**НИТУ «МИСиС»**

Институт ИТКН

**Кафедра инженерной кибернетики**

Направление подготовки: 01.03.04 прикладная математика Квалификация (степень): бакалавр

**КУРСОВАЯ РАБОТА**

**учебная дисциплина**

**«Нейронные сети»**

**VII семестр 2020 – 2021 у.г.**

**Учащийся: *Терентьева П.А.***

**Группа: БПМ-17-1**

**Проверил: *Кондыбаева А.Б.***

**Оценка:**

**Дата защиты:**

**Москва 2020**

Оглавление

[**1.** **Вступление** 3](#_Toc58687179)

[**2.** **Стандартные методы мониторинга работоспособности данных** 4](#_Toc58687180)

[**3.** **Главные задачи предварительной обработки данных** 5](#_Toc58687181)

[**3.1 Способы обработки пропущенных значений** 5](#_Toc58687182)

[**3.2** **Способы нормализации данных** 5](#_Toc58687183)

[**3.3** **Способы дискретизации данных** 6](#_Toc58687184)

[**3.4** **Способы сокращения объема данных** 6](#_Toc58687185)

[**3.5** **Способы очистки данных** 6](#_Toc58687186)

[**4.** **Требования к данным** 8](#_Toc58687187)

[**5.** **Формирование датасетов** 9](#_Toc58687188)

[**5.1 Определение датасета** 9](#_Toc58687189)

[**5.2 Особенности формирования датасетов** 9](#_Toc58687190)

[**6.** **Использование библиотек Python для подготовки данных для ИНС** 10](#_Toc58687191)

[**7.** **Список использованной литературы** 14](#_Toc58687192)

[**ПРИЛОЖЕНИЕ А. Код программы** 15](#_Toc58687193)

[**ПРИЛОЖЕНИЕ Б. Ссылка на Google colab с кодом программы** 18](#_Toc58687194)

# **Вступление**

Работу по созданию модели машинного обучения можно разделить на две части:

* подготовка входных и выходные данных для экспериментов;
* создание, обучение, тестирование и оптимизация параметров модели.

Первый этап занимает большую часть времени реализации проекта: по данным из некоторых источников, около 70%. При этом от его результатов на 80% зависит результат работы модели. Именно поэтому уделяется большое внимание процессу подготовки данных для искусственной нейронной сети.

Предварительная обработка и очистка данных — это важные задачи, которые необходимо выполнить, прежде чем набор данных можно будет использовать для обучения модели. Необработанные данные зачастую искажены и ненадежны, и в них могут быть пропущены значения. Использование таких данных при моделировании может приводить к неверным результатам. Эти задачи являются частью процесса обработки и анализа данных группы и обычно подразумевают первоначальное изучение набора данных, используемого для определения и планирования необходимой предварительной обработки.

Реальные данные собираются для последующей обработки из разных источников и процессов. Они могут содержать ошибки и повреждения, негативно влияющие на качество набора данных. Вот какими могут быть типичные проблемы с качеством данных:

* Неполнота: данные не содержат атрибутов, или в них пропущены значения.
* Шум: данные содержат ошибочные записи или выбросы.
* Несогласованность: данные содержат конфликтующие между собой записи или расхождения.

Качественные данные — это необходимое условие для создания качественных моделей прогнозирования. Чтобы избежать появления ситуации «мусор на входе, мусор на выходе» и повысить качество данных и, как следствие, эффективность модели, необходимо провести мониторинг работоспособности данных, как можно раньше обнаружить проблемы и решить, какие действия по предварительной обработке и очистке данных необходимы.

# **Стандартные методы мониторинга работоспособности данных**

Для проверки качества данных, обычно оценивают следующие показатели:

* Число записей.
* Количество атрибутов (или компонентов).
* Типы данных атрибута (номинальный, порядковый номер или непрерывный).
* Число отсутствующих значений.
* Правильно сформированные данные.

Если данные имеют формат TSV или CSV, следует проверить правильность разделения столбцов и строк соответствующими разделителями.

Если же данные имеют формат HTML или XML, важно проверить, что формат данных соответствует надлежащим стандартам.

Для извлечения структурированной информации из частично структурированных или неструктурированных данных также может потребоваться синтаксический анализ.

* Непоследовательные записи данных.

