Рисунок 7 – набор данных

В доказательство соответствия созданных данных заданным распределениям были составлены диаграммы соотношений количества сгенерированных данных по промежуткам:

Рисунок 8 – соотношение количества данных при равномерном распределении

Рисунок 9 – соотношение количества данных при нормальном распределении

Рисунок 10 – соотношение количества данных при экспоненциальном распределении

Рисунок 11 – соотношение количества данных при логнормальном распределении

Рисунок 12 – соотношение количества данных при хи-распределении

Рисунок 15 – количество данных при бета-распределении

Часть 2. Создание синтетических данных на основе линейной регрессионной модели

Вновь обратимся к библиотеке sklearn, а точнее к встроенному методу создания данных по регрессионной модели (как линейной, так и нелинейной).

Функция make\_regression() обеспечивает генерацию модели данных, которые описывают некоторые процессы с некоторым «идеальным» течением, которое можно описать линейной функцией, вокруг которой будет разбросан «шум» - результат функции зависимости входных параметров с прибавлением случайной величины, которая моделируется с помощью random (псевдослучайного генератора).

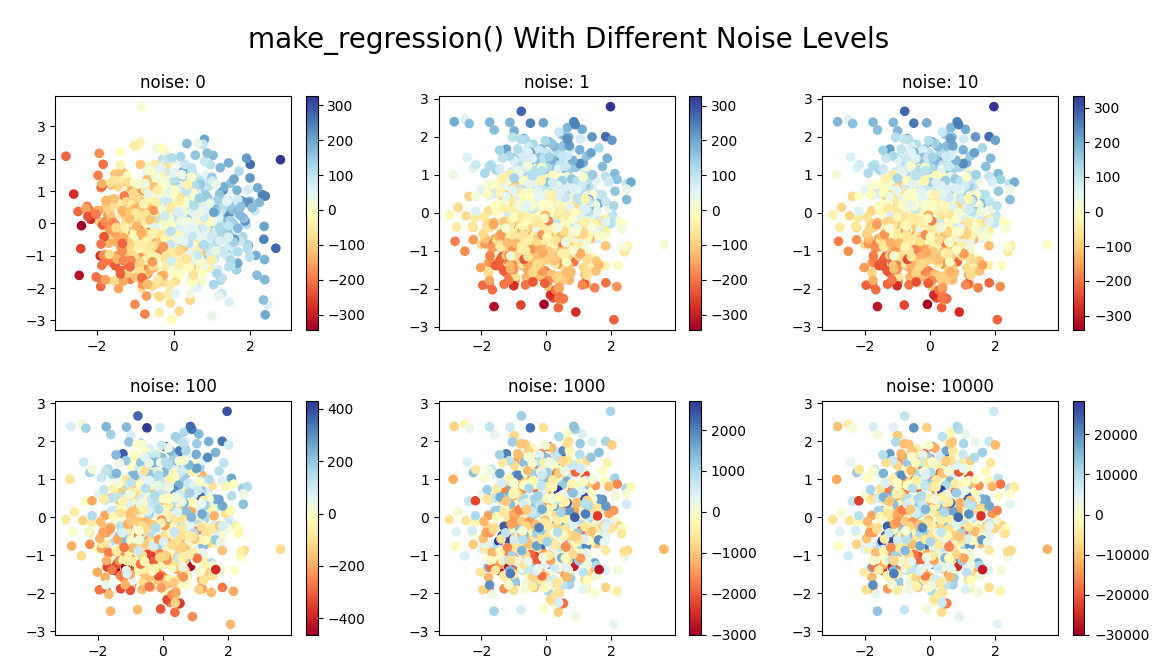


Рисунок 16 – графики распределения регрессионной модели с разными уровнями шума

Функция make\_regression() смоделировала линейную функцию с некоторыми псевдослучайными коэффициентами:

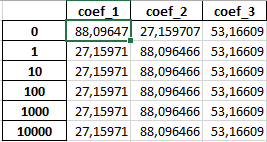


Рисунок 17 – коэффициенты регрессионной модели

При изменении шума коэффициенты остаются практически неизменными.

Однако критерий хаотичности модели заключается в уровне шума, который можно получить из сравнения значения, полученного в результате регрессии и чистого значения, полученного путем подстановки в уравнение:

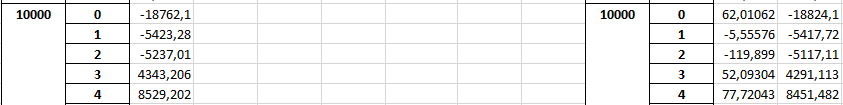


Рисунок 18 – значения при хаотичности 10000

В левой части представлено значение, выданное программой, в правой: сначала – значение, полученное без шума, следом – значение шума. Число 10000 – число, около которого по нормальному распределению формируется шум.



Рисунок 19 – соотношение данных попадающих под предел шума

В результате имеем закономерную картину: при нулевом шуме отклонений практически нет, поэтому все экземпляры выборки имеют шум ниже или равный 0, остальные имеют одинаковое соотношение:

Рисунок 20 – гистограмма размера разброса шума

Собственно, код программы:

import numpy as np  
  
import pandas as pd  
import matplotlib.colors  
import matplotlib.pyplot as plt  
from openpyxl.workbook import Workbook  
from mpl\_toolkits.mplot3d import axes3d  
  
import sklearn.datasets as dt  
  
from sklearn.neighbors import KernelDensity  
from sklearn.model\_selection import GridSearchCV  
seed = 10  
rand\_state = 10  
  
filename = input('Введите путь файла: ')  
  
# Define the color maps for plots  
color\_map = plt.cm.get\_cmap('RdYlBu')  
color\_map\_discrete = matplotlib.colors.LinearSegmentedColormap.from\_list("", ["red","cyan","magenta","blue"])  
  
fig, ax = plt.subplots(nrows=2, ncols=3, figsize=(16, 7))  
plt\_ind\_list = np.arange(6) + 231  
dataset\_x=[]  
dataset\_y=[]  
dataset\_coef = []  
spise\_noise = [0,1,10,100,1000,10000]  
for noise, plt\_ind in zip(spise\_noise, plt\_ind\_list):  
 x, y, coef = dt.make\_regression(n\_samples=1000,  
 n\_features=3,  
 noise=noise,  
 random\_state=rand\_state,  
 coef=True)  
  
 plt.subplot(plt\_ind)  
 dataset\_x.append(x)  
 dataset\_y.append(y)  
 dataset\_coef.append(coef)  
 my\_scatter\_plot = plt.scatter(x[:, 0],  
 x[:, 1],  
 c=y,  
 vmin=min(y),  
 vmax=max(y),  
 s=35,  
 cmap=color\_map)  
  
 plt.title('noise: ' + str(noise))  
 plt.colorbar(my\_scatter\_plot)  
  
  
dataset\_x\_new=np.concatenate(dataset\_x)  
dataset\_y\_new=np.concatenate(dataset\_y)  
ines = pd.MultiIndex.from\_product([spise\_noise,np.arange(1000)])  
colms\_x = pd.MultiIndex.from\_product([['x'],[1,2,3]])  
df\_x = pd.DataFrame(dataset\_x\_new,index=ines,columns=colms\_x)  
df\_y = pd.DataFrame(dataset\_y\_new,index=ines, columns=['y'])  
df\_coefs = pd.DataFrame(dataset\_coef, index=spise\_noise, columns=['coef\_1','coef\_2','coef\_3'])  
with pd.ExcelWriter(filename) as writer:  
 df\_x.to\_excel(writer)  
 df\_y.to\_excel(writer, startcol=5,startrow=2)  
 df\_coefs.to\_excel(writer, startcol=9, startrow=2)  
fig.subplots\_adjust(hspace=0.3, wspace=.3)  
plt.suptitle('make\_regression() With Different Noise Levels', fontsize=20)  
plt.show()

Регрессия используются во всех задачах, где требуется предсказание, например, поведения устройства при некоторых коэффициентах или настройках. Поле применения достаточно широко, применение математической регрессии достаточно просто в силу того, что выражается линейной функцией, что говорит о равномерности процесса даже с отклонениями.

