Министерство науки и высшего образования РФ

Томский государственный университет
систем управления и радиоэлектроники

В.Н. Будакова, И.В. Крюков, С.А. Давыденко, Е.Ю. Костюченко

НАБОРЫ БИОМЕДИЦИНСКИХ ДАННЫХ В ЗАДАЧАХ МАШИННОГО ОБУЧЕНИЯ

Методические указания к лабораторным работам
для студентов направлений подготовки
09.04.04 Программная инженерия

УДК 004.8
ББК 32.813.5
Д 13

Давыденко С.А. НАБОРЫ БИОМЕДИЦИНСКИХ ДАННЫХ В ЗАДАЧАХ МАШИННОГО ОБУЧЕНИЯ: Учебно-методические указания / В.Н. Будакова, И.В. Крюков, С.А. Давыденко, Е.Ю. Костюченко. — Томск: ТУСУР, 2024. — 111 с.

Настоящие учебно-методические указания содержат описания практических работ по дисциплине «Наборы биомедицинских данных в задачах машинного обучения» для направлений подготовки, входящих в укрупненную группу специальностей и направлений 09.04.04 Программная инженерия.

Одобрено на заседании кафедры КИБЭВС протокол №7 от 30.08.2024 года

УДК 004.8
ББК 32.813.5

Содержание

Введение

Целью преподавания дисциплины является изучение видов и методов обработки наборов данных для использования в машинном обучении.

Задачи изучения дисциплины:

- Изучить понятие наборов данных, их особенности и виды.
- Овладеть методами очистки и предварительной обработки данных.
- Освоить инструменты для работы с данными.
- Развить навыки работы с различными типами данных.

ЛАБОРАТОРНАЯ РАБОТА № 1

ПРЕДВАРИТЕЛЬНАЯ ОБРАБОТКА ДАННЫХ И ИХ АНАЛИЗ

Целью работы является ознакомление с теоретическими и практическими основами работы с разведочным анализом данных и его применение на примере открытого набора данных.

Краткие теоретические сведения

В основе разведочного анализа (EDA) лежит идея о том, что, прежде чем строить сложные модели или делать окончательные выводы, нужно тщательно изучить данные. Это включает в себя визуализацию, статистический анализ, проверку предположений и формулировку гипотез на основе наблюдаемых закономерностей.

EDA нужен:

– для понимания данных: прежде чем принимать решения или строить модели, важно понять, что представляют собой ваша информация, каковы её основные характеристики и структура;

– для очистки данных: EDA помогает выявить ошибки и аномалии в имеющихся материалах, которые могут исказить анализ, например, пропущенные значения или выбросы;

– для формулировки гипотез: на основе наблюдаемых закономерностей можно сформулировать гипотезы для дальнейшего тестирования в процессе аналитической работы;

– для выбора моделей: понимание данных позволяет выбрать наиболее подходящие статистические модели и методы анализа.

Важная составляющая разведочного анализа – визуализация данных. Библиотека Pandas имеет встроенные средства визуализации на основе графики Matplotlib. Библиотека Seaborn содержит больше высокоуровневых

возможностей и абстрагирует сложность, позволяя проектировать графики в соответствии с нуждами пользователя.

Сводные статистики и меры центральной тенденции позволяют понять типичные и наиболее значимые значения в наборе данных.

– Среднее (Mean) – сумма всех значений, разделённая на количество значений.

– Медиана (Median) – серединное значение набора данных, расположенного в порядке от меньшего к большему. Если в наборе данных нечетное количество значений, медианой будет серединное значение. Если же количество четное, тогда ищется среднее значение между двумя серединными.

– Мода (Mode) – значение, которое встречается наиболее часто в наборе данных.

Для совместного изучения признаков используется метод groupby, он позволяет группировать данные по одному или нескольким атрибутам, вычисляя затем агрегированные показатели в каждой группе.

С помощью матрицы корреляций можно визуально оценить, насколько признаки связаны друг с другом и не дублируют ли они информацию.

Для исследования трёх и более признаков полезным инструментов являются сводные таблицы (pivot tables).

Требования к выполнению лабораторной работы

1. Ознакомиться с методическими указаниями, а также представленными примерами в блокноте Google Colab (https://clck.ru/3EEpZj).

2. По действиям, представленным в ходе работы, дополнить элементы кода в блокноте Google Colab и выполнить анализ данных.

3. Самостоятельно провести анализ данных, ссылка на которые, представлена в конце блокнота Google Colab.

4. Ответить на контрольные вопросы.

1 Ход работы

Перед началом работы необходимо импортировать библиотеки для визуализации данных. Обычно встроенная графика Pandas, библиотеки Matplotlib и Seaborn используются совместно. Это позволяет создавать отчёты и визуализировать многомерные данные в виде наглядных таблиц, графиков и диаграмм.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set();
```

После настройки всех библиотек, необходимо скачать набор данных «Complete Blood Count (CBC) Test» с платформы Kaggle. Следующим шагом будет загрузка датасета в файловое пространство Colab с помощью специального функционала по выбору и загрузке файлов.

Данные, прошедшие предварительную подготовку и обработку, обычно имеют табличный формат и хранятся в виде CSV-файлов (а также TSV, XLS, XLSX и т.д.).

Наиболее существенные параметры настройки для просмотра файлов: тип разделителя ячеек (по умолчанию – запятая), наличие строки заголовка (указывается её номер; по умолчанию имена признаков считываются из первой строки файла), наличие колонки с индексами (идентификаторами) строк (также указывается номер; по умолчанию – отсутствует).

В данном случае используется метод «read_excel», а также происходит проверка первых пяти записей таблицы (рисунок 1.1).

```
df=pd.read_excel('/content/cbc information.xlsx')
df.head(5)
```

	ID	WBC	LYMp	MIDp	NEUTp	LYMn	MIDn	NEUTn	RBC	HGB	...	MCV	MCH	MCHC	RDWSD	RDWCV	PLT	MPV	PDW	PCT	PLCR
0	1	10.0	43.2	6.7	50.1	4.3	0.7	5.0	2.77	7.3	...	87.7	26.3	30.1	35.3	11.4	189.0	9.2	12.5	0.17	22.3
1	2	10.0	42.4	5.3	52.3	4.2	0.5	5.3	2.84	7.3	...	88.2	25.7	20.2	35.3	11.4	180.0	8.9	12.5	0.16	19.5
2	3	7.2	30.7	8.6	60.7	2.2	0.6	4.4	3.97	9.0	...	77.0	22.6	29.5	37.2	13.7	148.0	10.1	14.3	0.14	30.5
3	4	6.0	30.2	6.3	63.5	1.8	0.4	3.8	4.22	3.8	...	77.9	23.2	29.8	46.5	17.0	143.0	8.6	11.3	0.12	16.4
4	5	4.2	39.1	7.2	53.7	1.6	0.3	2.3	3.93	0.4	...	80.6	23.9	29.7	42.7	15.1	236.0	19.5	12.8	0.22	24.8

5 rows × 21 columns

Рисунок 1.1 – Просмотр первых записей таблицы

Для получения общей информации о наборе данных существует команда «info», она позволяет узнать тип каждого признака, а также есть ли в данных пропуски (рисунок 1.2).

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 21 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   ID      500 non-null    int64  
 1   WBC     500 non-null    float64
 2   LYMp    500 non-null    float64
 3   MIDp    500 non-null    float64
 4   NEUTp   500 non-null    float64
 5   LYMn    500 non-null    float64
 6   MIDn    500 non-null    float64
 7   NEUTn   500 non-null    float64
 8   RBC     500 non-null    float64
 9   HGB     500 non-null    float64
 10  HCT     500 non-null    float64
 11  MCV     500 non-null    float64
 12  MCH     500 non-null    float64
 13  MCHC    500 non-null    float64
 14  RDWSD   500 non-null    float64
 15  RDWCV   500 non-null    float64
 16  PLT     500 non-null    float64
 17  MPV     500 non-null    float64
 18  PDW     500 non-null    float64
 19  PCT     500 non-null    float64
 20  PLCR    500 non-null    float64
dtypes: float64(20), int64(1)
memory usage: 82.2 KB
```

Рисунок 1.2 – Общая информация о наборе данных

Исходя из результатов выполнения вышеуказанного метода, пропусков в наборе данных не было обнаружено, так как по каждому признаку имеется

полный объем записей, то есть в каждом столбце имеется 500 ненулевых значений.

Данный набор данных содержит только числовые переменные. Метод «describe()» позволяет собрать некоторую статистику по каждому числовому признаку (рисунок 1.3).

```
[6] df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
ID	500.0	250.50000	144.481833	1.00	125.750	250.50	375.250	500.0
WBC	500.0	7.37180	3.704331	0.80	5.100	6.70	8.700	45.7
LYMp	500.0	25.84500	11.273243	6.20	17.000	24.80	32.725	91.4
MIDp	500.0	8.69262	6.958640	0.50	6.900	7.80	8.900	77.0
NEUTp	500.0	77.51100	236.630786	0.70	58.875	66.95	73.925	5317.0
LYMn	500.0	1.88076	2.139243	0.20	1.200	1.60	2.100	41.8
MIDn	500.0	0.67208	0.826036	0.10	0.400	0.50	0.700	10.0
NEUTn	500.0	5.14094	4.600272	0.50	3.100	4.40	6.000	79.0
RBC	500.0	4.88286	4.403204	1.42	4.240	4.59	5.010	90.8
HGB	500.0	11.74002	5.628807	-10.00	9.900	11.50	13.100	87.1
HCT	500.0	46.15260	167.985830	2.00	32.975	37.10	41.000	3715.0
MCV	500.0	82.18138	42.318848	-79.30	76.575	82.80	86.500	990.0
MCH	500.0	37.45600	177.879640	10.90	23.175	25.70	27.425	3117.0
MCHC	500.0	30.95082	4.577839	11.50	29.700	30.90	31.900	92.8
RDWSD	500.0	38.07822	18.791718	7.20	35.300	37.20	39.000	390.0
RDWCV	500.0	13.65478	6.080670	8.90	12.200	12.70	13.700	124.0
PLT	500.0	171.22180	61.669883	11.30	135.000	163.50	197.250	508.0
MPV	500.0	12.87302	43.040472	0.14	9.200	9.70	10.200	919.0
PDW	500.0	13.72720	4.347798	8.40	12.000	13.30	14.900	97.0
PCT	500.0	0.26028	1.097659	0.01	0.130	0.15	0.190	13.6
PLCR	500.0	26.29202	7.145895	0.12	21.375	26.10	30.600	48.5

Рисунок 1.3 – Статистика по всем признакам

Анализ представленных статистик позволяет заключить, что все данные в таблице имеют числовую природу. Так как для всех столбцов вычислены значения: mean (среднее), std (стандартное отклонение), min (минимум), max (максимум), а также процентили (25%, 50%, 75%). Эти показатели применимы

только к числовым данным. О непрерывности данных можно судить по широкому диапазону значений (от min до max).

Помимо числового признака, существуют ещё порядковые и категориальные.

Далее каждый признак следует рассмотреть отдельно. Нагляднее всего использовать инструменты для визуализации данных. На рисунке 1.4 представлено распределение значений гемоглобина в крови пациентов.

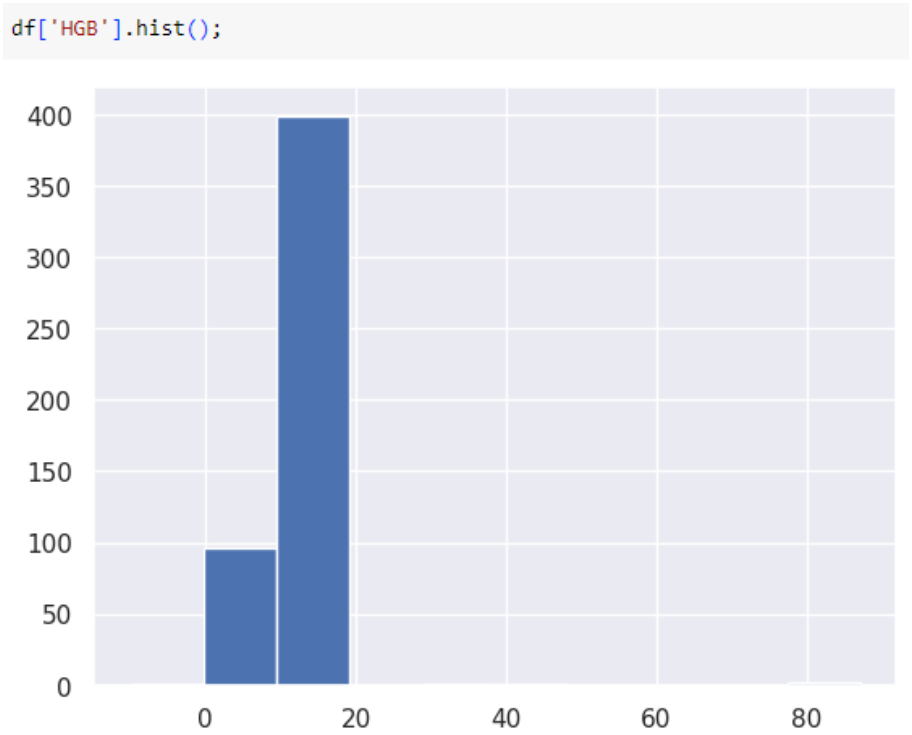


Рисунок 1.4 – Гистограмма для признака «HGB»

График с настройками по умолчанию оказался малоинформативным. Детального распределения на нем не видно. Чтобы это исправить, нужно добавить параметр «bins» (рисунок 1.5). Этот аргумент задает размер интервалов на оси X. По умолчанию используется 10.