Важно следить за допустимостью диапазона значений.

При обнаружении проблем с данными необходимо выполнить действия по обработке , что часто включает в себя очистку отсутствующих значений, нормализацию данных, дискретизация, обработку текста для удаления и/или замены внедренных символов, которые могут влиять на выравнивание данных, смешанные типы данных в общих полях и т. д.

# **Главные задачи предварительной обработки данных**

В качестве основных задач предварительной обработки данных выделяют следующие:

* **Очистка данных**: заполнение отсутствующих значений, обнаружение удаление шума данных и выбросов.
* **Преобразование данных**: нормализация данных для уменьшения размеров и шума.
* **Уплотнение данных** — создание выборки данных или атрибутов для упрощения обработки данных.
* **Дискретизация данных** — преобразование непрерывных атрибутов в категориальные, чтобы проще было использовать некоторые методы машинного обучения.
* **Очистка текста**: удалите внедренные символы, которые могут вызвать неправильное выравнивание данных, например внедренные вкладки в файле данных с разделителями-табуляцией, внедренные новые строки, которые могут нарушить работу записей, например.

## **3.1 Способы обработки пропущенных значений**

Первым делом при работе с пропущенными значениями следует определить причину их появления в данных, что поможет решить проблему. Существует несколько методов обработки пропущенных значений:

* **Удаление**: удаление записей с пропущенными значениями.
* **Фиктивная подстановка** — замена пропущенных значений фиктивными, например подстановка значения *unknown* (неизвестно) вместо категориальных или значения 0 вместо чисел.
* **Подстановка среднего значения**: пропущенные числовые данные можно заменить средним значением.
* **Подстановка часто используемого элемента**: пропущенные категориальные значения можно заменить наиболее часто используемым элементом.
* **Подстановка по регрессии**: использование регрессионного метода для замены пропущенных значений регрессионными.

## **Способы нормализации данных**

Нормализация данных позволяет масштабировать числовые значения в указанный диапазон. Ниже представлены распространенные методы нормализации данных.

* **Нормализация по методу минимакса**: линейное преобразование данных в диапазоне, например, от 0 до 1, где минимальное и максимальное масштабируемые значения соответствуют 0 и 1 соответственно.
* **Нормализация по Z-показателю**: масштабирование данных на основе среднего значения и стандартного отклонения: деление разницы между данными и средним значением на стандартное отклонение.
* **Десятичное масштабирование**: масштабирование данных путем удаления десятичного разделителя значения атрибута.

## **Способы дискретизации данных**

Данные можно дискретизировать, преобразовав непрерывные значения в номинальные атрибуты или интервалы. Это можно сделать несколькими способами.

* **Группирование равной ширины**: разделение диапазона всех возможных значений атрибута в группы (N) одинакового размера с последующим присвоением значений, относящихся к ячейке с соответствующим номером.
* **Группирование равной высоты**: разделение всех возможных значений атрибута в группы (N), содержащие одинаковое количество экземпляров, с последующим присвоением значений, относящихся к ячейке с соответствующим номером.

## **Способы сокращения объема данных**

Существуют различные методы, с помощью которых вы можете уменьшить размер данных для упрощения обработки данных. В зависимости от размера данных и домена вы можете применить такие методы:

* **Выборка записей**: создание выборки записей данных и выбор репрезентативного подмножества из общего набора данных.
* **Выборка атрибутов**: выбор в данных набора важнейших атрибутов.
* **Агрегирование**: разделение данных на группы и хранение числовых значений для каждой группы. Например, для уменьшения размера данных вы можете агрегировать числа, обозначающие ежедневный доход сети ресторанов за последние 20 лет, так, чтобы указывался ежемесячный доход.

## **Способы очистки данных**

**Текстовые поля в табличных данных** могут содержать символы, влияющие на выравнивание столбцов и (или) границы записи. Например, встроенные вкладки в файле с разделителями-табуляциями приводят к неправильному выравниванию столбца, а внедренные символы новой строки приведут к разрыву строк записи. Неправильная обработка кодировки текста при записи или чтении текста приводит к утере информации, непреднамеренному вводу нечитаемых символов (например, значений NULL) и может также повлиять на синтаксический анализ текста. Чтобы очистить текстовые поля, исправить выравнивание и извлечь структурированные текстовые данные из неструктурированных или полу-структурированных, могут потребоваться тщательные разбор и редактирование текста.