Часть 3. Создание синтетических данных на основе нелинейной регрессионной модели

В библиотеке sklearn существуют функции для формирования данных на основе нелинейных моделей. Представлены они 3-мя вариациями make\_friedman, которые представляют собой нелинейное многомерное моделирование (от 4х переменных):

Make\_friedman1() – вычисляет с помощью функции в которую включены синусоида и квадратичная функция, обычно использующиеся для объяснения колебательных движений и описания симметричности процесса:



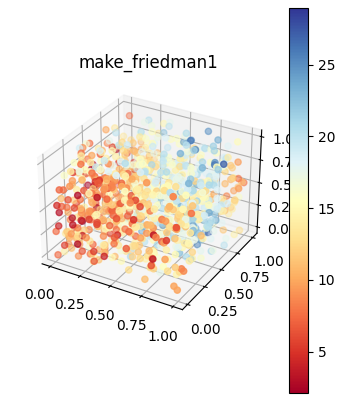


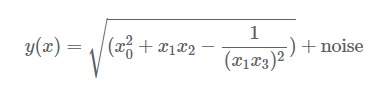
Рисунок 22 – результат моделирования на make\_friedman1

Так же была произведена проверка надежности функции:

Рисунок 23 – соотношение совпадений с аналитическим решением

В результате было выявлено, что в 96% случаев значение, полученное через уравнение, соответствует значению, выданному функцией.

Make\_friedman2() – вычисляет функцию квадратичного корня, функция такого вида обычно используется для расчета скорости, отклонения, ускорения и других, вычисляемых через квадрат величин:



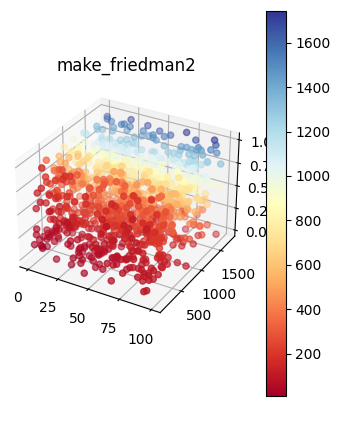


Рисунок 24 – результат моделирования функции make\_friedman2()

Так же была произведена проверка качества полученного результата:

Рисунок 25 - соотношение совпадений с аналитическим решением

На диаграмме видно, что в 95% случаев результат вычисленной функции и смоделированной совпадают.

Make\_friedman3() – функция арктангенса, обычно тангенсоиды используются как активационные функции в нейросетях, по причине того, что функция описывает бесконечно возрастающий процесс[[3]](#footnote-3), через который можно описать бесконечное множество монотонно[[4]](#footnote-4) меняющихся состояний.

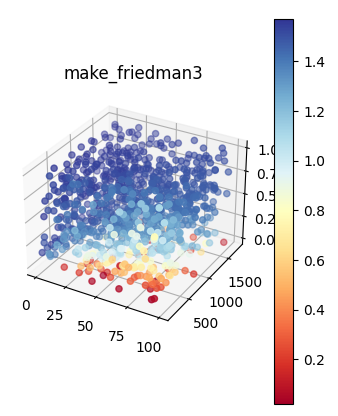


Рисунок 26 – результат моделирования make\_friedman3()

Качество полученного результата:

Рисунок 27 - соотношение совпадений с аналитическим решением

Здесь так же в 95% случаев вычисление через функцию совпало с результатом программы.

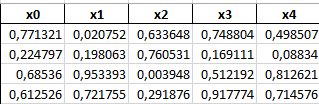


Рисунок 28 – коэффициенты make\_friedman1

В случае make\_friedman1 рандомизация коэффициентов от до не имеет сильный разброс, от этого не удается найти переменную, которая наиболее сильно бы влияла на результат функции.

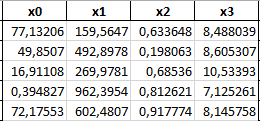


Рисунок 29 – коэффициенты make\_friedman3

С make\_friedman3 коэффициенты имеют сильный разброс, но результат остается в довольно узком диапазоне, в следствии этого так же с первого взгляда не выделить наиболее доминантную переменную:



Рисунок 30 – диапазон результатов функции

Однако в make\_friedman2 все более очевидно:

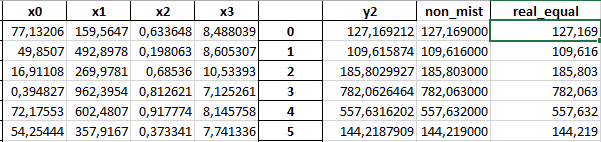


Рисунок 31 – коэффициенты make\_friedman2

Изменения результата функции наиболее связаны с , при резком скачке числа вверх происходит увеличение значения результата.

Код программы:

import numpy as np  
  
import pandas as pd  
import matplotlib.colors  
import matplotlib.pyplot as plt  
from openpyxl.workbook import Workbook  
from mpl\_toolkits.mplot3d import axes3d  
  
import sklearn.datasets as dt  
  
from sklearn.neighbors import KernelDensity  
from sklearn.model\_selection import GridSearchCV  
seed = 10  
rand\_state = 10  
  
filename = input('Введите путь файла для записи: ')  
# Define the color maps for plots  
color\_map = plt.cm.get\_cmap('RdYlBu')  
color\_map\_discrete = matplotlib.colors.LinearSegmentedColormap.from\_list("", ["red","cyan","magenta","blue"])  
fig = plt.figure(figsize=(18,5))  
  
x,y = dt.make\_friedman1(n\_samples=1000,n\_features=5,random\_state=rand\_state)  
dataset\_x1=x  
dataset\_y1=y  
ax = fig.add\_subplot(131, projection='3d')  
my\_scatter\_plot = ax.scatter(x[:,0], x[:,1],x[:,2], c=y, cmap=color\_map)  
fig.colorbar(my\_scatter\_plot)  
plt.title('make\_friedman1')  
  
x,y = dt.make\_friedman2(n\_samples=1000,random\_state=rand\_state)  
dataset\_x2 = x  
dataset\_y2 = y  
ax = fig.add\_subplot(132, projection='3d')  
my\_scatter\_plot = ax.scatter(x[:,0], x[:,1],x[:,2], c=y, cmap=color\_map)  
fig.colorbar(my\_scatter\_plot)  
plt.title('make\_friedman2')  
  
x,y = dt.make\_friedman3(n\_samples=1000,random\_state=rand\_state)  
dataset\_x3 = x  
dataset\_y3 = y  
ax = fig.add\_subplot(133, projection='3d')  
my\_scatter\_plot = ax.scatter(x[:,0], x[:,1],x[:,2], c=y, cmap=color\_map)  
fig.colorbar(my\_scatter\_plot)  
plt.suptitle('make\_friedman?() for Non-Linear Data',fontsize=20)  
plt.title('make\_friedman3')  
df\_x1 = pd.DataFrame(dataset\_x1, columns=['x0','x1','x2','x3','x4'])  
print(df\_x1)  
df\_y1 = pd.DataFrame(dataset\_y1, columns=['y1'])  
print(df\_y1)  
df\_x2 = pd.DataFrame(dataset\_x2, columns=['x0','x1','x2','x3'])  
print(df\_x2)  
df\_y2 = pd.DataFrame(dataset\_y2,columns=['y2'])  
print(df\_y2)  
df\_x3 = pd.DataFrame(dataset\_x3,columns=['x0','x1','x2','x3'])  
print(df\_x3)  
df\_y3 = pd.DataFrame(dataset\_y3,columns=['y3'])  
print(df\_y3)  
with pd.ExcelWriter(filename) as writer:  
 df\_x1.to\_excel(writer, sheet\_name='Sheet1')  
 df\_y1.to\_excel(writer, sheet\_name='Sheet1', startcol=6)  
 df\_x2.to\_excel(writer, sheet\_name='Sheet2')  
 df\_y2.to\_excel(writer, sheet\_name='Sheet2', startcol=5)  
 df\_x3.to\_excel(writer, sheet\_name='Sheet3')  
 df\_y3.to\_excel(writer, sheet\_name='Sheet3', startcol=5)  
plt.show()

Все 3 функции строятся на критерии Фридмана, проверяющем однородность статистических данных, проще говоря проверяет насколько велико различие между объектами одной выборки. Это используется, когда мы хотим определить насколько различно качество методов работы, которую выполняют с одной и той же целью, но разными способами.