```
df['HGB'].hist(bins=20);
```

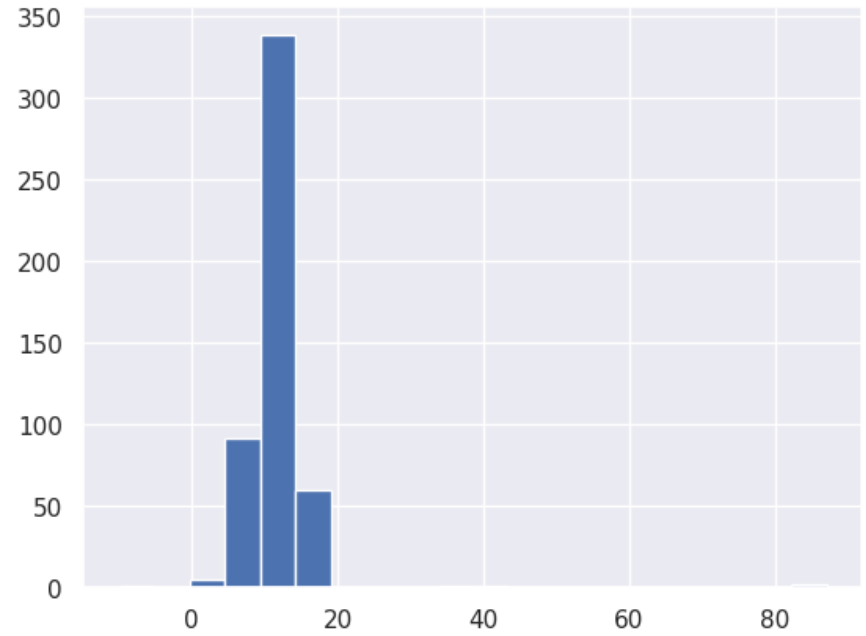


Рисунок 1.5 – Гистограмма с дополнительным параметром

После добавления параметра гистограмма приобрела вид нормального распределения. Однако при данном выборе визуализации данных не видны выбросы (outliers) – точки, «выбивающиеся» из общей картины. Поэтому иногда полезнее применить boxplot («ящик с усами»), показанный на рисунке 1.6.

```
sns.boxplot(df['HGB']);
```

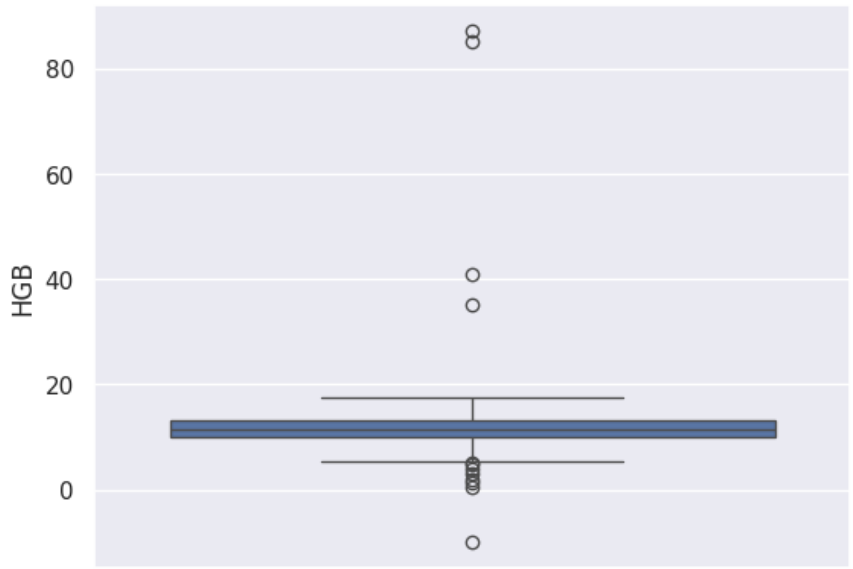


Рисунок 1.6 – Исследование выбросов у признака «HGB»

Отдельные точки на графике соответствуют выбросам – нетипичным для данной выборки значениям. По рисунку 1.6 можно увидеть, что их оказалось достаточно. Если усы очень короткие, а вне них видна плотная концентрация точек, это указывает на сильное влияние выбросов.

Следующим шагом предстоит изучить связи между несколькими признаками. Для этого существует операция группировка. Метод «groupby» позволяет группировать данные по одному или нескольким атрибутам, вычисляя затем агрегированные показатели в каждой группе (рисунок 1.7).

```
df.groupby('PCT')['PLT'].mean()
```

PCT	PLT
0.01	17.000000
0.03	37.000000
0.04	40.000000
0.05	45.500000
0.07	78.166667
0.08	85.600000
0.09	91.888889
0.10	112.823529

Рисунок 1.7 – Использование метода «groupby»

Поскольку тромбокрит (PCT) – это доля тромбоцитов в общем объеме крови, его уровень напрямую зависит от количества тромбоцитов (PLT). Изменение количества тромбоцитов будет отражаться на значении тромбокрита. Эти вычисления можно также визуализировать с помощью встроенной графики Pandas (рисунок 1.8).

```
df.groupby('PCT')['PLT'].mean().plot(kind='bar')
plt.ylabel('PLT')
plt.show();
```

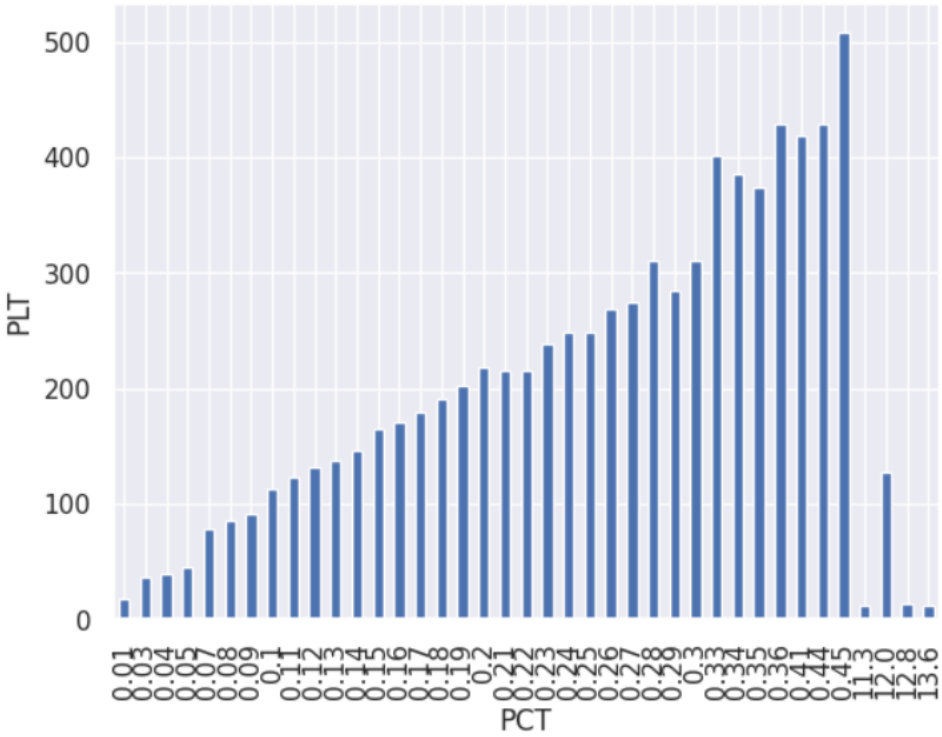


Рисунок 1.8 – Визуализация вычислений

Для исследования пар числовых признаков является диаграмма рассеяния (scatter plot) (рисунок 1.9).

```
plt.scatter(df['PCT'], df['PLT']);
```

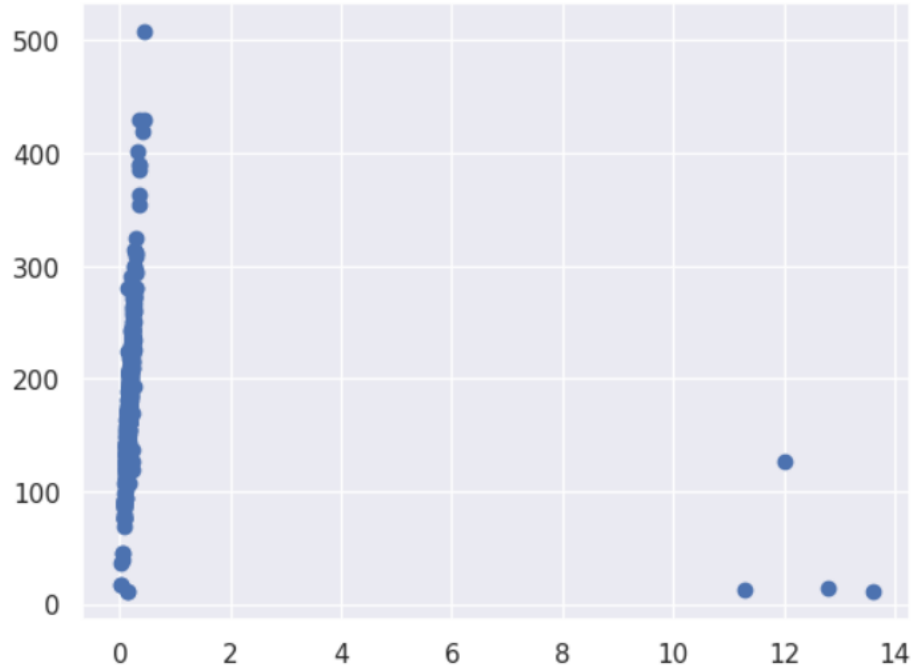


Рисунок 1.9 – Распределение PCT и PLT в крови людей

Нормальные показатели для данных элементов анализа крови: PCT – от 0,108 до 0,282, PLT – от 180 до 320 млрд/литр. Исходя из полученного графика, можно сделать вывод, что примерно треть пациентов имеет отклонения от нормы. Так же между элементами четко видна прямолинейная связь. Если точки на диаграмме расположены так, что их можно приблизительно описать прямой линией (в данном случае восходящая линия идет снизу вверх), это указывает на положительную линейную зависимость. Для изучения совместного распределения двух числовых признаков полезным может оказаться метод jointplot библиотеки Seaborn (рисунок 1.10).

```
sns.jointplot(x='LYMn', y='WBC', data=df);
```

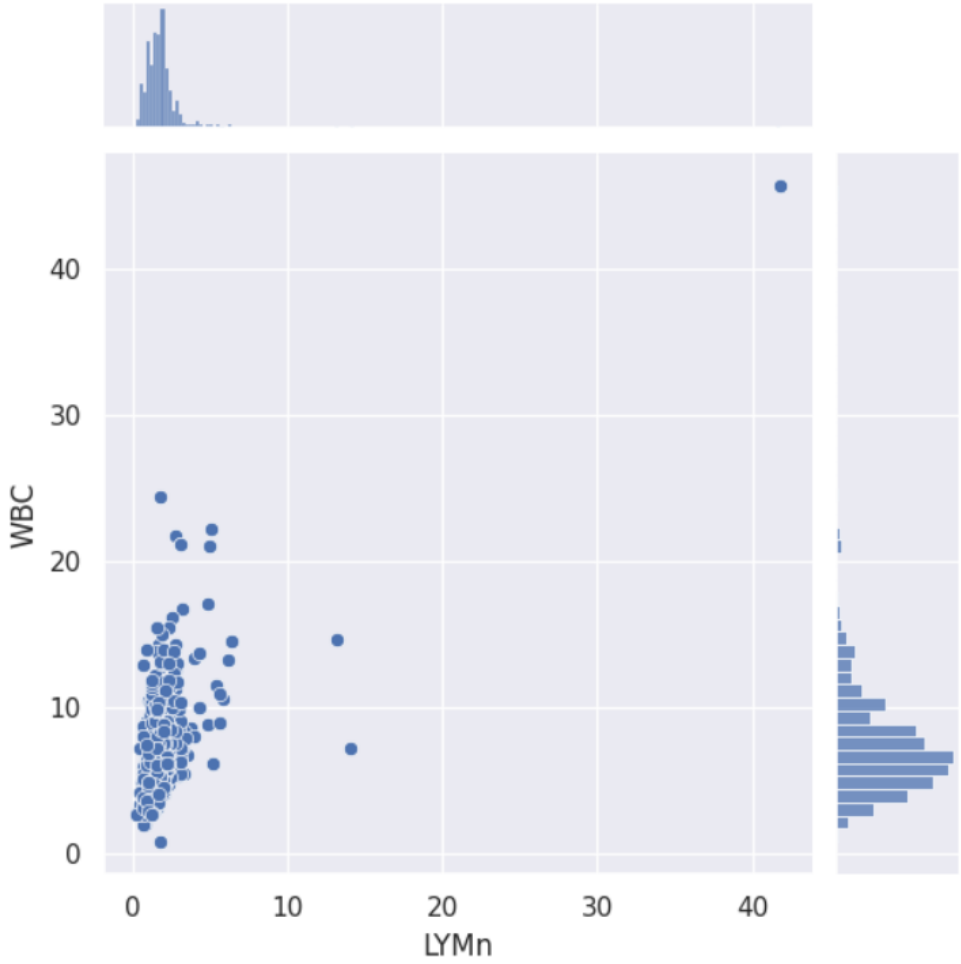


Рисунок 1.10 – Результат применения метода jointplot

Ошибки и аномалии в данных чётко видны на графике 1.10. Также можно заключить, что без учёта выбросов рассмотренные элементы имеют

распределения, близкие к нормальному, так как данные находятся очень близко друг к другу и создают скопление точек на графике.

Чтобы увидеть все существующие связи между признаками рассматриваемого набора данных, достаточно вывести матрицу корреляции числовых признаков. С помощью этого инструмента можно визуально оценить, насколько признаки кореллируют друг с другом и не дублируют ли они информацию.