**Функция просмотра данных** позволяет ознакомиться с данными заблаговременно. Это поможет вам выявить те или иные проблемы с данными и применить соответствующие методы для решения этих проблем. Важно понимать, что породило проблемы, как они могли появиться. Этот процесс также помогает выбрать действия по обработке данных, которые необходимо предпринять для их устранения. Определение окончательных вариантов использования и персонажей можно также использовать для определения приоритетов при обработке данных.

# **Требования к данным**

Выделяют три основных требования к датасетам:

* Достаточность

Число обучающих примеров должно быть достаточным для обучения. Для нейронной сети необходимо, чтобы число обучающих примеров было в несколько раз больше, чем число весов межнейронных связей, в противном случае модель может не приобрести способности к обобщению. Кроме этого, размер выборки должен быть достаточным для формирования обучающего и тестового множеств.

* Равномерность представления классов

Примеры различных классов должны быть представлены в обучающей выборке примерно в одинаковых пропорциях. Если один из классов будет преобладать, то это может привести к «перекосу» в процессе обучения модели, и данный класс будет определен моделью как наиболее вероятный для любых новых наблюдений.

* Разнообразие

Большое число разнообразных комбинаций вход-выход в обучающих примерах. Способность ИНС к обобщению не будет достигнута, если число примеров достаточное, но все они одинаковые, то есть представляющие лишь часть классов, характерных для исходного множества.

# **Формирование датасетов**

## **5.1 Определение датасета**

Датасет для машинного обучения – это обработанная и структурированная информация в табличном виде. Строки таблицы называют объектами, а столбцы – признаками. Различают 2 вида признаков:

- независимые переменные – предикторы;

- зависимые переменные – целевые признаки, которые вычисляются на основе одного или нескольких предикторов.

Для каждого этапа машинного обучения необходим свой набор данных:

- для непосредственного обучения модели нужна обучающая выборка, по которой производится настройка (оптимизация параметров) алгоритма;

- для оценки качества модели используется тестовая выборка, которая в идеальном случае не должна зависеть от обучающей;

- для выбора наилучшей модели машинного обучения понадобится проверочная (валидационная) выборка, которая также не должна пересекаться с обучающей.

## **5.2 Особенности формирования датасетов**

Методы формирования обучающих и оценочных выборок зависят от класса задачи, решаемой с помощью машинного обучения.

Для задач классификации данные следует разделить так, чтобы в полученных наборах численное соотношение объектов разных классов было таким же, как в исходной генеральной совокупности;

Для задач регрессионного анализа необходимо одинаковое распределение целевой переменной в полученных наборах, которые будут использоваться для обучения и контроля качества.

При соблюдении этих условий объемы обучающей и оценочных выборок могут существенно различаться. Например, размер валидационного датасета может составлять всего 10% генеральной совокупности. Главное в формировании выборок - не объединять обучающий датасет с оценочным, так как это грозит переобучением модели. В этом случае модель получит высокие оценки качества в процессе тренировки, но не покажет такого результата на реальных данных.

# **Использование библиотек Python для подготовки данных для ИНС**

Для обработки данных популярны следующие библиотеки: Pandas, Scikit-learn, а также Seaborn и Matplotlib для визуализации данных.

Библиотека pandas – это программная библиотека на языке Python для обработки и анализа данных.

Рассмотрим основные функции из этой библиотеки.

Для чтения файла формата csv используется функция **read\_csv(“название файла”)**. Использовать эту функцию удобно хотя бы потому, что она заменяет несколько строк кода стандартного Python, где надо решить, как хранить данные, затем открыть файл, прочитать его построчно, отделить значения друг от друга. В Pandas данные хранятся в датафреймах (от англ. dataframe). Датафреймы – это таблицы. Также, библиотека Pandas предоставляет еще одну новую для Python структуру данных – серии. Серия – это одномерный массив. По сути, колонки датафреймов это серии.

Чтобы посмотреть считанный файл используется функция **head().** Пример отображения датасета “survey.csv” приведен на рисунке 1. В качестве параметра можно указать количество строк, которые следует отобразить.