Например, мы имеем оборудование и подразделения, которые выполняют одну и ту же работу на станках разного производителя, подразделения имеют рабочих, значения выборки – показатель производительности рабочего. Решением задачи будет вывод: дает ли оборудование одной фирмы лучший результат, чем оборудование других фирм.

Часть 4. Создание синтетических данных на основе задачи множества классов

Когда необходимо сгенерировать данные, которые бы относились к нескольким классам, для этой задачи походит make\_classification.

Задача нескольких классов встает при распознавании рукописного текста иностранных языков, в особенности иероглифов или слитной речи. В данном работе использован простой пример: классификация по четвертям координатной оси:

Наиболее важной переменной функции make\_classification выступает class\_sep – она выражает величину разницы значений между классами.

При переменной class\_sep = 0,01 мы имеет следующий график разброса:

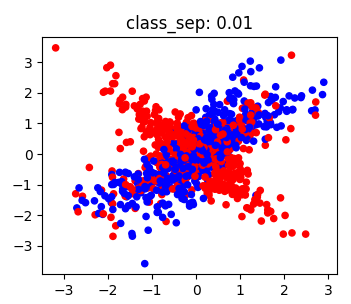


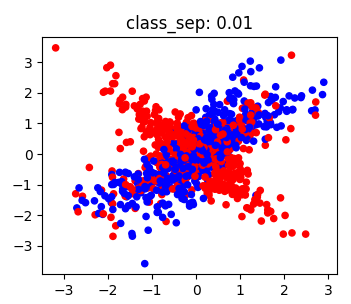
Рисунок 32 – график разброса при class\_sep=0,01

Полученная картина говорит о довольно сложной и неточной классификации, в чем можно убедиться по соотношению точек:

Рисунок 33 – соотношение количества точек в квадратах класса 0 при 0,01

Рисунок 34 – соотношение количества точек в квадратах класса 1 при 0,01

Класс с индексом 0 – скопление красных точек, класс с индексом 1 – скопление синих точек. Для точности определения кластеризации были введены квадранты:



II quatro

I quatro

IV quatro

III quatro

Рисунок 35 – система координат

Диаграммы показывают процентное соотношение скопления точек в разных квадратах. Большое количество точек класса 1 собрано в квадратах 1 и 3: в 1ом – 46%, в 3ем – 39%. Точки же класса 0 примерно в равном количестве находятся во всех квадратах.

При различии 0,1:

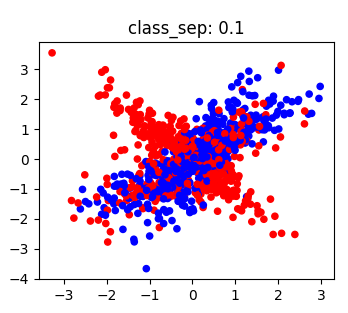


Рисунок 36 – график разброса при параметре class\_sep = 0.1

Рисунок 37 - соотношение количества точек в квадратах класса 0 при 0,1

Рисунок 38 - соотношение количества точек в квадратах класса 1 при 0,1

При параметре class\_sep = 0.1 так же не наблюдается больших изменений относительно значения class\_sep = 0.01, однако происходит смещение точек класса 0 в квадраты 2 и 4.

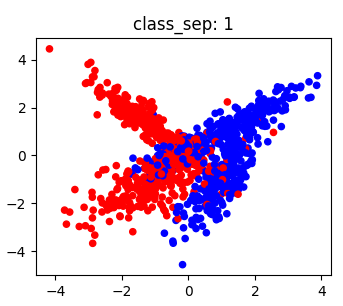


Рисунок 39 – график разброса при параметре class\_sep = 1

Здесь уже кластеризация видна намного четче, в чем можно убедиться, обратившись к диаграммам:

Рисунок 40 - соотношение количества точек в квадратах класса 0 при 1

Существенное смещение красных точек позволило выделить оба класса в 2 более четких кластера. Теперь 85% находятся в квадратах 2 и 3.

Рисунок 41 - соотношение количества точек в квадратах класса 1 при 1

А точки класса 1 (синие) теперь преимущественно находятся в квадрате 1 и 4 (86%), данная картина уже соответствует простой кластеризации, когда классы проще отделить друг от друга.

При class\_sep = 10 кластеры отделены друг от друга наиболее четко:

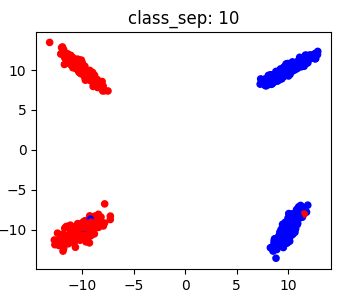


Рисунок 42 – график разброса при параметре class\_sep = 10

Рисунок 43 - соотношение количества точек в квадратах класса 0 при 10

Рисунок 44 - соотношение количества точек в квадратах класса 1 при 10

По диаграммам видно, что класс 0 занимает полностью 2 и 3 квадраты, а класс 1 – 1 и 4 квадраты.

Это четкая классификация – наиболее простой случай классификации, в нем отсутствует нечеткая логика и не требуются дополнительные исследования, при маленьких значениях параметра clаss\_sep функция распределяла экземпляры по классам нечетко, однако четкость или нечеткость разделения зависит от величины рассматриваемого масштаба. Если бы мы рассматривали график разброса class\_sep = 0.01 в более мелком масштабе (<0.01) то фрагменты классов выглядели бы более четко, однако при формировании данных с множеством классов их требуется рассматривать в заданном изначально масштабе.

Примером задачи классификации может быть распознавание фотографий.

Фотография – большой объект, на котором может присутствовать что угодно. Классификация – задача присвоения этим объектам меток. По присвоенным меткам происходит идентификация изображения на фотографии, например, собака или кошка, если собака, то какого цвета, какой породы, старая или щенок и т.д. Классификация широко применима и в компьютерном зрении, машина определяет, что конкретно она видит по знакомым меткам, если машина обучалась видеть лишь собак, то и кошек, и лошадей, и других четвероногих животных будет видеть как «необычных собак», либо искренне недоумевать что передней находится.

Часть 5. Создание синтетических данных на основе задачи назначения лейблов объектам

Лейбл – маркировка объекта данных, относящегося к некоторому классу при условии, что нам известны размеры классов.

Эта задача хорошо проиллюстрирована в рекомендательных системах[[5]](#footnote-5). Например, фильм «Аватар» относится к жанрам «Боевик», «Киноэпопея», «Фэнтези», «Научная фантастика», «Приключения». Посмотрев этот фильм на платформе, пользователю в рекомендациях могут выпасть фильмы таких же жанров: «Властелин колец», «Гарри Поттер», «Сокровища нации» и другие.

При генерации данных, которые бы соответствовали нескольким классам, снова обращаемся к sklearn, методу make\_multilabel\_classification().

В аргументы функции мы задаем количество наборов входных данных на каждый лейбл, количество переменных во входных данных, сколько классов сформировать и сколько лейблов можно присвоить объекту.