```
colormap = plt.cm.RdBu
plt.figure(figsize=(14,12))

plt.title('Pearson Correlation of Features', y=1.05, size=18)
sns.heatmap(df.corr(),
        linewidths=0.1, vmax=1.0,
        square=True, cmap=colormap, annot=True, fmt='.1g')
```

Следующим шагом необходимо провести вычисления не на всей обучающей выборке, а на некоторой её части. Для этого нужно знать и понимать способы доступа к ячейкам в датафреймах.

```
r = df['RDWCV']
type(r)
«pandas.core.series.Series»
```

Таблица представляет собой набор именованных столбцов (Series). Доступ к столбцам осуществляется по ключу – названию столбца, как в словарях Python. Со строками действия осуществляются иначе. При попытке использовать обычный индекс в обращении к строке появляется ошибка «Traceback (most recent call last)». В таком случае лучше обратиться к «неявному» признаку (рисунок 1.11).

```
first_patient = df.iloc[0]
print(first_patient)
```

```
ID         1.00
WBC       10.00
LYMp      43.20
MIDp       6.70
NEUTp     50.10
LYMn       4.30
MIDn       0.70
NEUTn      5.00
RBC        2.77
HGB        7.30
HCT       24.20
MCV       87.70
MCH       26.30
MCHC      30.10
RDWSD     35.30
RDWCV     11.40
PLT      189.00
MPV        9.20
PDW       12.50
PCT        0.17
PLCR      22.30
Name: 0, dtype: float64
```

Рисунок 1.11 – Использование «неявного» индекса

Чтобы узнать, например, относительную ширину распределения тромбоцитов по объему первого пациента, нужно применить явное индексирование (loc).

```
print(df.loc[0, 'PDW'])
```

В переменную *r* был записан весь столбец с данными о ширине распределения эритроцитов. Нормальные показатели варьируются в пределах от 11 до 16. Решением к задаче для поиска пациентов с отклонением от нормы RDWCV является код:

```
r[(r > 16) | (r < 11)].shape[0]
```

Контрольные вопросы

1. Что включает в себя разведочный анализ данных (EDA) и для чего он нужен?

2. Какие существую сводные статистики?

3. Какие инструменты используются при изучении сразу двух признаков?

4. На какие виды можно разделить признаки в наборах данных?

ЛАБОРАТОРНАЯ РАБОТА № 2

СРАВНЕНИЕ АРХИТЕКТУР СВЕРТОЧНЫХ НЕЙРОННЫХ СЕТЕЙ ДЛЯ КЛАССИФИКАЦИИ ИЗОБРАЖЕНИЙ МРТ

Целью данной лабораторной работы является сравнение различных архитектур сверточных нейронных сетей для задачи классификации изображений МРТ путем изучения и реализации моделей EfficientNet и Mask R-CNN.

Краткие теоретические сведения

В современном мире технологии компьютерного зрения используются во многих сферах нашей жизни. Их применяют для таргетирования рекламы в маркетинге, для анализа снимков и дальнейшей помощи при определении диагнозов в медицине, в беспилотных автомобилях и так далее.

На данный момент лучше всего с обработкой изображений справляются свёрточные нейронные сети (CNN). Основное отличие этой архитектуры от построения классических нейронных сетей заключается в наличии слоёв свёртки (convolutional layer) – скрытых слоев нейронной сети, в которых происходит свёртка изображения, с помощью фильтров, а также слоев субдискретизации (pooling layer) и полносвязных слоев (fully connected layers).

EfficientNet — это семейство моделей свёрточных нейронных сетей (CNN), разработанных в 2019 году. Главной идеей EfficientNet является пропорциональное масштабирование глубины, ширины и разрешения модели, что позволяет достичь лучшего баланса между точностью и вычислительной сложностью.

Основная идея EfficientNet заключается в том, что все три аспекта модели (глубина, ширина и разрешение) важны для достижения высокой точности классификации изображений, и все они должны быть пропорционально увеличены для получения лучших результатов. В EfficientNet используется метод компоновки моделей, называемый «комплексным масштабированием»

(compound scaling), который позволяет создавать модели различной сложности, начиная от EfficientNet-B0 с 5 миллионами параметров и заканчивая EfficientNet-B7 с 66 миллионами параметров.

EfficientNet был обучен на большом количестве данных ImageNet и достиг высокой точности классификации изображений, соперничая с другими популярными моделями CNN, такими как VGG, ResNet и Inception. Кроме того, EfficientNet показал отличные результаты на других наборах данных, таких как CIFAR-100 и inaturalist18, и был использован в задачах переобучения (transfer learning) для различных приложений, таких как распознавание лиц, детектирование объектов и сегментация изображений.

VGG-16 — это модель свёрточной нейронной сети (CNN), разработанная группой исследователей под руководством Simonyan и Zisserman в 2014 году. VGG-16 является одной из самых популярных архитектур CNN и была одной из лучших моделей на конкурсе ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2014 года.

Архитектура VGG-16 состоит из 16 слоев, в том числе 13 свёрточных слоев и 3 полносвязных слоёв (fully connected layers). Все свёрточные слои в VGG-16 используют ядро 3x3, что делает модель более компактной и простой в реализации, чем другие модели CNN с большими ядрами. Кроме того, все слои в VGG-16 имеют одинаковую глубину, что упрощает сравнение различных версий модели.

VGG-16 была обучена на большом количестве данных ImageNet и достигла высокой точности классификации изображений. В результате, преобученная модель VGG-16 часто используется в качестве базовой модели для задач переобучения (transfer learning), когда модель обучается на другом наборе данных, например, на данных медицинских изображений.

Требования к выполнению лабораторной работы

1. Ознакомиться с методическими указаниями.

2. Выполнить ход работы.

3. Самостоятельно провести анализ данных, ссылка на которые,

представлена в электронном курсе.

4. Для реализованных архитектур оценить точность прогнозирования и скорость выполнения.

5. Ответить на контрольные вопросы.

1 Ход работы

1.1 EfficientNet

Изображения в наборе данных представляют собой МРТ изображения головного мозга в формате JPEG. Набор данных сбалансирован по классам, то есть количество изображений с опухолью и без опухоли примерно одинаково. Постройте модель сверточной нейронной сети на основе архитектуры EfficientNet.

Импортируйте библиотеки и выполните предварительную обработку данных. Библиотеки matplotlib и sklearn используются для обработки данных и оценки модели. Библиотека tensorflow используется для создания и обучения модели. Также в коде выполняются следующие действия:

– игнорируются предупреждения, которые могут появиться во время выполнения кода;

– устанавливаются пути к директориям, содержащим набор данных для обучения модели;

– выводится количество изображений в директориях с меткой «без опухоли» и «с опухолью».

После этого, код продолжается с загрузки изображений из директорий и их предварительной обработки перед обучением модели.

```
import warnings
warnings.filterwarnings('ignore')
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.metrics import classification_report , confusion_matrix , accuracy_score
from sklearn.model_selection import train_test_split
import cv2
from PIL import Image
```

```
import tensorflow as tf
from tensorflow import keras
from keras import Sequential
from keras.layers import Input, Dense,Conv2D , MaxPooling2D, Flatten,BatchNormalization,Dropout
from tensorflow.keras.preprocessing import image_dataset_from_directory
import tensorflow_hub as hub
```

Установите путь к директории, содержащей набор данных для обучения модели.

```
folder_path_no = "/kaggle/input/brain-mri-images-for-brain-tumor-detection/no"
folder_path_yes = "/kaggle/input/brain-mri-images-for-brain-tumor-detection/yes"
folder_no = os.listdir(folder_path_no)
folder_yes = os.listdir(folder_path_yes)
```

Создайте метки *«labels»* для изображений в наборе данных. Метки представляют собой числа, которые соответствуют классу каждого изображения. В данном случае, метка «0» присвоена изображениям без опухоли, а метка «1» - изображениям с опухолью. Список *no_label* содержит метку «0» для каждого изображения в директории с меткой «без опухоли», а список *yes_label* содержит метку «1» для каждого изображения в директории с меткой «с опухолью». Затем, оба списка объединяются в один список *labels*, который содержит метки для всех изображений в наборе данных. Длина списка *labels* выводится в консоль и должна быть равна общему количеству изображений в наборе данных.

```
no_label = [0]*len(folder_no)
yes_label = [1]*len(folder_yes)

labels = no_label + yes_label
print(len(labels))
```

Выполните загрузку и обработку изображений из директорий с метками «без опухоли» и «с опухолью». Изображения загружаются с помощью библиотеки PIL и преобразуются в массивы NumPy. Также выполняются следующие действия:

– каждое изображение уменьшается до размера 224x224 пикселей с помощью метода resize;

– если изображение не в формате RGB, оно преобразуется в формат RGB с помощью метода convert;

– каждое изображение преобразуется в массив NumPy с помощью метода np.array.

После обработки, все изображения добавляются в список *data*. В результате, список *data* содержит все обработанные изображения из обеих директорий.

```
data = []
for img in folder_no:
    image = Image.open("/kaggle/input/brain-mri-images-for-brain-tumor-detection/no/"+img)
    image = image.resize((224,224))
    image = image.convert("RGB")
    image = np.array(image)
    data.append(image)
for img in folder_yes:
    image = Image.open("/kaggle/input/brain-mri-images-for-brain-tumor-detection/yes/"+img)
    image = image.resize((224,224))
    image = image.convert("RGB")
    image = np.array(image)
    data.append(image)
```

Преобразуйте список *data* и список *labels* в массивы NumPy *X* и *Y* соответственно. Массив *X* должен содержать все предварительно обработанные изображения из набора данных, а массив *Y* содержать соответствующие метки для каждого изображения. Теперь массив *X* и массив *Y* можно использовать для обучения модели EfficientNet.

Выполните разделение набора данных на обучающую и тестовую выборки с помощью функции train_test_split из библиотеки sklearn.model_selection:

– x и y - массивы данных и меток, которые нужно разделить;

– test_size=0.10 - доля данных, которые будут помещены в тестовую выборку (в данном случае, 10%);

– shuffle=True - параметр, указывающий на то, что данные нужно перемешать перед разделением.

Результат выполнения функции - четыре массива: *x_train* и *y_train* (обучающая выборка) и *x_test* и *y_test* (тестовая выборка).

Отобразите 16 случайных изображений из обучающей выборки *x_train* с их соответствующими метками *y_train*. Используем библиотеку matplotlib для создания графика и отображения изображений. Пример вывода изображений представлен на рисунке 1.1.

```
class_labels=["No Tumor","Tumor"]
plt.figure(figsize=(16,20))

for i in range(16):
    plt.subplot(4,4,i+1)
    plt.imshow(x_train[i])
    plt.title(f"{class_labels[y_train[i]]}")
    plt.axis("off")
```

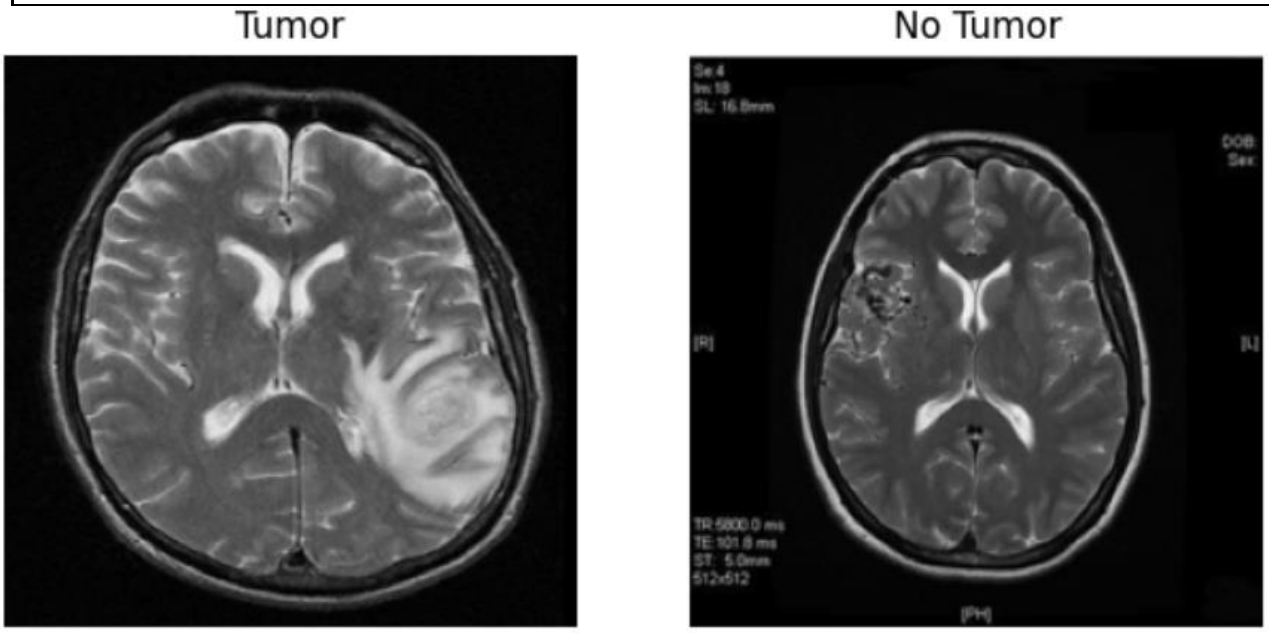


Рисунок 1.1 – Вывод изображений МРТ

Выполните масштабирование значений пикселей в изображениях обучающей и тестовой выборок. Масштабирование заключается в делении значений пикселей на максимальное значение, которое может принять пиксель.

После масштабирования, значения пикселей в изображениях обучающей выборки хранятся в массиве *x_train_scaled*, а значения пикселей в изображениях тестовой выборки - в массиве *x_test_scaled*. После масштабирования, значения пикселей в изображениях будут находиться в диапазоне от 0 до 1.