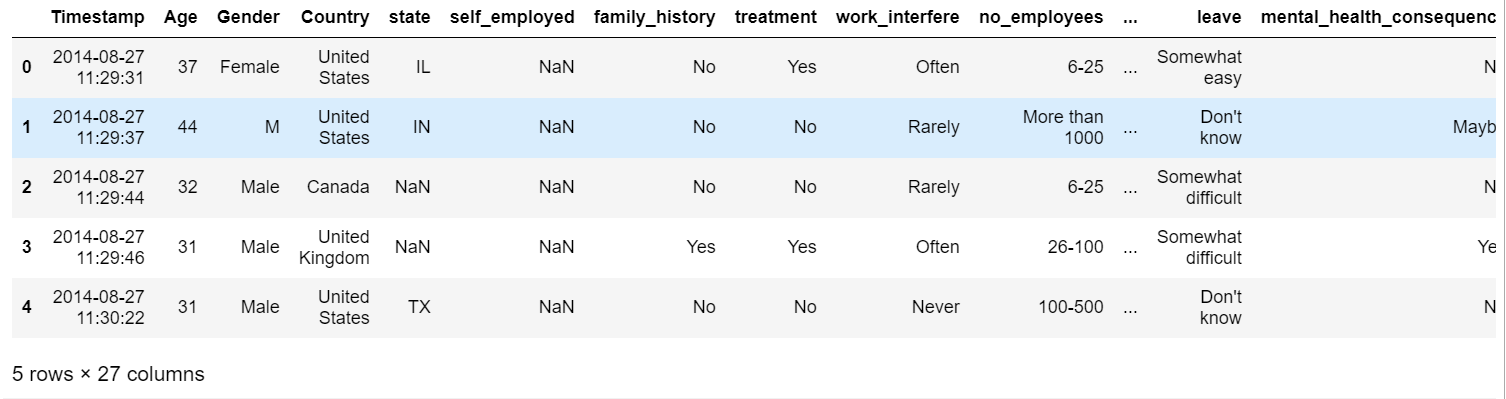


Рисунок 1 – результат df.head()

С помощью функции **shape()** можно узнать размерность датафрейма. Функция вернет количество строк и столбцов таблицы.

Более подробную информацию можно получить с помощью **info()**. Данная функция возвращает информацию о размере объекта, типах данных и использовании памяти. Пример использования info() для датасета «survey» приведен на рисунке 2.

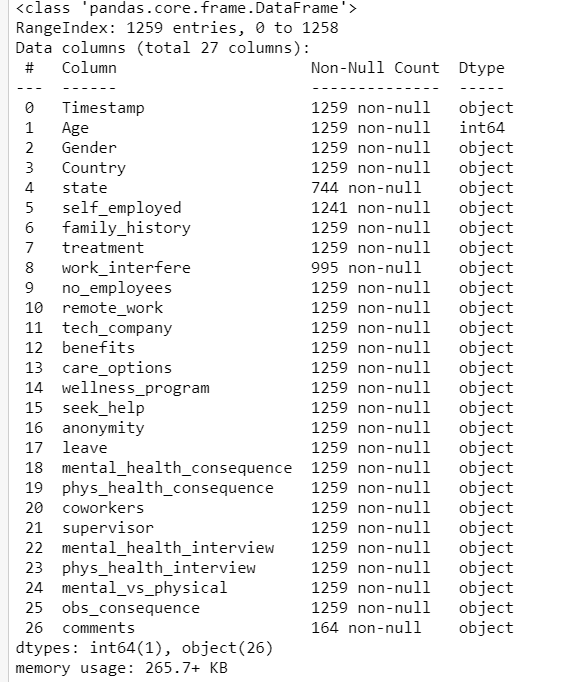


Рисунок 2 – результат отображения df.info()

Количество уникальных значений в каждом столбце можно узнать посредством **nunique()**.

Есть несколько подходов к обработке пропущенных значений. Один из них – это удаление строк, в которых есть пропущенные значения, однако этот путь может привести к потере важной информации. Для обработки пропущенных значений удобно использовать класс SimpleImputer библиотеки Scikit-learn. В параметрах объекта SimpleImputer следует указать параметр strategy, который будет определять алгоритм заполнения пропущенных значений. Доступны следующие опции:

- “mean”, заполняет пропущенное значение, используя среднее значение по каждому столбцу; может быть использован только с числовыми данными

- “median”, заполняет пропущенное значение, используя медиану по каждому столбцу; может быть использован только с числовыми данными

- “most\_frequent”, заполняет пропущенное значение, используя самое частое значение по каждому столбцу; может быть использован как для числовых, так и для строковых данных

- “constant”, заполняет пропущенное значение fill\_value (указывается в качестве дополнительного параметра); может быть использован как для числовых, так и для строковых данных.