Код программы с генерацией большой выборки:

import numpy as np  
  
import pandas as pd  
import matplotlib.colors  
import matplotlib.pyplot as plt  
from openpyxl.workbook import Workbook  
  
import sklearn.datasets as dt  
  
from sklearn.neighbors import KernelDensity  
from sklearn.model\_selection import GridSearchCV  
seed = 10  
rand\_state = 10  
  
filename = input("Введите путь файла: ")  
# Define the color maps for plots  
color\_map = plt.cm.get\_cmap('RdYlBu')  
color\_map\_discrete = matplotlib.colors.LinearSegmentedColormap.from\_list("", ["red","cyan","magenta","blue"])  
fig, ax = plt.subplots(nrows=1, ncols=4, figsize=(18, 7))  
plt\_ind\_list = np.arange(3) + 131  
  
dataset\_x = []  
dataset\_y = []  
dataset\_sparse = []  
labels = [1,2,4]  
for label, plt\_ind in zip(labels, plt\_ind\_list):  
 x, y = dt.make\_multilabel\_classification(n\_samples=1000,  
 n\_features=4,  
 n\_labels=label,  
 n\_classes=5,  
 random\_state=rand\_state)  
 target = np.sum(y \* [1,1,1,1,1], axis=1)  
 dataset\_x.append(x)  
 dataset\_y.append(y)  
 plt.subplot(plt\_ind)  
 my\_scatter\_plot = plt.scatter(x[:, 0],  
 x[:, 1],  
 c=target,  
 vmin=min(target),  
 vmax=max(target),  
 cmap=color\_map)  
 plt.title('n\_labels: ' + str(label))  
n\_ds\_x = np.concatenate(dataset\_x)  
n\_ds\_y = np.concatenate(dataset\_y)  
cols\_x = pd.MultiIndex.from\_product([['x'],[1,2,3,4]])  
cols\_y = pd.MultiIndex.from\_product([['y'],[1,2,3,4,5]])  
ines = pd.MultiIndex.from\_product([['label\_1','label\_2','label\_4'],np.arange(1000)])  
df\_x = pd.DataFrame(n\_ds\_x,columns=cols\_x, index=ines)  
df\_y = pd.DataFrame(n\_ds\_y,columns=cols\_y,index=ines)  
with pd.ExcelWriter(filename) as writer:  
 df\_x.to\_excel(writer)  
 df\_y.to\_excel(writer, startcol=7)  
fig.subplots\_adjust(hspace=0.3, wspace=.3)  
plt.suptitle('make\_multilabel\_classification() With Different n\_labels Values', fontsize=20)  
plt.show()

И результат работы программы:

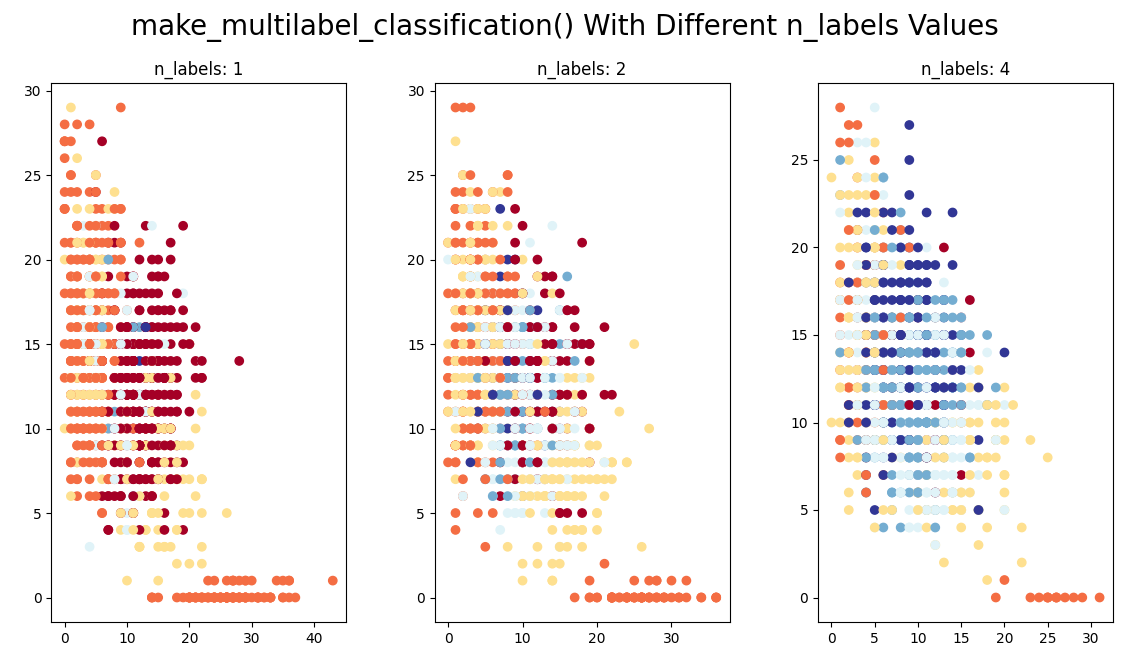


Рисунок 45 – график разброса объектов по классам

Получившиеся данные:

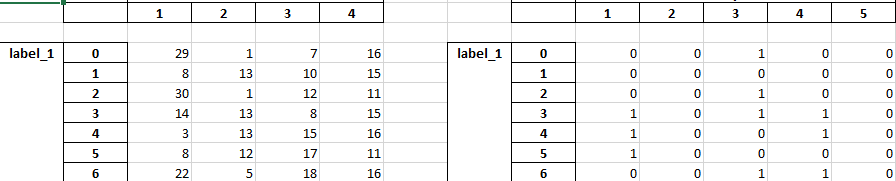


Рисунок 46 – таблица данных

Мы имеем 4 входных параметра, 5 классов и варианты назначения количества лейблов. Графики разброса недостаточно хорошо отображают работу функции, так как показана зависимость лишь от двух переменных. Сократим до 20 строк на лейбл, 3 переменных, 3х классов и возможность назначения от 1ого до 3х лейблов.

Получившийся сокращенный код:

import matplotlib.colors  
import matplotlib.pyplot as plt  
import sklearn.datasets as dt  
import pandas as pd  
import openpyxl  
from mpl\_toolkits.mplot3d import Axes3D  
import numpy as np  
filename=input("Введите путь: ")  
rand\_state = 10  
pl\_ind = np.arange(3)+131  
color\_map = plt.cm.get\_cmap('RdYlBu')  
color\_map\_discrete = matplotlib.colors.LinearSegmentedColormap.from\_list("", ["red","cyan","magenta","blue"])  
labels = [1,2,3]  
dataset\_x=[]  
dataset\_y=[]  
fig = plt.figure(figsize=(18,5))  
for label, plt\_ind in zip(labels, pl\_ind):  
 x, y = dt.make\_multilabel\_classification(n\_samples=20,  
 n\_features=3,  
 n\_labels=label,  
 n\_classes=3,  
 random\_state=rand\_state)  
 dataset\_x.append(x)  
 dataset\_y.append(y)  
 target = np.sum(y\*[1,1,1], axis=1)  
 ax=fig.add\_subplot(plt\_ind, projection = '3d',facecolor='black')  
 ax.scatter (x[:,0],x[:,1],x[:,2],c=target,s=40, cmap=color\_map)  
 plt.title('n\_labels: ' + str(label))  
ines = pd.MultiIndex.from\_product([['label1','label2','label3'], np.arange(20)])  
df\_x = pd.DataFrame(np.concatenate(dataset\_x),index=ines, columns=['x','y','z'])  
df\_y = pd.DataFrame(np.concatenate(dataset\_y),index=ines, columns=['class1','class2','class3'])  
with pd.ExcelWriter(filename) as writer:  
 df\_x.to\_excel(writer)  
 df\_y.to\_excel(writer, startcol=5)  
plt.suptitle('make\_multilabel\_classification() With Different n\_labels Values', fontsize=20)  
plt.show()