```
model = Sequential()
model.add(Input(shape=(224,224,3)))
model.add(Conv2D(filters=80,kernel_size=(3,3),padding="valid", strides=(1,1) ,activation="relu"))
```

```
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Conv2D(filters=64,kernel_size=(3,3),padding="valid", strides=(1,1), activation="relu"))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Flatten())
model.add(Dense(units=500, activation='relu'))
model.add(Dropout(0.3))
model.add(Dense(units=500, activation='relu'))
model.add(Dropout(0.3))
model.add(Dense(units=1, activation='sigmoid'))
model.compile(optimizer="adam", loss="binary_crossentropy",
              metrics=["accuracy"])
model.summary()
```

Запустите обучение модели на обучающей выборке *x_train_scaled* и *y_train* в течение 10 эпох. Параметры обучения:

– *epochs* - количество эпох обучения;

– *validation_data* - данные для валидации, которые используются для оценки точности модели во время обучения.

Результаты обучения сохраняются в переменную history, которая содержит информацию о значении функции потерь и точности в течение каждой эпохи для обучающей выборки, так и для данных валидации.

Оцените модель на тестовой выборке *x_test_scaled* и *y_test* и выведите точность модели на тестовых данных с помощью метода evaluate. В данном случае, переменная *loss* содержит значение функции потерь, а переменная *acc* - точность модели на тестовых данных.

```
loss, acc = model.evaluate(x_test_scaled,y_test)
print("Accuracy on Test Data:",acc)
```

Выполните предсказания модели на тестовой выборке *x_test_scaled* и преобразуйте их в бинарные метки (0 или 1). Сначала, метод predict модели используется для получения предсказаний для каждой картинки в тестовой выборке. Результат - массив с вероятностями принадлежности к каждому классу для каждой картинки. Затем, список *y_pred* создается путем преобразования вероятностей в бинарные метки, если вероятность принадлежности к классу «Tumor» больше или равна 0.5, то метка устанавливается в 1, иначе в 0. Наконец, выводятся первые 6 значений из списка *y_pred*, которые представляют собой

предсказанные метки для первых 6 картинок в тестовой выборке.

```
y_pred = model.predict(x_test_scaled)
y_pred = [1 if i>=0.5 else 0 for i in y_pred]
y_pred[:6]
```

Выведите матрицу запутанностей и отчет о классификации для предсказаний модели на тестовой выборке.

```
print("Confusion Matrix:\n",confusion_matrix(y_test,y_pred))
print()
print("Classification Report:\n",classification_report(y_test,y_pred))
```

Отобразите 16 случайных изображений из тестовой выборки *x_test* вместе с их фактическими и предсказанными метками. Используйте библиотеку matplotlib для создания графика и отображения изображений. Пример вывода изображений представлен на рисунке 1.2.

```
class_labels=["No Tumor","Tumor"]
plt.figure(figsize=(16,20))

for i in range(16):
    plt.subplot(4,4,i+1)
    plt.imshow(x_test[i])
    plt.title(f"Actual label:{class_labels[y_test[i]]}\nPredicted label:{class_labels[y_pred[i]]}")
    plt.axis("off")
```

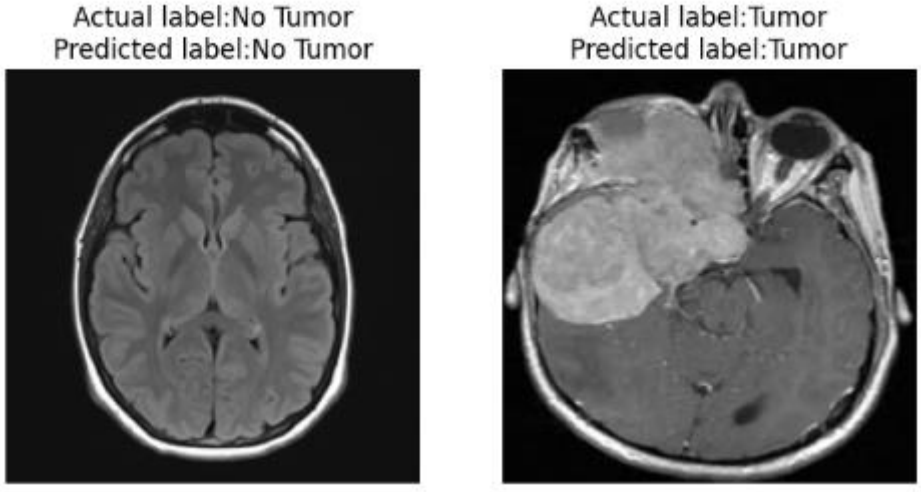


Рисунок 1.2 – Вывод изображений МРТ и результаты прогнозов

Загрузите предобученную модель EfficientNet-B0 из TF Hub и добавьте к ней полносвязный слой с одним выходом и сигмоидной активацией.

```
path = "https://tfhub.dev/google/efficientnet/b0/classification/1" # 224x224x3
efficient_model = hub.KerasLayer(path,input_shape=(224,224,3),trainable=False)
eff_model = Sequential()
eff_model.add(efficient_model)
eff_model.add(Dense(units=1, activation="sigmoid"))
eff_model.summary()
```

Скомпилируйте модель EfficientNet и запустите ее обучение на обучающей выборке *x_train_scaled* и *y_train* в течение 10 эпох. Используйте следующие параметры обучения:

– optimizer = "adam" - используемый оптимизатор для обучения модели.

– loss = "binary_crossentropy" - функция потерь, используемая для обучения модели.

– metrics = ["accuracy"] - метрика, используемая для оценки качества модели во время обучения.

– epochs = 10 - количество эпох обучения.

– validation_data = (x_test_scaled,y_test) - данные для валидации, которые используются для оценки точности модели во время обучения.

```
eff_model.compile(optimizer="adam",
            loss="binary_crossentropy",
            metrics=["accuracy"])
history_4 = eff_model.fit(x_train_scaled , y_train ,
          epochs=10, validation_data=(x_test_scaled,y_test))
```

Оцените модель EfficientNet на обучающей выборке *x_train_scaled* и *y_train* и выведите точность модели на обучающей выборке. Импортируйте функции для построения кривых Precision-Recall и ROC-кривой из библиотеки sklearn.metrics.

```
from sklearn.metrics import roc_curve, precision_recall_curve, auc
```

Постройте ROC-кривую (Receiver Operating Characteristic) для

предсказаний модели EfficientNet на тестовой выборке и вычислите площадь под кривой (Area Under the Curve, AUC).

```
y_prob = eff_model.predict(x_test_scaled)

fpr, tpr, threshold = roc_curve(y_test, y_prob)

roc_auc = auc(fpr, tpr)
print(f"ROC - Area :{roc_auc}")
```

Визуализируйте график ROC-кривой на основе массивов FPR и TPR, полученных ранее (рисунок 1.3).

```
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (AUC = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc='lower right')
plt.show()
```

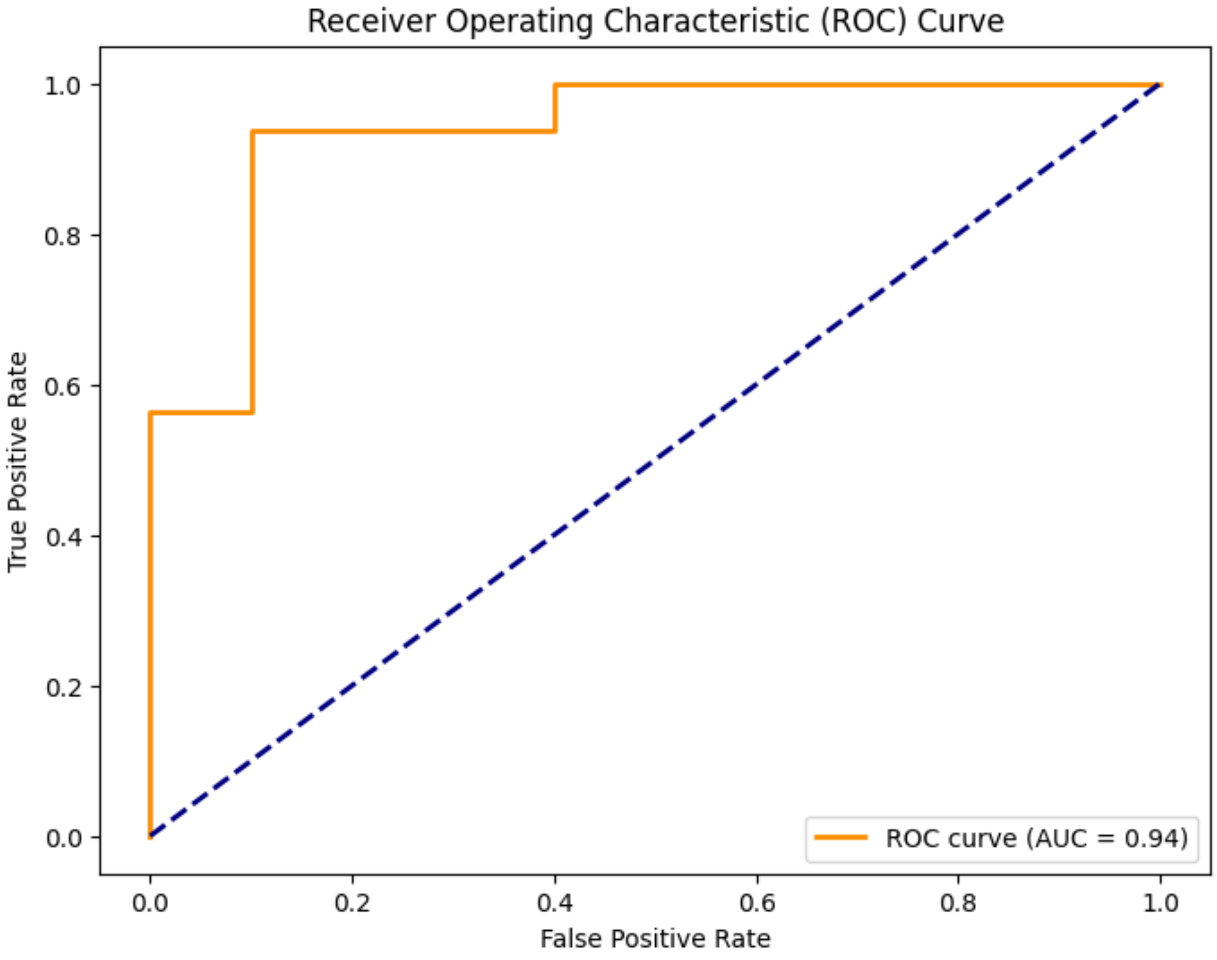


Рисунок 1.3 – ROC-кривая

Постройте Precision-Recall кривую (PR-curve) для предсказаний модели на тестовой выборке и вычислим площадь под кривой (AUC). Сначала, метод predict() модели используется для получения предсказанных вероятностей для каждой картинки в тестовой выборке. Затем, функция precision_recall_curve() используется для построения Precision-Recall кривой на основании массивов истинных меток *y_test* и предсказанных вероятностей *y_prob*. Функция возвращает массивы значений precision и recall для различных порогов (рисунок 1.4).

```
y_prob = eff_model.predict(x_test_scaled)

precision, recall, threshold = precision_recall_curve(y_test, y_prob)

# Compute ROC curve and ROC area
pr_auc = auc(recall,precision)
print(f"ROC - Area :{pr_auc}")
```

```
plt.figure(figsize=(8, 6))
plt.step(recall, precision, color='b', where='post', label=f'PR curve (AUC = {pr_auc:.2f})')
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title('Precision-Recall (PR) Curve')
plt.legend(loc='upper right')
plt.show()
```

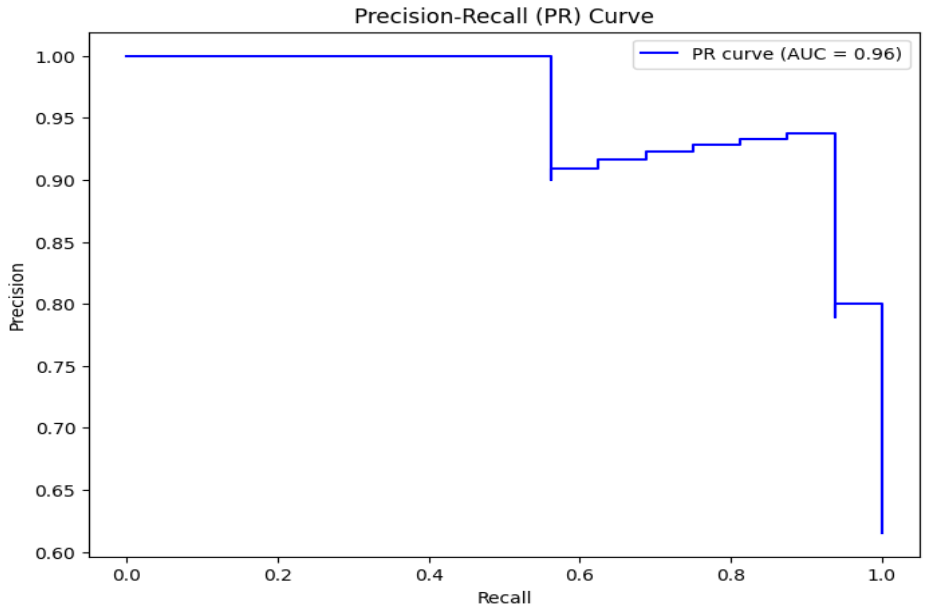


Рисунок 1.4 – PR-curve

1.2 VGG-16

Импортируйте необходимые библиотеки и установите некоторые параметры для работы с моделью.