Как было сказано ранее, на этапе подготовки данных часто используется нормализация данных. Библиотека Scikit-learn предоставляет несколько функций для нормализации данных:

- MinMaxScaler()

- StandardScaler()

- Normalizer()

- RobustScaler()

- QuantileTransformer()

- PowerTransformer()

Для работы с категориальными признаками используют алгоритмы кодирования. Простейший кодировщик sklearn.preprocessing.LabelEncoder сопоставляет каждой категории некоторое целое число. Этот метод сильно упрощает данные, фактически проецируя категориальный признак на вещественную прямую. Более того, появляются ложные интерпретации. Например, сумма некоторых признаков может давать другой признак, хотя это не свойство данных, а свойство выбранной кодировки. Существует простейший метод, лишенный этого недостатка, который называют one-hot-кодированием или dummy-кодированием. Для кодируемого категориального признака создаются N новых признаков, где N- число категорий. Каждый i-й новый признак – бинарный характеристический признак i-й категории. Этот способ кодировки реализован как в Scikit-Learn (OneHotEncoder), так и в Pandas (get\_dummies). Однако, эти функции работают с числовыми данными, если же, данные содержат строковые значения, удобнее использовать sklearn.preprocessing.LabelBinarizer.

Для разделения датасета на выборки для обучения и тестирования используется slkearn.model\_selection.train\_test\_split. В качестве обязательного параметра функции указывается массив, который следует разделить. Размер тестовой выборки и выборки для обучения можно не указывать, в этом случае размер тестовой выборки будет равен 0.25 от исходного массива.

При анализе датасета удобно использовать графики. Библиотеки Seaborn и Matplotlib предоставляют функции для визуализации данных. Благодаря этим библиотекам можно нарисовать такие виды графиков, как: линейные, гистограммы, круговые диаграммы, точечные графики и другие. Примеры визуализации данных с помощью графиков представлены на рисунках 3-5.