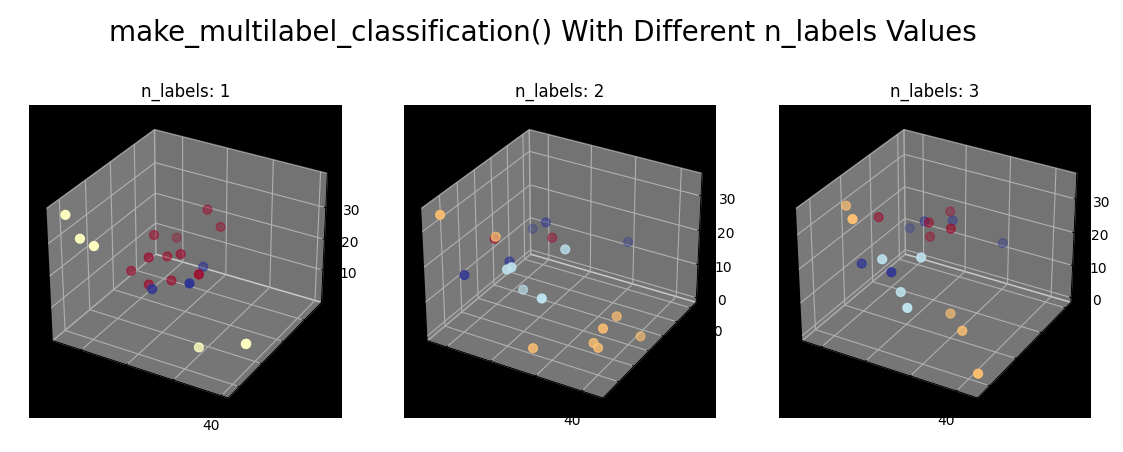


Рисунок 47 – график разброса объектов по классам

Так как изначально было задано три входных переменных, то можно построить трехмерный график разброса.

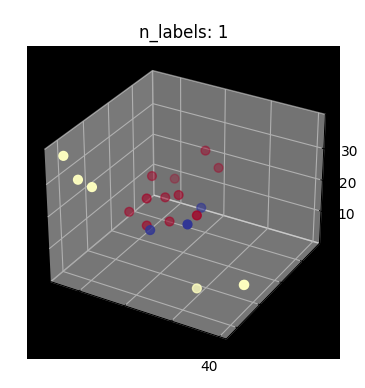


Рисунок 48 – график разброса для одного лейбла

На графике видно, присутствует всего 3 цвета. Это означает, что все объекты, причисленные к разным классам, имеют принадлежность только к одному классу, так как имеем возможность назначения только одного лейбла к объекту.

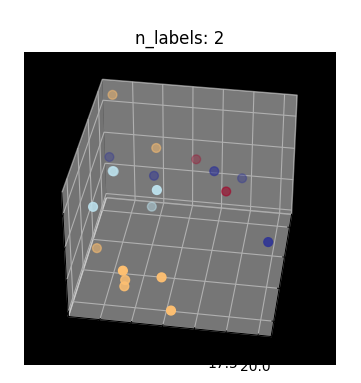


Рисунок 49 – график разброса для двух лейбов

Здесь мы наблюдаем уже другую картину: некоторым объектам присвоено два лейбла, от этого цвет выглядит смешанным.

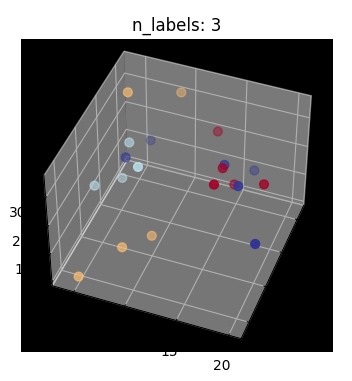


Рисунок 50 – график разброса для трёх лейблов

Здесь наблюдаем похожу картину, однако генерация данных произошла иначе, расположение точек изменилось, группирование классов произошло иначе.

Извлечем данные и посмотрим как произошло формирование.

Рисунок 51 – гистограмма принадлежностей к классам

Это гистограмма соотношения количества принадлежностей классам.

При возможности назначения одного лейбла: в большинстве случаев данные были назначены как не принадлежащие ни к одному классу (11 случаев), далее, как данные принадлежащие к одному классу (6 случаев), и как данные принадлежащие к 2-м классам (3 случая). Такое распределение произошло из-за количества классов – их 3, соответственно, задача функции распределить данные по всем классам, графики не отображают классификацию, они отображают лейбелизацию, поэтому мы не видим четкого разделения скоплений.

При возможности назначение 2х лейблов: не назначены (2 случая), 1 класс (8 случаев), 2 класса (5 случаев), 3 класса (5 случаев).

При возможности назначения 3х лейблов: не назначены (5 случаев), 1 класс (5 случаев), 2 класса (4 случая), 3 класса (6 случаев).

Это не соответствует правилу назначения лейблов. Однако можно рассмотреть каким образом формируется принадлежность.

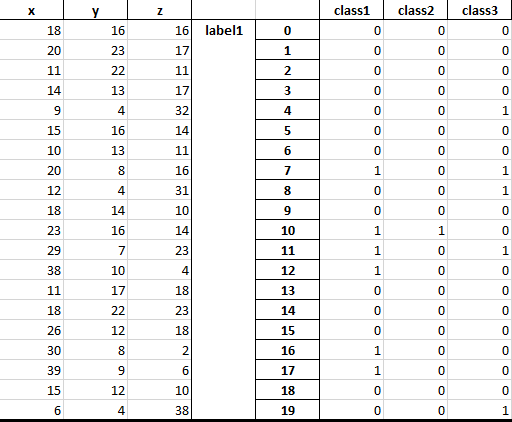


Рисунок 52 – таблица сформированных данных

В строках 0,1,2,3,5,6,9,13,14,15,18 разница между переменными не велика, они все примерно находятся в одном окружении, разница между не ними не больше 10-12 единиц. В строках 4,8,12,16,17,19 одна переменная сильно больше двух других, сумма двух меньших переменных не превысит значения большей переменной. В строках 7,10,11 стоят принадлежности к 2-м классам, строка 10 похожа на строки с нулевой принадлежностью, но строки 7 и 10 не похожи ни на один другой вариант.

Метод make\_multilabel\_classification() подходит для нестрогого формирования данных, в целом справляется со своей задачей, однако при извлечении данных нужно учитывать, что при назначении классов может оказаться, что сформированные данные не принадлежат ни к одному классу. При лейбелизации мы самостоятельно назначаем величину случайности формирования данных.

Часть 6. Создание синтетических данных на основе задачи кластеризации

Кластеризация, она же кластерный анализ, статическая процедура над данными, выполняющая выборку объектов, упорядочивающая их в группы по некоторым сходным признакам. Широко применима в обучении «без учителя».[[6]](#footnote-6)

Данные для кластеризации могут быть:

* Признаковыми описаниями объектов. Задача об ирисах Фишера содержит в себе признаковые описания объектов: длина, ширина и толщина лепестков – признаки объекта.
* Матрицей расстояний. При кластеризации объекты могут быть описаны метриками в метрическом пространстве[[7]](#footnote-7).
* Матрицей сходства. Она же матрица мер конвергенции, содержит меры сходства между объектами, обычно выражающиеся множеством [0,1]. Оперирует выражениями «больше», «меньше», «равно».

Sklearn позволяет сделать кластеризацию методом make\_blobs().

В это функции было задано 2 переменных на единицу данных выборки, назначено 4 центра, вокруг которых будет происходить формирование классов, величина разброса данных относительно центра, а также извлечение координат центров.