```
import numpy as np
from tqdm import tqdm
import cv2
import os
import shutil
import itertools
import imutils
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelBinarizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix
import plotly.graph_objs as go
from plotly.offline import init_notebook_mode, iplot
from plotly import tools
from keras.preprocessing.image import ImageDataGenerator
from keras.applications.vgg16 import VGG16, preprocess_input
from keras import layers
from keras.models import Model, Sequential
from keras.optimizers import Adam, RMSprop
from keras.callbacks import EarlyStopping
init_notebook_mode(connected=True)
RANDOM_SEED = 123
```

Создайте несколько новых директорий для хранения данных. Разделите данные на обучающую, валидационную и тестовую выборки. Директория IMG_PATH содержит данные об изображениях с наличием или отсутствием опухоли.

```
IMG_PATH = '../input/brain-mri-images-for-brain-tumor-detection/brain_tumor_dataset/'
for CLASS in os.listdir(IMG_PATH):
    if not CLASS.startswith('.'):
        IMG_NUM = len(os.listdir(IMG_PATH + CLASS))
        for (n, FILE_NAME) in enumerate(os.listdir(IMG_PATH + CLASS)):
            img = IMG_PATH + CLASS + '/' + FILE_NAME
            if n < 5:
                shutil.copy(img, 'TEST/' + CLASS.upper() + '/' + FILE_NAME)
            elif n < 0.8*IMG_NUM:
                shutil.copy(img, 'TRAIN/'+ CLASS.upper() + '/' + FILE_NAME)
            else:
                shutil.copy(img, 'VAL/'+ CLASS.upper() + '/' + FILE_NAME)
```

Загрузите изображения из указанной директории, преобразуйте их в массивы NumPy и верните массивы изображений *X*, массивы меток *y* и словарь

labels, где ключами являются уникальные метки классов, а значениями - соответствующие числа.

Переберите все файлы в директории и для каждого файла считайте изображение с помощью OpenCV, добавьте его в список *X* и соответствующую метку в список *Y*. В результате, *X* содержит все загруженные изображения в виде массива NumPy, а *Y* содержит соответствующие метки для каждого изображения. Словарь *labels* используется для отображения меток классов в виде чисел.

Также отобразите матрицу запутанностей в виде тепловой карты с помощью библиотеки Matplotlib.

```
def load_data(dir_path, img_size=(100,100)):
    """
    Load resized images as np.arrays to workspace
    """
    X = []
    y = []
    i = 0
    labels = dict()
    for path in tqdm(sorted(os.listdir(dir_path))):
        if not path.startswith('.'):
            labels[i] = path
            for file in os.listdir(dir_path + path):
                if not file.startswith('.'):
                    img = cv2.imread(dir_path + path + '/' + file)
                    X.append(img)
                    y.append(i)
            i += 1
    X = np.array(X)
    y = np.array(y)
    print(f'{len(X)} images loaded from {dir_path} directory.')
    return X, y, labels
```

Настройте директории для обучающих, тестовых и валидационных данных и задайте размер изображений.

```
TRAIN_DIR = 'TRAIN/'
TEST_DIR = 'TEST/'
VAL_DIR = 'VAL/'
IMG_SIZE = (224,224)

X_train, y_train, labels = load_data(TRAIN_DIR, IMG_SIZE)
X_test, y_test, _ = load_data(TEST_DIR, IMG_SIZE)
X_val, y_val, _ = load_data(VAL_DIR, IMG_SIZE)
```

Обрежьте изображения в наборе данных. Функция обрезки принимает набор изображений (set_name) и необязательный параметр add_pixels_value,

который определяет, сколько пикселей добавить к кадрированному изображению. Для каждого изображения в наборе данных выполняются следующие шаги:

– Изображение преобразуется в оттенки серого с помощью функции cv2.cvtColor(), а затем применяется фильтр Гаусса для сглаживания шума.

– Изображение в оттенках серого подвергается пороговому значению, а затем выполняются операции для удаления любых небольших областей шума.

– На пороговом изображении находятся контуры, а затем выбирается контур с максимальной площадью.

– Находятся крайние точки (левая, правая, верхняя, нижняя) контура с максимальной площадью.

– Создается новое обрезанное изображение, основанное на крайних точках, с добавлением пикселей, указанных в *add_pixels_value*. Новое изображение добавляется в список *set_new*.

– После обработки всех изображений в наборе данных список *set_new* преобразуется в массив NumPy и возвращается функцией.

```
def crop_imgs(set_name, add_pixels_value=0):
    set_new = []
    for img in set_name:
        gray = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
        gray = cv2.GaussianBlur(gray, (5, 5), 0)
        thresh = cv2.threshold(gray, 45, 255, cv2.THRESH_BINARY)[1]
        thresh = cv2.erode(thresh, None, iterations=2)
        thresh = cv2.dilate(thresh, None, iterations=2)
        cnts = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
        cnts = imutils.grab_contours(cnts)
        c = max(cnts, key=cv2.contourArea)
        extLeft = tuple(c[c[:, :, 0].argmin()][0])
        extRight = tuple(c[c[:, :, 0].argmax()][0])
        extTop = tuple(c[c[:, :, 1].argmin()][0])
        extBot = tuple(c[c[:, :, 1].argmax()][0])
        ADD_PIXELS = add_pixels_value
        new_img = img[extTop[1]-ADD_PIXELS:extBot[1]+ADD_PIXELS, extLeft[0]-
ADD_PIXELS:extRight[0]+ADD_PIXELS].copy()
        set_new.append(new_img)
    return np.array(set_new)
```

Создайте визуальное представление каждого этапа обработки изображения с помощью matplotlib. Оно должно отображать исходное изображение, изображение с найденным контуром, изображение с найденными крайними

точками и окончательное обрезанное изображение (рисунок 1.5).

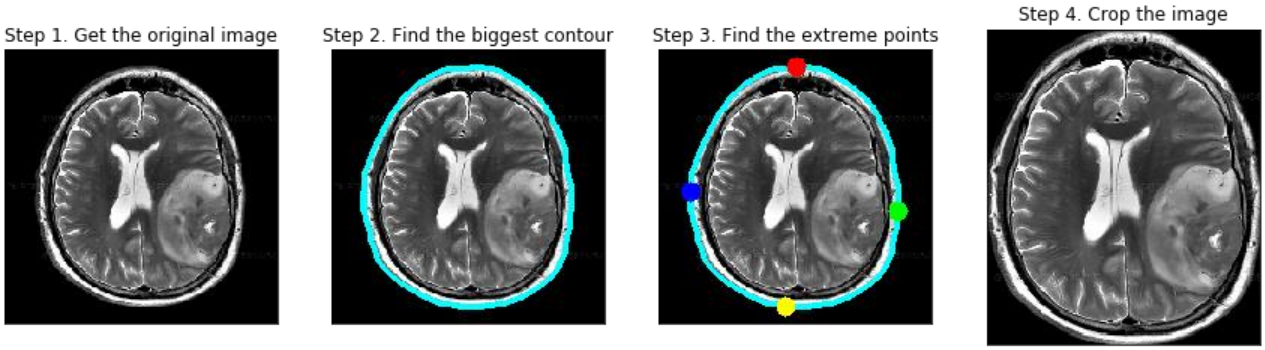


Рисунок 1.5 – Обрезка изображений

Используйте функцию crop_imgs(), чтобы обрезать все изображения в обучающем, валидационном и тестовом наборах данных. Сохраните новые изображения в директорию

```
!mkdir TRAIN_CROP TEST_CROP VAL_CROP TRAIN_CROP/YES TRAIN_CROP/NO
TEST_CROP/YES TEST_CROP/NO VAL_CROP/YES VAL_CROP/NO
save_new_images(X_train_crop, y_train, folder_name='TRAIN_CROP/')
save_new_images(X_val_crop, y_val, folder_name='VAL_CROP/')
save_new_images(X_test_crop, y_test, folder_name='TEST_CROP/')
```

Используйте функцию preprocess_imgs() для предварительной обработки обрезанных изображений в обучающем, валидационном и тестовом наборах данных. Функция preprocess_imgs() принимает набор изображений и размер изображения и возвращает новый набор предварительно обработанных изображений.

– *X_train_prep* — это новый набор предварительно обработанных изображений из обучающего набора данных.

– *X_test_prep* — это новый набор предварительно обработанных изображений из тестового набора данных.

– *X_val_prep* — это новый набор предварительно обработанных изображений из валидационного набора данных.

```
X_train_prep = preprocess_imgs(set_name=X_train_crop, img_size=IMG_SIZE)
X_test_prep = preprocess_imgs(set_name=X_test_crop, img_size=IMG_SIZE)
X_val_prep = preprocess_imgs(set_name=X_val_crop, img_size=IMG_SIZE)
```

Создайте объект ImageDataGenerator из библиотеки Keras, который используется для генерации изображений в режиме реального времени с различными преобразованиями. Объект demo_datagen настроен со следующими параметрами:

– *rotation_range* = 15: Максимальная степень поворота изображения в градусах. В данном случае изображения могут поворачиваться на 15 градусов в любом направлении.

– *width_shift_range* = 0.05: Максимальное значение сдвига изображения по ширине. В данном случае изображения могут сдвигаться на 5% своей ширины.

– *height_shift_range* = 0.05: Максимальное значение сдвига изображения по высоте. В данном случае изображения могут сдвигаться на 5% своей высоты.

– *rescale* = 1./255: Фактор масштабирования пиксельных значений. В данном случае пиксельные значения умножаются на 1/255, чтобы привести их в диапазон от 0 до 1.

– *shear_range* = 0.05: Максимальная степень сдвига (наклона) изображения. В данном случае изображения могут быть сдвинуты на 5% своей высоты или ширины.

– *brightness_range* = [0.1, 1.5]: Диапазон изменения яркости. В данном случае яркость изображения может быть уменьшена до 10% или увеличена до 150% исходной яркости.

– *horizontal_flip* = True: Разрешает горизонтальное отражение (зеркальное отображение) изображений.

– *vertical_flip* = True: Разрешает вертикальное отражение (зеркальное отображение) изображений.

```
demo_datagen = ImageDataGenerator(
    rotation_range=15,
    width_shift_range=0.05,
    height_shift_range=0.05,
```

```
    rescale=1./255,
    shear_range=0.05,
    brightness_range=[0.1, 1.5],
    horizontal_flip=True,
    vertical_flip=True
)
```

Создайте папку preview и сгенерируйте 20 модернизированных версий одного изображения из обучающего набора данных с использованием объекта ImageDataGenerator *demo_datagen*. После выполнения этого кода модернизованные изображения можно использовать для визуальной оценки различных преобразований, выполняемых ImageDataGenerator, и для проверки того, как модель будет реагировать на различные версии одного и того же изображения во время обучения.

```
os.mkdir('preview')
x = X_train_crop[0]
x = x.reshape((1,) + x.shape)
i = 0
for batch in demo_datagen.flow(x, batch_size=1, save_to_dir='preview', save_prefix='aug_img',
save_format='jpg'):
    i += 1
    if i > 20:
        break
```

Отобразите исходное изображение и 21 модернизированное изображение, как показано на рисунке 1.6.

```
plt.imshow(X_train_crop[0])
plt.xticks([])
plt.yticks([])
plt.title('Original Image')
plt.show()
plt.figure(figsize=(15,6))
i = 1
for img in os.listdir('preview/'):
    img = cv2.cv2.imread('preview/' + img)
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    plt.subplot(3,7,i)
    plt.imshow(img)
    plt.xticks([])
    plt.yticks([])
    i += 1
    if i > 3*7:
        break
```

```
plt.suptitle('Augemented Images')
plt.show()
```

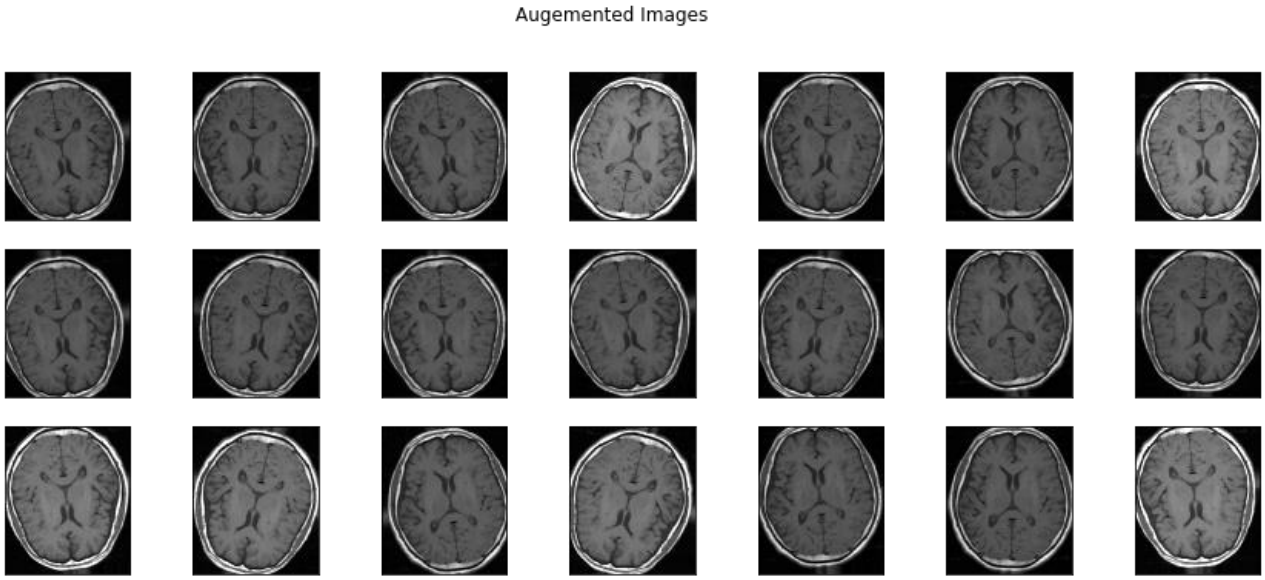


Рисунок 1.6 – Модернизированные изображения

Настройте генераторы изображений для обучающего и валидационного наборов данных с помощью ImageDataGenerator. Это необходимо для генерации изображений в режиме реального времени во время обучения модели.