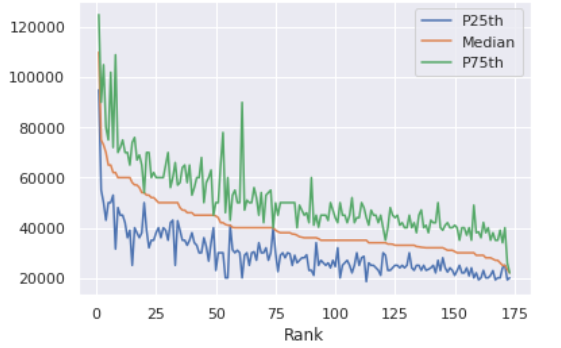


Рисунок 3 – Линейных график

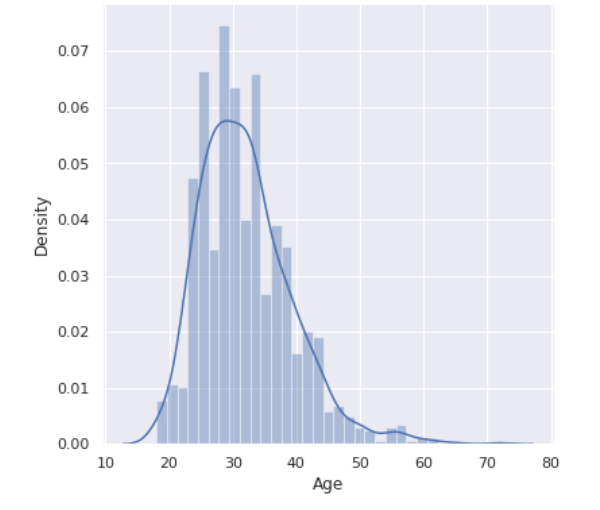


Рисунок 4 - Гистограмма

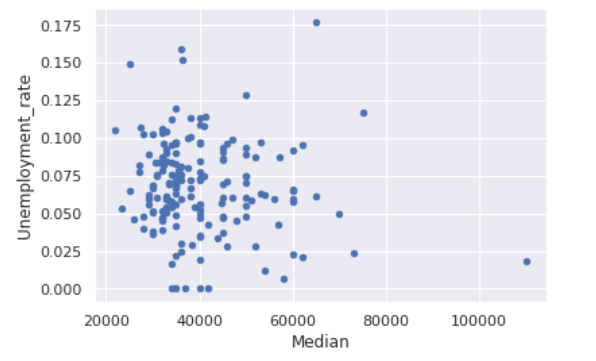


Рисунок 5 – Точечный график

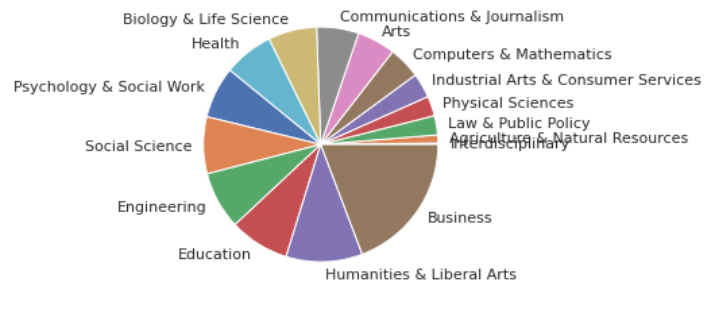


Рисунок 6 – Круговая диаграмма.

# **Список использованной литературы**

1. https://docs.microsoft.com/ru-ru/azure/machine-learning/team-data-science-process/prepare-data
2. https://www.aviapanorama.ru/2020/04/na-kakih-dannyh-v-rossii-obuchajut-nejroseti/
3. https://www.bigdataschool.ru/blog/data-preparation-operations.html
4. https://www.bigdataschool.ru/blog/dataset-data-preparation.html
5. https://cyberleninka.ru/article/n/formirovanie-obuchayuschey-vyborki-pri-ispolzovanii-iskusstvennyh-neyronnyh-setey-v-zadachah-poiska-oshibok-baz-dannyh/viewer
6. Абдрахманов М.И «Pandas. Работа с данными», 2020 г., 171 с.

# **ПРИЛОЖЕНИЕ А. Код программы**

import pandas as pd

import numpy as np

df = pd.read\_csv("/content/sample\_data/survey.csv")

df.head()

df.shape # размер датафрейма

df.info() # размерность данных

df.describe() # отображение статистики

df.nunique() # количество уникальных значений в каждом столбце

#Способы обработки пропущенных значений. Удаление

features = df.drop('Gender', 1) #удаление столбца Gender

labels = df['Gender']

#Способы обработки пропущенных значений. Подстановка часто используемого элемента

from sklearn.impute import SimpleImputer

imputer = SimpleImputer(missing\_values=np.nan, strategy='most\_frequent') #пропущенные значения заменяются самым частым значением в столбце

imputer.fit(features)

features = imputer.transform(features)

#Способы обработки пропущенных значений. Подстановка среднего значения

mean\_value=df['Age'].mean()

df['NewAge']=df['Age'].fillna(mean\_value)

df.head()

#Способы обработки пропущенных значений. Фиктивная подстановка

from sklearn.impute import SimpleImputer

imputer = SimpleImputer(missing\_values=np.nan, strategy='constant', fill\_value='const') #пропущенные значения заменяются самым частым значением в столбце

imputer.fit(features)

features = imputer.transform(features)

#нормализация данных

from sklearn import preprocessing

float\_array = df['Age'].values.astype(float).reshape(-1,1)

min\_max\_scaler = preprocessing.MinMaxScaler() #нормализация по формуле (x - min(x))/(max(x)-min(x))

scaled\_array = min\_max\_scaler.fit\_transform(float\_array)

df['NewAge1']=scaled\_array

df.head()

float\_array = df['Age'].values.astype(float).reshape(-1,1)

standard\_scaler = preprocessing.StandardScaler() #нормализация по формуле (x-x.mean())/x.std()

scaled\_array = standard\_scaler.fit\_transform(float\_array)

df['NewAge2']=scaled\_array

df.head()