Код программы:

import numpy as np  
  
import pandas as pd  
import matplotlib.colors  
import matplotlib.pyplot as plt  
from openpyxl.workbook import Workbook  
from mpl\_toolkits.mplot3d import axes3d  
  
import sklearn.datasets as dt  
  
from sklearn.neighbors import KernelDensity  
from sklearn.model\_selection import GridSearchCV  
seed = 10  
rand\_state = 10  
filename = input('Введите путь к файлу: ')  
# Define the color maps for plots  
color\_map = plt.cm.get\_cmap('RdYlBu')  
color\_map\_discrete = matplotlib.colors.LinearSegmentedColormap.from\_list("", ["red","cyan","magenta","blue"])  
fig, ax = plt.subplots(nrows=1, ncols=4, figsize=(16, 5))  
plt\_ind\_list = np.arange(4) + 231  
dataset\_x=[]  
dataset\_lab=[]  
for std, plt\_ind in zip([0.1, 1, 5, 10], plt\_ind\_list):  
 x, label,centers = dt.make\_blobs(n\_features=2,  
 centers=4,  
 cluster\_std=std,  
 random\_state=rand\_state,  
 return\_centers=True)  
 dataset\_x.append(x)  
 dataset\_lab.append(label)  
 plt.subplot(plt\_ind)  
 my\_scatter\_plot = plt.scatter(x[:, 0],  
 x[:, 1],  
 c=label,  
 vmin=min(label),  
 vmax=max(label),  
 cmap=color\_map\_discrete)  
 plt.title('cluster\_std: ' + str(std))  
  
nds\_x = np.concatenate(dataset\_x)  
nds\_l = np.concatenate(dataset\_lab)  
ines\_x = pd.MultiIndex.from\_product([[0.1, 1, 5, 10],np.arange(len(dataset\_x[0]))])  
cols = pd.MultiIndex.from\_product([['centers'],['x','y']])  
df\_x =pd.DataFrame(nds\_x,index=ines\_x,columns=['x','y'])  
df\_l =pd.DataFrame(nds\_l,index=ines\_x,columns=['label'])  
df\_c = pd.DataFrame(centers, index=np.arange(len(centers)), columns=cols)  
with pd.ExcelWriter(filename) as writer:  
 df\_x.to\_excel(writer)  
 df\_l.to\_excel(writer,startcol=4)  
 df\_c.to\_excel(writer, startcol=8)  
fig.subplots\_adjust(hspace=0.3, wspace=.3)  
plt.suptitle('make\_blobs() With Different cluster\_std Values', fontsize=20)  
plt.show()

Результат работы программы:

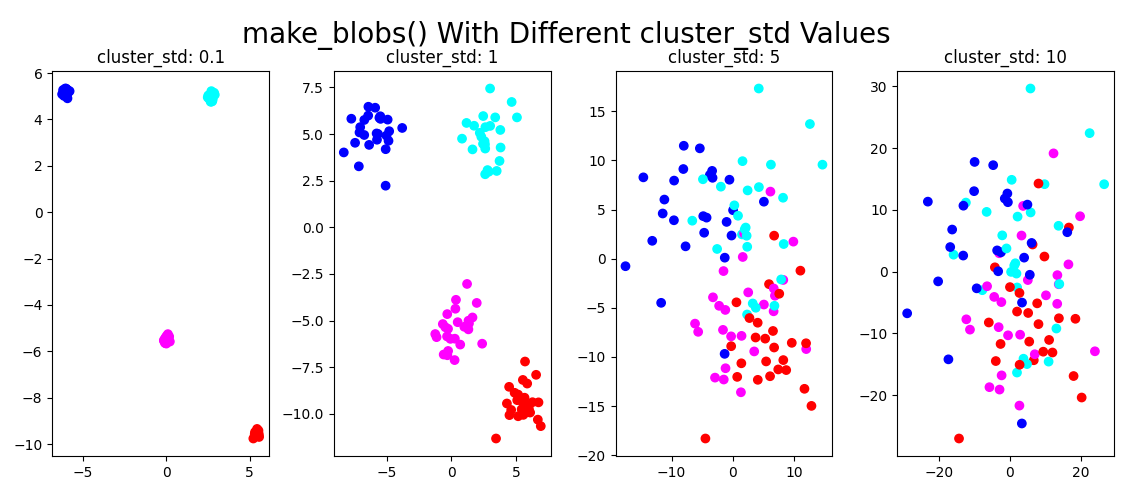


Рисунок 53 – сформированные кластеры с различными величинами разброса

Извлеченные данные:

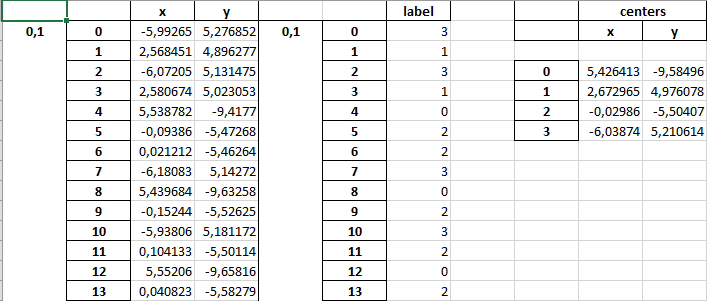


Рисунок 54 – извлеченные данные

Мы получили координаты, которые соответствуют каждой точке на представленных выше картах кластеров.

Проведем проверку соответствует ли полученные данные действительности.

Из полученных данных было выполнено распределение по кластерам и степени разброса на разные листы, в результате получили структурированные данные:

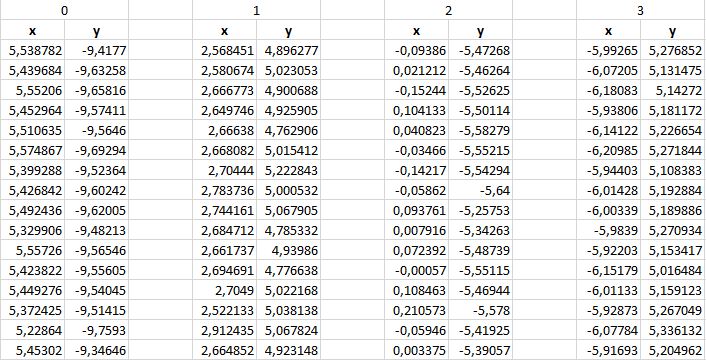


Рисунок 55 – реструктуризированные данные

После чего были заново построены карты кластеризации:

Рисунок 56 – карта кластеризации при величине разброса 0,1

Рисунок 57 – карта кластеризации при величине разброса 1

Рисунок 58 – карта кластеризации при величине разброса 5

Рисунок 59 – карта кластеризации при величине разброса 10

Карты кластеризации, полученные в результате моделирования в Excel не отличаются от карт, смоделированных make\_blobs(), следовательно, метод достаточно достоверный.

Рассмотрим насколько точным является разброс вокруг центров кластеров:

Рисунок 60 – отклонение от разброса величиной 0,1

Рисунок 61 – отклонение от разброса величиной 1

Рисунок 62 – отклонение от разброса величиной 5

Рисунок 63 – отклонение от разброса величиной 10

На всех четырех графиках разброса наблюдается одно и тоже расположение точек, это означает, что генерация данных для классов происходит по фиксированному сценарию. Значения точек показывают их расстояние от центра, который находится в точке (0;0).

Рассчитаем максимальные отклонения от центра, чтобы определить какой кластер является наименьшим, а какой наибольшим.

*- Кластер 0. Самые дальние точки: (3,24); (12,17); (15,11).*

*Вывод: max(cat0) = cat01=24,19*

*- Кластер 1. Самые дальние точки: (4,25); (20,17); (24,9).*

*Вывод: max(cat1) = cat12=26,25*

*- Кластер 2. Самые дальние точки: (13,25); (20,15); (24,7).*

*Вывод: max(cat2) = cat21=28,18*

*- Кластер 3. Самые дальние точки: (9,30); (23,12).*

*Вывод: max(cat3) = cat31=31,32*

Наименьшее отклонение имеет кластер 0, в нем точки образуются «плотнее» к центру, из этого можно сделать вывод, что это кластер с самыми маленькими метриками расстояний. Максимальное расстояние от центра до дальней точки – 24,19.

Следующий по размеру – кластер 1. Максимальное расстояние от центра до дальней точки – 26,25.