```
TRAIN_DIR = 'TRAIN_CROP/'
VAL_DIR = 'VAL_CROP/'
train_datagen = ImageDataGenerator(
    rotation_range=15,
    width_shift_range=0.1,
    height_shift_range=0.1,
    shear_range=0.1,
    brightness_range=[0.5, 1.5],
    horizontal_flip=True,
    vertical_flip=True,
    preprocessing_function=preprocess_input
)
test_datagen = ImageDataGenerator(
    preprocessing_function=preprocess_input
)
train_generator = train_datagen.flow_from_directory(
    TRAIN_DIR,
    color_mode='rgb',
    target_size=IMG_SIZE,
    batch_size=32,
    class_mode='binary',
    seed=RANDOM_SEED
)
validation_generator = test_datagen.flow_from_directory(
```

```
    VAL_DIR,
    color_mode='rgb',
    target_size=IMG_SIZE,
    batch_size=16,
    class_mode='binary',
    seed=RANDOM_SEED
)
```

Загрузите предварительно обученную модель VGG-16 без верхних полносвязных слоев (include_top=False) с помощью библиотеки Keras. Модель загружается с помощью предварительно обученных весов.

Создайте модель классификации на основе предварительно обученной модели VGG-16. Модель можно использовать для обучения с помощью генераторов изображений *train_generator* и *validation_generator*, созданных ранее.

```
NUM_CLASSES = 1
model = Sequential()
model.add(base_model)
model.add(layers.Flatten())
model.add(layers.Dropout(0.5))
model.add(layers.Dense(NUM_CLASSES, activation='sigmoid'))
model.layers[0].trainable = False
model.compile(
    loss='binary_crossentropy',
    optimizer=RMSprop(lr=1e-4),
    metrics=['accuracy']
)
model.summary()
```

Обучите модель на обучающих данных с использованием EarlyStopping для прекращения обучения, если модель не показывает улучшения в течение 6 эпох.

```
EPOCHS = 30
es = EarlyStopping(
    monitor='val_acc',
    mode='max',
    patience=6
)
history = model.fit_generator(
    train_generator,
    steps_per_epoch=50,
    epochs=EPOCHS,
    validation_data=validation_generator,
    validation_steps=25,
```

```
    callbacks=[es]
)
```

Визуализируйте результаты обучения модели с помощью matplotlib. Постройте два графика: один для точности (рисунок 1.7) и один для функции потерь (рисунок 1.8).

```
acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs_range = range(1, len(history.epoch) + 1)
plt.figure(figsize=(15,5))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Train Set')
plt.plot(epochs_range, val_acc, label='Val Set')
plt.legend(loc="best")
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.title('Model Accuracy')
plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Train Set')
plt.plot(epochs_range, val_loss, label='Val Set')
plt.legend(loc="best")
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.title('Model Loss')
plt.tight_layout()
plt.show()
```

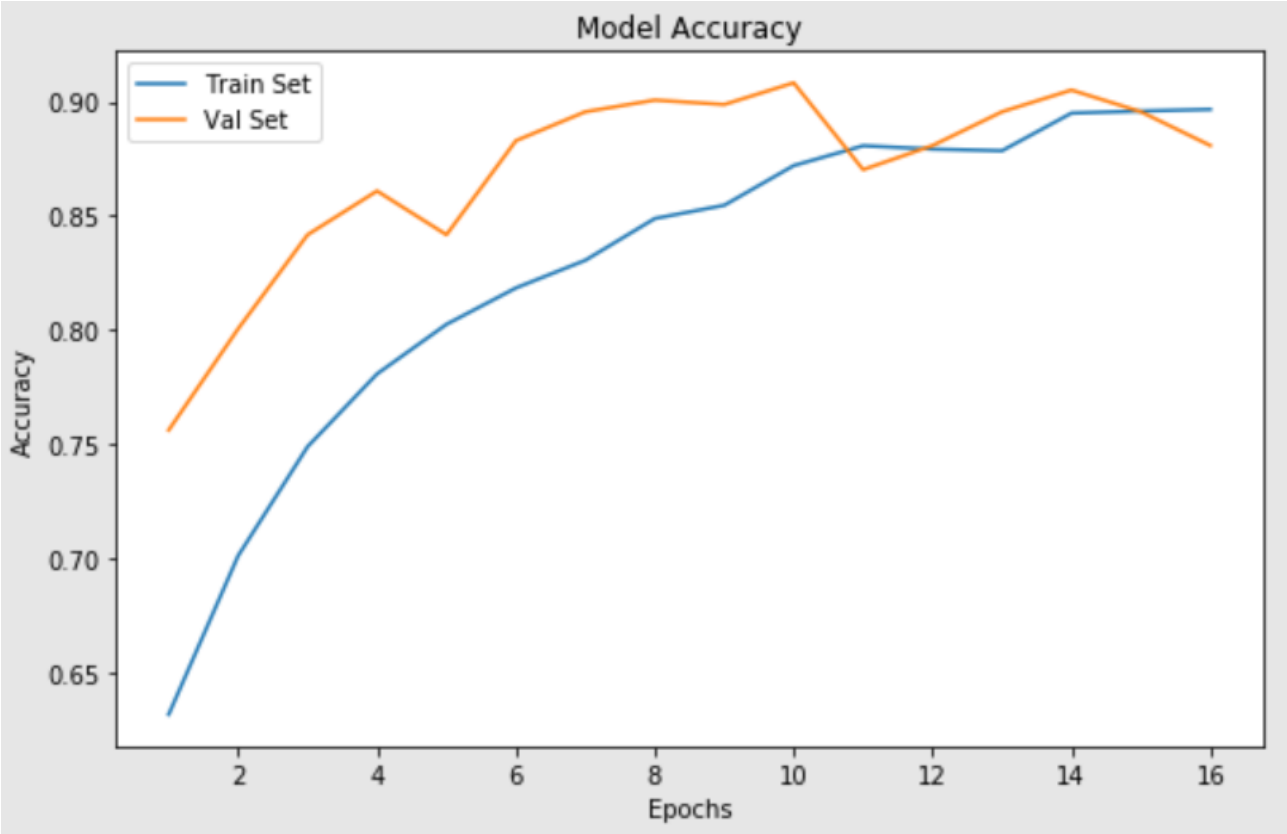


Рисунок 1.7 – График точности

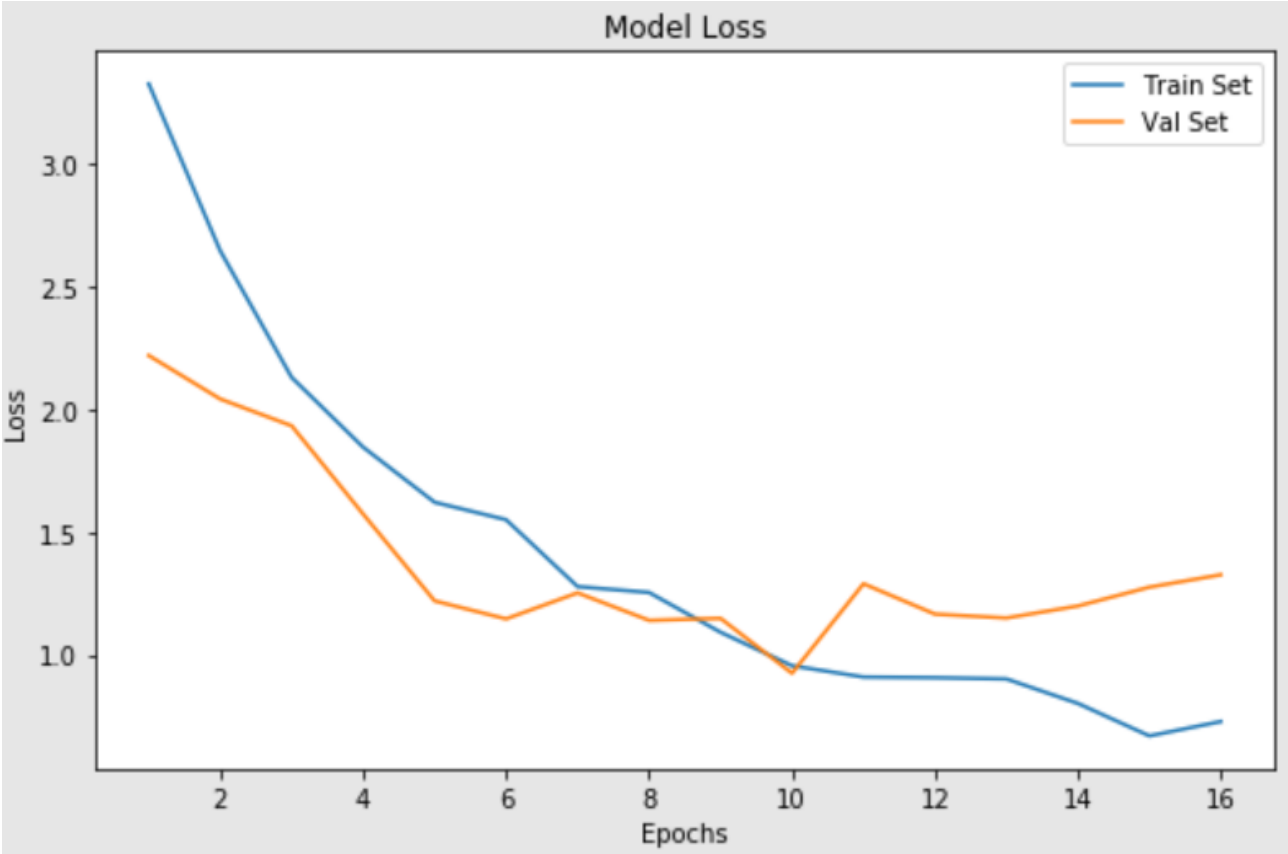


Рисунок 1.8 – График потерь

Постройте матрицу запутанностей на основе валидационных данных. Пример изображен на рисунке 1.9

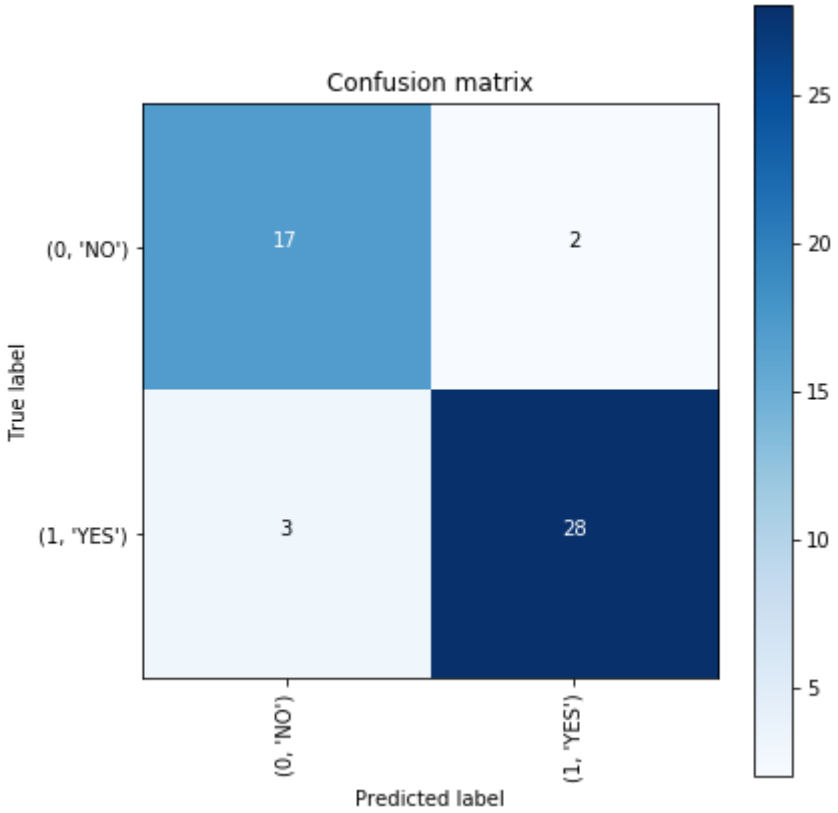


Рисунок 1.9 – Матрица запутанностей для валидационных данных

Постройте матрицу запутанностей на основе тестовых данных. Пример изображен на рисунке 1.10.