float\_array = df['Age'].values.astype(float).reshape(-1,1)

normal\_scaler = preprocessing.Normalizer() #нормализация по формуле x / np.linalg.norm(x)

scaled\_array = normal\_scaler.fit\_transform(float\_array)

df['NewAge3']=scaled\_array

df.head()

float\_array = df['Age'].values.astype(float).reshape(-1,1)

robust\_scaler = preprocessing.RobustScaler() #нормализация по формуле (xi - Q1(x)) /(Q3(x) - Q1(x)), устойчив к выбросам

scaled\_array = robust\_scaler.fit\_transform(float\_array)

df['NewAge4']=scaled\_array

df.head()

float\_array = df['Age'].values.astype(float).reshape(-1,1)

quantile\_scaler = preprocessing.QuantileTransformer()

scaled\_array = quantile\_scaler.fit\_transform(float\_array)

df['NewAge5']=scaled\_array

df.head()

float\_array = df['Age'].values.astype(float).reshape(-1,1)

power\_scaler = preprocessing.PowerTransformer()

scaled\_array = power\_scaler.fit\_transform(float\_array)

df['NewAge6']=scaled\_array

df.head()

#поиск неявных дубликатов

male\_terms = ["male", "m", "mal", "msle", "malr", "mail", "make", "cis male", "man", "maile", "male (cis)", "cis man"]

female\_terms = ["female", "f", "woman", "femake", "femaile", "femake", "cis female", "cis-female/femme", "female (cis)", "femail", "cis woman"]

#выделяются три основных значения параметра "пол": "Male", "Female", "Other"

def clean\_gender(response):

    if response.lower().rstrip() in male\_terms:

        return "Male" #все возможные способы указания мужского пола  заменяются на "Male"

    elif response.lower().rstrip() in female\_terms:

        return "Female" #все возможные способы указания женского пола  заменяются на "Female"

    else:

        return "Other" #другие способы указания пола заменяются на "Male"

df['Gender'] = df["Gender"].apply(lambda x: clean\_gender(x)) #применяется функция clean\_gender к столбцу 'Gender'

#Обнаружение выбросов

df.Age.loc [(df.Age <14) | (df.Age> 110)] = np.nan #числа из столбца Age, которые меньше 14 или больше 110 заменяются на NAN

#

%matplotlib inline

import seaborn as sns

sns.set(color\_codes=True)

plot = sns.histplot(df.Age.dropna()) #гистограмма для столбца Age с удаленными пропущенными значениями

plot.figure.set\_size\_inches(6,6)

#кодирование данных

df['leave'].value\_counts(dropna=False)

from sklearn import preprocessing

label\_encoder = preprocessing.LabelEncoder()

label\_encoder.fit(df['leave'])

label\_encoder.transform(df['leave'])

#кодирование данных.One Hot

df\_2 = pd.get\_dummies(df,drop\_first=True)

df\_2.head()

#кодирование данных. LabelBinarizer

from sklearn.preprocessing import LabelBinarizer

lb = LabelBinarizer()

df['Major'] = lb.fit\_transform(df['Major'].astype(str))

df.head()

#Разделение данных для обучения и тестирования

from sklearn.model\_selection import train\_test\_split

features\_train, features\_test, labels\_train, labels\_test = train\_test\_split(features, labels, test\_size=0.2, random\_state = 0) # Feature scaling with StandardScaler

features\_train.size

features\_test.size

import matplotlib.pyplot as plt

df = pd.read\_csv("https://raw.githubusercontent.com/fivethirtyeight/"

                 "data/master/college-majors/recent-grads.csv")

df.head()

#Линейный график

df.plot(x="Rank", y=["P25th", "Median", "P75th"])

plt.show()

#гистограмма

median\_column = df["Median"]

median\_column.plot(kind="hist")

#корреляция данных

df.plot(x="Median", y="Unemployment\_rate", kind="scatter")

#группировка

groups = df.groupby("Major\_category")["Total"].sum().sort\_values()

groups.plot(kind="barh", fontsize=4)

#pie chart

groups.plot(kind="pie", label="")

# **ПРИЛОЖЕНИЕ Б. Ссылка на Google colab с кодом программы**

https://colab.research.google.com/drive/13M2ov3WRjR5gSOJRB4QkjxDwJunPYmsH?usp=sharing