Следующий по размеру – кластер 2. Максимальное расстояние от центра до дальней точки – 28,18.

Наибольший по размеру оказался кластер 3, у него самое большое отклонение от центра – 31,32.

Расчеты производились по масштабам графика разброса величины отклонения от 10.

Часть 7. Создание синтетических данных на основе задачи о спектральной кластеризации

Спектральная кластеризация – один из способов кластеризации данных. Применим в том случае, когда данные неграфовые. Метод работает с корнями графов, используется для идентификации узлов в сообществе графов на основе соединяющих ребер.

Сначала формируется матрица с собственными векторами и собственными значениями, которые можно преобразовать в граф со смежными вершинами.

Для формирования кругов используется метод k-средних.

Алгоритм представляет собой версию [EM-алгоритма](https://ru.wikipedia.org/wiki/EM-%D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC), применяемого также для разделения смеси [гауссиан](https://ru.wikipedia.org/wiki/%D0%93%D0%B0%D1%83%D1%81%D1%81%D0%B8%D0%B0%D0%BD%D0%B0). Он разбивает [множество](https://ru.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B6%D0%B5%D1%81%D1%82%D0%B2%D0%BE) элементов [векторного пространства](https://ru.wikipedia.org/wiki/%D0%9B%D0%B8%D0%BD%D0%B5%D0%B9%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D1%81%D1%82%D1%80%D0%B0%D0%BD%D1%81%D1%82%D0%B2%D0%BE) на заранее известное число кластеров *k*.

Основная идея заключается в том, что на каждой итерации перевычисляется [центр масс](https://ru.wikipedia.org/wiki/%D0%A6%D0%B5%D0%BD%D1%82%D1%80_%D0%BC%D0%B0%D1%81%D1%81) для каждого кластера, полученного на предыдущем шаге, затем векторы разбиваются на кластеры вновь в соответствии с тем, какой из новых центров оказался ближе по выбранной метрике.

Алгоритм завершается, когда на какой-то итерации не происходит изменения внутрикластерного расстояния. Это происходит за конечное число итераций, так как количество возможных разбиений конечного множества конечно, а на каждом шаге суммарное квадратичное отклонение *V* уменьшается, поэтому зацикливание невозможно.

Итак, код программы:

import numpy as np  
  
import pandas as pd  
import matplotlib.colors  
import matplotlib.pyplot as plt  
from openpyxl.workbook import Workbook  
from mpl\_toolkits.mplot3d import axes3d  
  
import sklearn.datasets as dt  
  
from sklearn.neighbors import KernelDensity  
from sklearn.model\_selection import GridSearchCV  
seed = 11  
rand\_state = 11  
filename = input('Введите путь к файлу: ')  
# Define the color maps for plots  
color\_map = plt.cm.get\_cmap('RdYlBu')  
color\_map\_discrete = matplotlib.colors.LinearSegmentedColormap.from\_list("", ["red","cyan","magenta","blue"])  
fig, ax = plt.subplots(nrows=1, ncols=4, figsize=(16, 5))  
plt\_ind\_list = np.arange(4) + 141  
dataset\_x=[]  
dataset\_lab=[]  
for noise, plt\_ind in zip([0, 0.1, 0.5, 1], plt\_ind\_list):  
 x, label = dt.make\_circles(noise=noise, random\_state=rand\_state)  
 dataset\_x.append(x)  
 dataset\_lab.append(label)  
 plt.subplot(plt\_ind)  
 my\_scatter\_plot = plt.scatter(x[:, 0],  
 x[:, 1],  
 c=label,  
 vmin=min(label),  
 vmax=max(label),  
 cmap=color\_map\_discrete)  
 plt.title('noise: ' + str(noise))  
nds\_x = np.concatenate(dataset\_x)  
nds\_l = np.concatenate(dataset\_lab)  
ines\_x = pd.MultiIndex.from\_product([[0, 0.1, 1, 2],np.arange(len(dataset\_x[0]))])  
cols\_x = pd.MultiIndex.from\_product([['x'],[1,2]])  
df\_x =pd.DataFrame(nds\_x,index=ines\_x,columns=cols\_x)  
df\_l =pd.DataFrame(nds\_l,index=ines\_x,columns=['label'])  
with pd.ExcelWriter(filename) as writer:  
 df\_x.to\_excel(writer)  
 df\_l.to\_excel(writer,startcol=4,startrow=2)  
fig.subplots\_adjust(hspace=0.3, wspace=.3)  
plt.suptitle('make\_circles() With Different Noise Levels', fontsize=20)  
plt.show()

Результат работы программы:

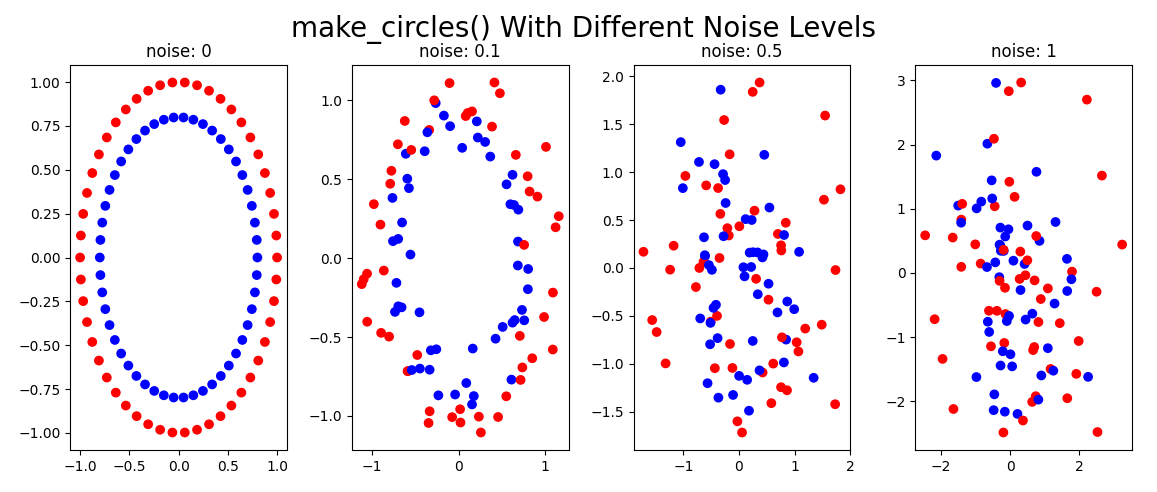


Рисунок 64 – результат работы программы

Извлеченные данные:

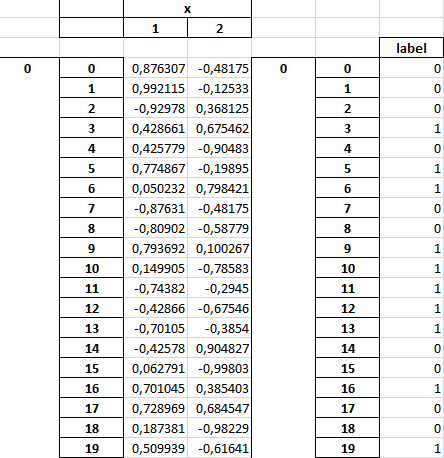


Рисунок 65 – извлеченные данные

Результат моделирования в Excel:

Рисунок 66 – моделирование с нулевым шумом

Рисунок 67 – моделирование с шумом 0,1

Рисунок 68 – моделирование с шумом 0,5

Рисунок 69 – моделирование с шумом 1

Часть 8. Пример «зашумления» изображений

Для усложнения задачи распознавания и повышения качества сети распознавания используют генерацию новых изображений с шумом. В этой части приведен пример генерации «зашумленных лиц».