```
predictions = model.predict(X_test_prep)
predictions = [1 if x>0.5 else 0 for x in predictions]
accuracy = accuracy_score(y_test, predictions)
print('Test Accuracy = %.2f' % accuracy)
confusion_mtx = confusion_matrix(y_test, predictions)
cm = plot_confusion_matrix(confusion_mtx, classes = list(labels.items()), normalize=False)
```

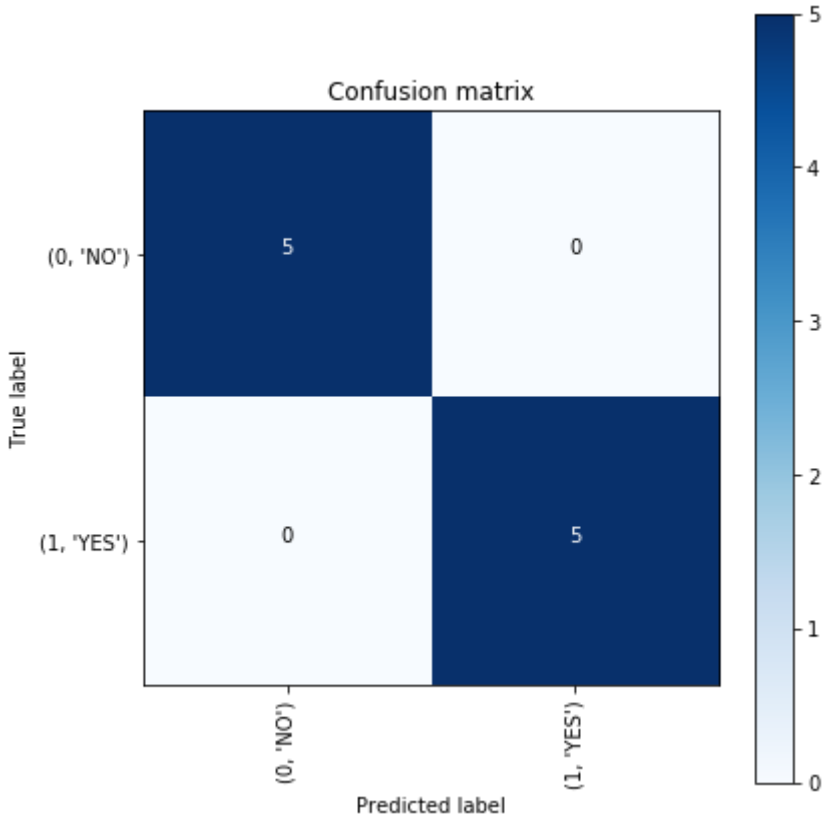


Рисунок 1.10 – Матрица запутанностей для тестовых данных

Контрольные вопросы

1. Какие виды нейронных сетей существуют?
2. В чем заключаются основные отличия VGG-16 и EfficientNet?
3. Почему необходимо при обработке изображений найти крайние точки и, при необходимости, обрезать изображение?
4. В чем различия валидационных и тестовых данных?

ЛАБОРАТОРНАЯ РАБОТА № 3

ОТБОР ПРИЗНАКОВ ДЛЯ ЭФФЕКТИВНОГО ПРОГНОЗИРОВАНИЯ ИСХОДОВ ЛЕЧЕНИЯ ПАЦИЕНТОВ

Целью данной лабораторной работы является построение модели прогнозирования на основе отобранных признаков и оценка точности и надежности полученной модели.

Краткие теоретические сведения

В современной медицине прогнозирование исходов лечения является ключевым аспектом медицинской помощи, поскольку оно позволяет врачам принимать обоснованные решения о методах лечения и обеспечивать оптимальную медицинскую помощь пациентам. Развитие информационных технологий и искусственного интеллекта создало новые возможности для решения этой проблемы, позволяя создавать точные и надежные модели прогнозирования на основе данных о пациентах.

Болезни сердца, также называемые сердечно-сосудистыми заболеваниями - это широкий термин, используемый для обозначения заболеваний и состояний, влияющих на сердце и систему кровообращения. Это одна из основных причин инвалидности во всем мире. Поскольку сердце является одним из наиболее важных органов тела, его заболевания также влияют на другие органы и части тела. Существует несколько различных типов и форм сердечных заболеваний. Наиболее распространенные из них вызывают сужение или закупорку коронарных артерий, нарушение работы клапанов сердца, увеличение размеров сердца и некоторые другие, приводящие к сердечной недостаточности и сердечному приступу.

Требования к выполнению лабораторной работы

1. Ознакомиться с методическими указаниями;

2. Выполнить ход работы;

3. Самостоятельно провести анализ данных, ссылка на которые представлена в курсе.

4. Ответить на контрольные вопросы.

1 Ход работы

Перед началом работы необходимо импортировать библиотеки для визуализации данных.

```
import numpy as np
import pandas as pd
from scipy import stats
import seaborn as sns
from IPython.core.display import HTML
import matplotlib.pyplot as plt
from scipy.stats import uniform
import warnings
warnings.filterwarnings('ignore')
import os
```

Далее необходимо узнать формат данных в файле набора данных (директория /kaggle/input/heart-disease/heart.csv), а также визуализировать несколько их экземпляров. Визуализация данных представлена на рисунке 1.1.

```
data = pd.read_csv('/kaggle/input/heart-disease/heart.csv')
print('Shape of the data is ', data.shape)
Shape of the data is  (303, 14) #Кортеж из 14 столбцов и 303 строк
data.head()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

Рисунок 1.1 – Визуализация данных

С помощью команды data.dtypes можно увидеть типы данных столбцов (рисунок 1.2).

```
age           int64
sex           int64
cp            int64
trestbps      int64
chol          int64
fbs           int64
restecg       int64
thalach       int64
exang         int64
oldpeak     float64
slope         int64
ca            int64
thal          int64
target        int64
dtype: object
```

Рисунок 1.2 –Типы данных признаков

Из типов данных мы видим, что все признаки имеют значение int64 или float64. Это возникает из-за того, что некоторые категориальные признаки, включая информацию о наличии заболевания у пациента, уже закодированы в виде меток. Далее в таблицах 1.1 и 1.2 расшифрованы данные и обозначены их допустимые границы.

Таблица 1.1 – Таблица данных

Переменная	Обозначение	Допустимые границы
age	Возраст	Любой
sex	Пол	1 = мужчина 0 = женщина

Продолжение таблицы 1.1

Переменная	Обозначение	Допустимые границы
cp	Боль в груди	0: типичная стенокардия 1: атипичная стенокардия 2: боль, не связанная со стенокардией 3: бессимптомная
trestbps	Артериальное давление в состоянии покоя	мм.рт.ст. при поступлении в больницу
chol	Уровень холестерина в сыворотке крови	миллиграмм/децалитр
fbs	Уровень сахара в крови натощак > 120 мг/л	1 = да 0 = нет
restecg	Результаты электрокардиографии в состоянии покоя	0: в норме 1: аномалия зубца ST-T (инверсия зубца T и/или подъем или понижение ST > 0,05) 2: указывает на вероятную или определенную гипертрофию левого желудочка по критериям Эстеса.
thalach	Максимально достигнутая частота сердечных сокращений	Любой
exang	Стенокардия, вызванная физической нагрузкой	1 = да 0 = нет
oldpeak	Депрессия ST, вызванная физической нагрузкой, относительно наклона в покое	0: с наклоном вверх 1: без изменений 2: с наклоном вниз
slope	Наклон сегмента ST на пике физической нагрузки	Любой
ca	Количество крупных сосудов, окрашенных методом флюороскопии	От 0 до 3
thal	Результат эхокардиографии	0 = ошибка (в исходном наборе данных 0 соответствует NaN) 1 = исправленный дефект 2 = нормальный 3 = обратимый дефект
target	Метка	0 = болезни нет 1 = болезнь есть

В ходе проверки корректности данных можно заметить следующее:

– данные №93, 159, 164, 165 и 252 имеют значение ca=4, что не соответствует допустимым значениям согласно таблице 1.1;

– данные №49 и 282 имеют значение thal = 0, что также неверно, они также являются NAN в исходном наборе данных.

Следовательно, необходимо убрать данные строки из набора данных. Таким образом набор данных составит 296 экземпляров вместо 303.

```
data = data[data['ca'] < 4]
data = data[data['thal'] > 0]
print(f'The length of the data now is {len(data)} instead of 303')
```

Следующим шагом переименуйте столбцы для большей ясности.

```
data = data.rename(
   columns = {'cp':'chest_pain_type',
            'trestbps':'resting_blood_pressure',
            'chol': 'cholesterol',
            'fbs': 'fasting_blood_sugar',
            'restecg' : 'resting_electrocardiogram',
            'thalach': 'max_heart_rate_achieved',
            'exang': 'exercise_induced_angina',
            'oldpeak': 'st_depression',
            'slope': 'st_slope',
            'ca':'num_major_vessels',
            'thal': 'thalassemia'},
   errors="raise")
```

```
data['sex'][data['sex'] == 0] = 'female'
data['sex'][data['sex'] == 1] = 'male'
data['chest_pain_type'][data['chest_pain_type'] == 0] = 'typical angina'
data['chest_pain_type'][data['chest_pain_type'] == 1] = 'atypical angina'
data['chest_pain_type'][data['chest_pain_type'] == 2] = 'non-anginal pain'
data['chest_pain_type'][data['chest_pain_type'] == 3] = 'asymptomatic'
data['fasting_blood_sugar'][data['fasting_blood_sugar'] == 0] = 'lower than 120mg/ml'
data['fasting_blood_sugar'][data['fasting_blood_sugar'] == 1] = 'greater than 120mg/ml'
data['resting_electrocardiogram'][data['resting_electrocardiogram'] == 0] = 'normal'
data['resting_electrocardiogram'][data['resting_electrocardiogram'] == 1] = 'ST-T wave abnormality'
data['resting_electrocardiogram'][data['resting_electrocardiogram'] == 2] = 'left ventricular hypertrophy'
data['exercise_induced_angina'][data['exercise_induced_angina'] == 0] = 'no'
data['exercise_induced_angina'][data['exercise_induced_angina'] == 1] = 'yes'
data['st_slope'][data['st_slope'] == 0] = 'upsloping'
data['st_slope'][data['st_slope'] == 1] = 'flat'
data['st_slope'][data['st_slope'] == 2] = 'downsloping'
data['thalassemia'][data['thalassemia'] == 1] = 'fixed defect'
data['thalassemia'][data['thalassemia'] == 2] = 'normal'
data['thalassemia'][data['thalassemia'] == 3] = 'reversable defect'
```

С помощью команды data.dtypes выведите названия столбцов и их типы данных (рисунок 1.3). Также выведите таблицу данных (рисунок 1.4).

```
age                          int64
sex                         object
chest_pain_type             object
resting_blood_pressure       int64
cholesterol                  int64
fasting_blood_sugar         object
resting_electrocardiogram   object
max_heart_rate_achieved      int64
exercise_induced_angina     object
st_depression              float64
st_slope                    object
num_major_vessels            int64
thalassemia                 object
target                       int64
dtype: object
```

Рисунок 1.3 – Измененные названия данных

	age	sex	chest_pain_type	resting_blood_pressure	cholesterol	fasting_blood_sugar	resting_electrocardiogram	max_heart_
0	63	male	asymptomatic	145	233	greater than 120mg/ml	normal	150
1	37	male	non-anginal pain	130	250	lower than 120mg/ml	ST-T wave abnormality	187
2	41	female	atypical angina	130	204	lower than 120mg/ml	normal	172
3	56	male	atypical angina	120	236	lower than 120mg/ml	ST-T wave abnormality	178
4	57	female	typical angina	120	354	lower than 120mg/ml	ST-T wave abnormality	163

Рисунок 1.4 – Повторная визуализация данных

В измененном наборе существуют три типа данных: object, int и floats. Сгруппируйте их в соответствии с их типами.

```
# числовые характеристики
num_feats = ['age', 'cholesterol', 'resting_blood_pressure', 'max_heart_rate_achieved', 'st_depression',
'num_major_vessels']
# категориальные (бинарные)
bin_feats = ['sex', 'fasting_blood_sugar', 'exercise_induced_angina', 'target']
# категориальные (мульти-)
nom_feats= ['chest_pain_type', 'resting_electrocardiogram', 'st_slope', 'thalassemia']
cat_feats = nom_feats + bin_feats
```

Рассмотрите статистическую сводку, а также распределение некоторых из данных, начиная с целевого показателя (рисунок 1.5). Можно заметить, что

целевой показатель достаточно сбалансирован: около 46% не имеют сердечных заболеваний и около 54% - с сердечными заболеваниями.

```
mypal= ['#FC05FB', '#FEAEFE', '#FCD2FC','#F3FEFA', '#B4FFE4','#3FFEBA']
plt.figure(figsize=(7, 5),facecolor='#F6F5F4')
total = float(len(data))
ax = sns.countplot(x=data['target'], palette=mypal[1::4])
ax.set_facecolor('#F6F5F4')
for p in ax.patches:
    height = p.get_height()
    ax.text(p.get_x()+p.get_width()/2.,height + 3,'{:1.1f} %'.format((height/total)*100), ha="center",
            bbox=dict(facecolor='none', edgecolor='black', boxstyle='round', linewidth=0.5))
ax.set_title('Target variable distribution', fontsize=20, y=1.05)
sns.despine(right=True)
sns.despine(offset=5, trim=True)
```

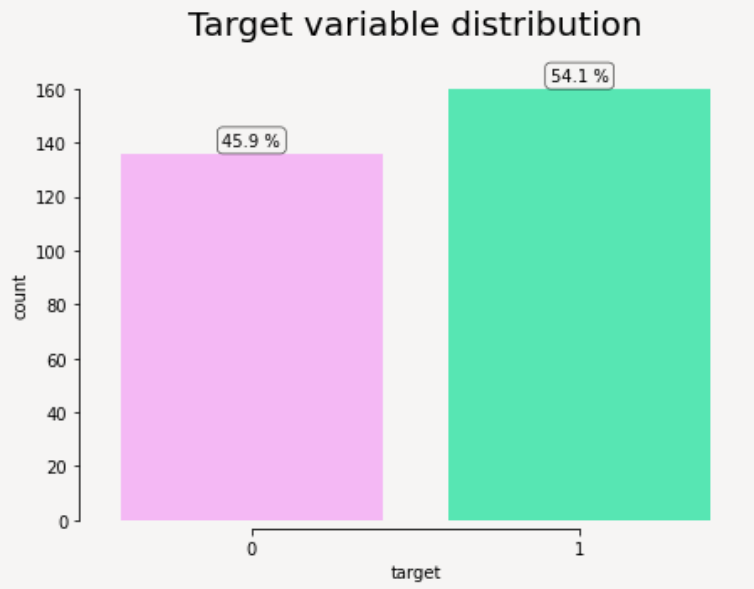


Рисунок 1.5 – Статистическое распределение целевого значения

Чтобы получить глобальную статистическую сводку для числовых характеристик можно применить метод pandas data.describe(). Ниже на рисунке 1.6 приведены ключевые значения.

	count	mean	std	min	25%	50%	75%	max
age	296.0	54.523649	9.059471	29.0	48.0	56.0	61.00	77.0
cholesterol	296.0	247.155405	51.977011	126.0	211.0	242.5	275.25	564.0
resting_blood_pressure	296.0	131.604730	17.726620	94.0	120.0	130.0	140.00	200.0
max_heart_rate_achieved	296.0	149.560811	22.970792	71.0	133.0	152.5	166.00	202.0
st_depression	296.0	1.059122	1.166474	0.0	0.0	0.8	1.65	6.2
num_major_vessels	296.0	0.679054	0.939726	0.0	0.0	0.0	1.00	3.0

Рисунок 1.6 – Статистическая сводка числовых характеристик

Исходя из полученной статистической сводки, можно сделать следующие выводы о числовых характеристиках:

Возраст:

– средний возраст в наборе данных составляет 54,5 года;

– самому старшему из них 77 лет, а самому молодому - 29 лет.