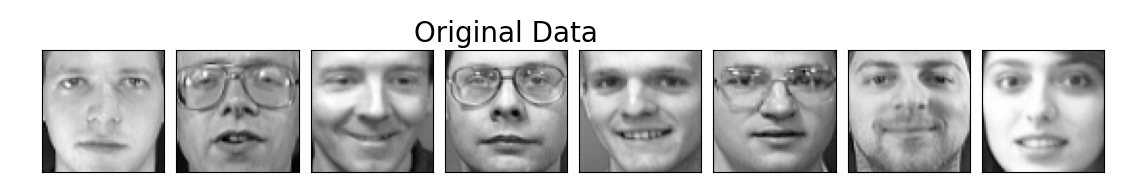


Рисунок 70 – оригинальные лица

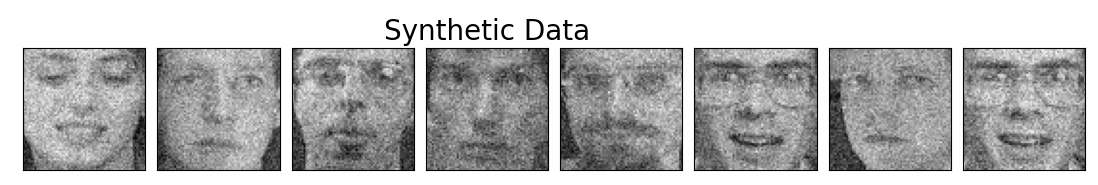


Рисунок 71 – зашумленные лица

Код программы:

import numpy as np  
  
import pandas as pd  
import matplotlib.colors  
import matplotlib.pyplot as plt  
from openpyxl.workbook import Workbook  
from mpl\_toolkits.mplot3d import axes3d  
  
import sklearn.datasets as dt  
  
from sklearn.neighbors import KernelDensity  
from sklearn.model\_selection import GridSearchCV  
seed = 11  
rand\_state = 11  
  
# Define the color maps for plots  
color\_map = plt.cm.get\_cmap('RdYlBu')  
color\_map\_discrete = matplotlib.colors.LinearSegmentedColormap.from\_list("", ["red","cyan","magenta","blue"])  
# Fetch the dataset and store in X  
faces = dt.fetch\_olivetti\_faces()  
X= faces.data  
  
# Fit a kernel density model using GridSearchCV to determine the best parameter for bandwidth  
bandwidth\_params = {'bandwidth': np.arange(0.01,1,0.05)}  
grid\_search = GridSearchCV(KernelDensity(), bandwidth\_params)  
grid\_search.fit(X)  
kde = grid\_search.best\_estimator\_  
  
# Generate/sample 8 new faces from this dataset  
new\_faces = kde.sample(8, random\_state=rand\_state)  
  
# Show a sample of 8 original face images and 8 generated faces derived from the faces dataset  
fig,ax = plt.subplots(nrows=2, ncols=8,figsize=(18,6),subplot\_kw=dict(xticks=[], yticks=[]))  
for i in np.arange(8):  
 ax[0,i].imshow(X[10\*i,:].reshape(64,64),cmap=plt.cm.gray)  
 ax[1,i].imshow(new\_faces[i,:].reshape(64,64),cmap=plt.cm.gray)  
ax[0,3].set\_title('Original Data',fontsize=20)  
ax[1,3].set\_title('Synthetic Data',fontsize=20)  
fig.subplots\_adjust(wspace=.1)  
plt.show()

Заключение

В данной курсовой работы были рассмотрены несколько способов генерации данных, каждый из них подходит под конкретную задачу, подбирать решение для которой нужно с учётом возможностей языка программирования, имеющихся ресурсов и результата, который мы ожидаем получить, ориентируясь на исходные данные. Рекомендую рассмотреть весь функционал библиотеки scklearn.datasets, в ней можно найти дополнительные способы генерации синтетических данных, работающие на основе других статистических распределений и задач теории вероятности.

Эта работа получилась более обзорной, чем исследовательской. К работе прилагаются программные коды, написанные в среде PyCharm 2021.1 на языках Python 3.8.3 и Python 3.9, а также таблицы Excel 2013, содержащие отчеты по аналитике.

Рекомендую к прочтению приведенные ниже статьи, они позволят лучше понять изложенный в данной курсовой материал.

Список использованных источников

1. Руководство пользователя по библиотеке Sklearn.datasets. Режим обращения - <https://scikit-learn.org/stable/modules/classes.html#module-sklearn.datasets>
2. Generating Synthetic Data with Numpy and Scikit-Learn by Mehreen Saeed. Режим обращения - https://stackabuse.com/generating-synthetic-data-with-numpy-and-scikit-learn/
3. Статья: Типичные распределения вероятности: шпаргалка data scientist-а by [Sean Owen](https://blog.cloudera.com/blog/2015/12/common-probability-distributions-the-data-scientists-crib-sheet/). Режим обращения - https://habr.com/ru/post/331060/
4. Linear Regression for Machine Learning by [Jason Brownlee](https://machinelearningmastery.com/author/jasonb/). Режим обращения - https://machinelearningmastery.com/linear-regression-for-machine-learning/
5. Статья: Критерий Фридмана. Режим обращения - <http://www.machinelearning.ru/wiki/index.php?title=%D0%9A%D1%80%D0%B8%D1%82%D0%B5%D1%80%D0%B8%D0%B9_%D0%A4%D1%80%D0%B8%D0%B4%D0%BC%D0%B0%D0%BD%D0%B0>
6. Статья: Классификация. Режим обращения - <http://www.machinelearning.ru/wiki/index.php?title=%D0%9A%D0%BB%D0%B0%D1%81%D1%81%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D1%8F>
7. Deep dive into multi-label classification..! (With detailed Case Study). Toxic-comments classification. By Kartik Nooney. Режим обращения - <https://towardsdatascience.com/journey-to-the-center-of-multi-label-classification-384c40229bff>
8. Обзор алгоритмов кластеризации данных. Режим обращения - <https://habr.com/ru/post/101338/>
9. Аугментация на лету — важный инструмент в обучении нейронных сетей. Режим обращения - https://habr.com/ru/company/smartengines/blog/499566/

1. Дистилляция данных – собирательное название методов сокращения обучающей выборки путем создания искусственных объектов, называемых «представители класса» в которых агрегированы ключевые общие признаки. На основе «представителей класса» строятся алгоритмы машинного обучения одинаково эффективные как для дистиллированной выборки, так и для полноразмерной. [↑](#footnote-ref-1)
2. Адекватность поведения модели – соответствие выдаваемых моделью результатов, ожидаемых разработчиком. [↑](#footnote-ref-2)
3. В бесконечно возрастающие функции входят все функции, которые имеют бесконечно малые приближения в верхнем пределе, наиболее знакомые – тангенсоиды (школьный курс) [↑](#footnote-ref-3)
4. Монотонность функции – свойство функции (только возрастает или только убывает) [↑](#footnote-ref-4)
5. Рекомендательные системы – системы, построенные на алгоритмах выявления предпочтений пользователя, платформы Netflix, Яндекс.Музыка, ivi.ru и другие, содержащие развлекательный контент, подбирают видео и музыку для пользователя по наиболее встречающимся лейблам. [↑](#footnote-ref-5)
6. Обучение «без учителя» - один из видов машинного обучения, в ходе обучения модель обучается спонтанно, самостоятельно пытаясь определить взаимосвязи, структуру и сделать выводы по входной выборке данных. Наиболее знаменитые модели с самообучением: самоорганизующаяся сеть Кохонена, автоассоциатор, глубокая сеть доверия. Применяется в распознавании образов и лингвистическом анализе. [↑](#footnote-ref-6)
7. Метрическое пространство – непустое множество, в котором описывается расстояние между двумя объектами, именующееся метрикой. Метрики могут быть дискретными (множество [0;1]), в евклидовом пространстве (abs(y-x)), множество вершин связного графа G (поиск пути по графу: алгоритмы Дейкстры, Форда-Беллмана и другие, «жадные алгоритмы» для построения минимально сложного графа). Возможны так же другие пространства, в которых можно свести поиск расстояния до функции d(x,y)=||y-x||. [↑](#footnote-ref-7)