Холестерин:

– средний уровень холестерина составляет 247,15;

– максимальный уровень составляет 564, а минимальный - 126.

Артериальное давление в состоянии покоя:

– среднее значение - 131;

– максимальное - 200;

– минимальное - 94.

Максимальная частота сердечных сокращений:

– средняя максимальная зарегистрированная частота сердечных сокращений составляет 149,5 ударов в минуту;

– максимальная частота – 202 удара в минуту;

– минимальная частота – 71 удар в минут.

Депрессия на сегменте ST:

– среднее значение st_depression равно 1,06;

– максимальное значение равно 6,2;

– минимальное - 0.

Количество крупных кровеносных сосудов:

– наблюдается максимум 3;

– минимум 0 крупных кровеносных сосудов;

– среднее значение равно 0,68.

Рассмотрите распределение числовых характеристик на графике плотности с помощью представленного ниже кода.

```
L = len(num_feats)
ncol= 2
nrow= int(np.ceil(L/ncol))
#remove_last= (nrow * ncol) - L
fig, ax = plt.subplots(nrow, ncol, figsize=(16, 14),facecolor='#F6F5F4')
```

```
fig.subplots_adjust(top=0.92)
i = 1
for col in num_feats:
    plt.subplot(nrow, ncol, i, facecolor='#F6F5F4')
    ax = sns.kdeplot(data=data, x=col, hue="target", multiple="stack", palette=mypal[1::4])
    ax.set_xlabel(col, fontsize=20)
    ax.set_ylabel("density", fontsize=20)
    sns.despine(right=True)
    sns.despine(offset=0, trim=False)

    if col == 'num_major_vessels':
        sns.countplot(data=data, x=col, hue="target", palette=mypal[1::4])
        for p in ax.patches:
            height = p.get_height()
            ax.text(p.get_x()+p.get_width()/2.,height + 3,'{:1.0f}'.format((height)),ha="center",
                bbox=dict(facecolor='none', edgecolor='black', boxstyle='round', linewidth=0.5))
    i = i +1
plt.suptitle('Distribution of Numerical Features' ,fontsize = 24);
```

Результат полученного распределения представлен на рисунке 1.7.

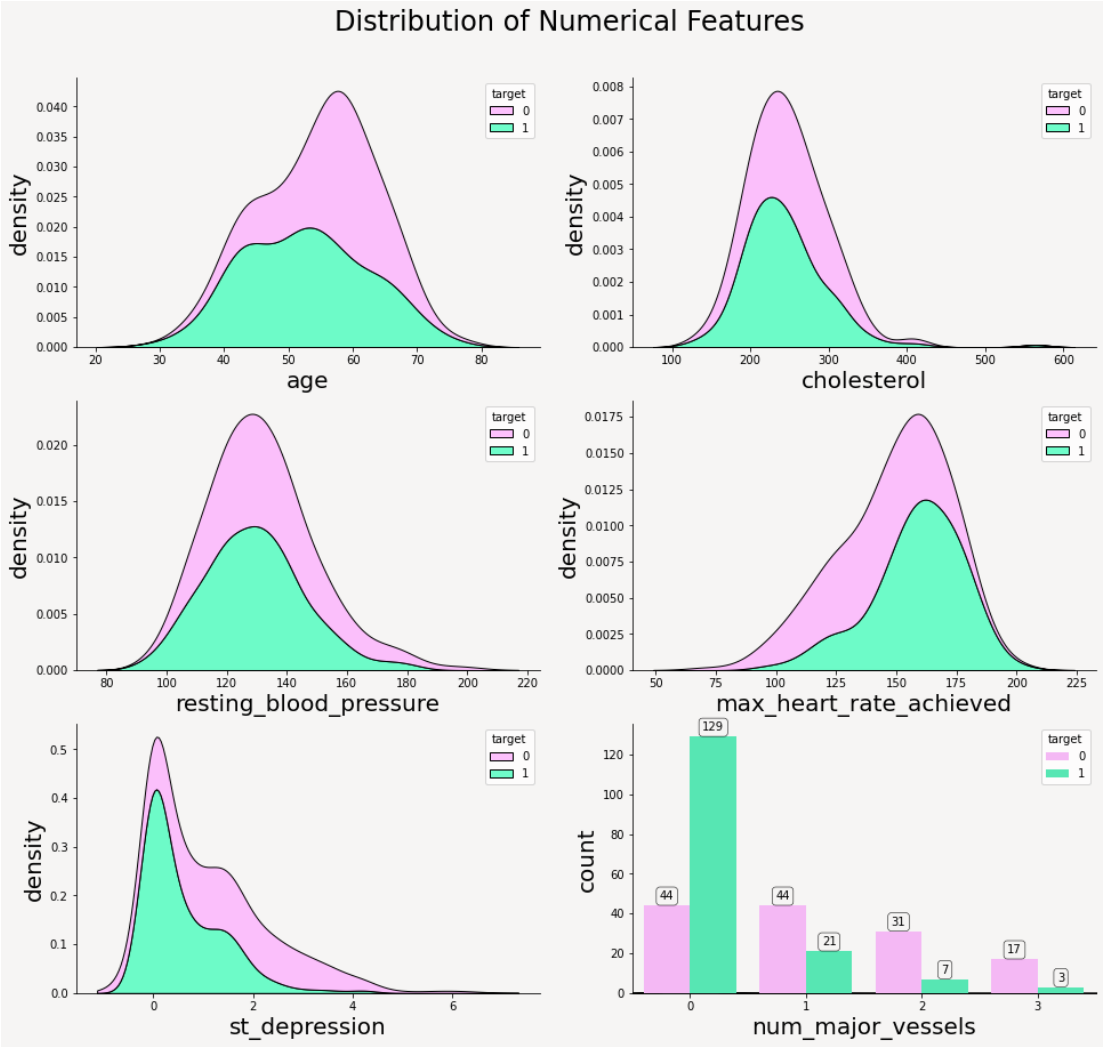


Рисунок 1.7 – Графики плотности числовых характеристик

Также постройте парные графики для данных числовых характеристик (рисунок 1.8).

Рисунок 1.8 – Парные графики для числовых характеристик

Выведите несколько графиков некоторых данных, для выявления линейной зависимости от возраста, аналогичные первому столбцу на парном графике выше.

```
fig, ax = plt.subplots(1,4, figsize=(20, 4))
sns.regplot(data=data[data['target'] ==1], x='age', y='cholesterol', ax = ax[0], color=mypal[0], label='1')
sns.regplot(data=data[data['target'] ==0], x='age', y='cholesterol', ax = ax[0], color=mypal[5], label='0')
sns.regplot(data=data[data['target'] ==1], x='age', y='max_heart_rate_achieved', ax = ax[1], color=mypal[0],
label='1')
sns.regplot(data=data[data['target'] ==0], x='age', y='max_heart_rate_achieved', ax = ax[1], color=mypal[5],
label='0')
sns.regplot(data=data[data['target'] ==1], x='age', y='resting_blood_pressure', ax = ax[2], color=mypal[0],
label='1')
```

```
sns.regplot(data=data[data['target'] ==0], x='age', y='resting_blood_pressure', ax = ax[2], color=mypal[5],
label='0')
sns.regplot(data=data[data['target'] ==1], x='age', y='st_depression', ax = ax[3], color=mypal[0], label='1')
sns.regplot(data=data[data['target'] ==0], x='age', y='st_depression', ax = ax[3], color=mypal[5], label='0')
plt.suptitle('Reg plots of selected features')
plt.legend();
```

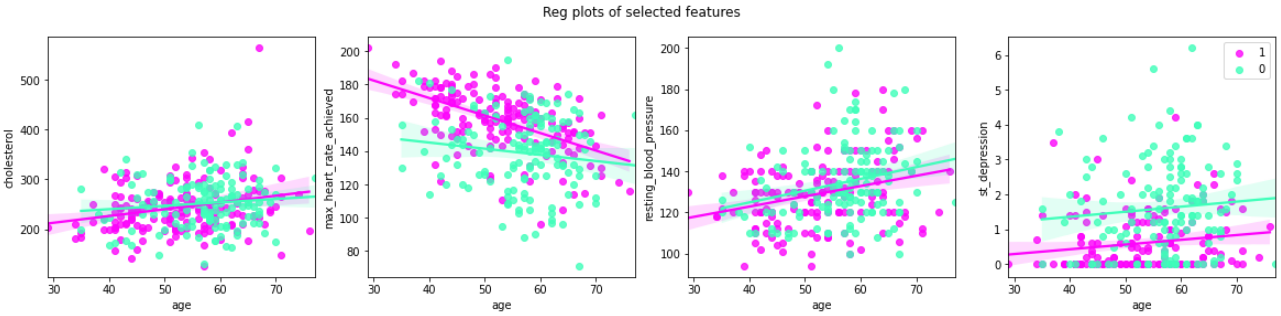


Рисунок 1.9 – Графики выбранных данных

Исходя из этих графиков можно заметить следующее:

– За исключением maximum_heart_rate_achieved, остальные показатели положительно и линейно связаны с возрастом (хотя и более слабая связь с st_depression);

– У более молодых пациентов с более высокой максимальной частотой сердечных сокращений вероятность сердечных заболеваний выше;

– Более низкая частота сердечных сокращений независимо от возраста также может свидетельствовать о сердечных заболеваниях.

Далее используйте график подсчета для визуализации различных категориальных признаков по отношению к целевой переменной. Две вещи, на которые можно обратить внимание это распределение каждой категории в наборе данных и их вклад в вероятность правильного прогнозирования целевой переменной. Например, наличие заболевания (=1) или отсутствие заболевания (=0). Ниже приводится краткое описание категориальных признаков.

Боль в груди:

– Более 75% пациентов испытывают боль в груди, характерную как для стенокардии, так и для других заболеваний;

– Пациенты, у которых наблюдалась атипичная стенокардия или боль в груди, не связанная со стенокардией, чаще страдают сердечными заболеваниями.

Электрокардиограмма в состоянии покоя:

– Пациентов с гипертрофией левого желудочка меньше всего (около 1,4%). Остальные пациенты с аномалией ST-T делятся примерно в соотношении 50:50 между пациентами с нормальными показателями REC;

– Аномалия ST-T, по-видимому, лучше коррелирует с целевым показателем, т.е. у большинства пациентов, прошедших этот вид теста REC, в конечном итоге обнаружились сердечные заболевания.

Угол наклона сегмента ST:

– У большинства пациентов наблюдается нисходящий или плоский наклон сегмента ST в тесте REC;

– Наклонный угол наклона сегмента ST является явным признаком того, что у пациента может быть сердечное заболевание.

Талассемия:

– У большинства пациентов имеется нормальный или обратимый дефект;

– У пациентов с дефектами талассемии (обратимыми + фиксированными) вероятность сердечных заболеваний ниже. В то время как у пациентов с нормальной талассемией вероятность сердечных заболеваний выше.

Уровень сахара в крови натощак

– Пациенты с низким уровнем сахара в крови натощак (менее 120 мг/мл) составляют большинство в нашем наборе данных, составляющем около 85% выборки;

– Низкий уровень сахара в крови в состоянии покоя увеличивает вероятность сердечных заболеваний (около 54%).

Стенокардия, вызванная физической нагрузкой

– У двух третей пациентов стенокардия, вызванная физической

нагрузкой, не проявлялась;

– У 76% пациентов со стенокардией, вызванной физической нагрузкой, не было проблем с сердцем. В то время как у около 69% пациентов, у которых не было стенокардии, вызванной физической нагрузкой, были диагностированы проблемы с сердцем.

Пол

– Большинство пациентов в выборке — мужчины.

Далее используйте корреляционную тепловую карту. Корреляционная тепловая карта – это полезный инструмент для графического представления того, как два объекта связаны друг с другом. В зависимости от типов данных объектов нам необходимо использовать соответствующие методы расчета коэффициента корреляции. Коэффициент корреляции Пирсона — это мера линейной корреляции между двумя наборами данных. Это соотношение между ковариацией двух переменных и произведением их стандартных отклонений; таким образом, это нормализованное измерение корреляционного момента, так что результат всегда имеет значение от -1 до 1. Визуальное представление коэффициента корреляции Пирсона представлено на рисунке 1.10.

```
df_ = data[num_feats]
corr = df_.corr(method='pearson')
mask = np.triu(np.ones_like(corr, dtype=bool))
f, ax = plt.subplots(figsize=(8, 5), facecolor=None)
cmap = sns.color_palette(mypal, as_cmap=True)
sns.heatmap(corr, mask=mask, cmap=cmap, vmax=1.0, vmin=-1.0, center=0, annot=True,
        square=False, linewidths=.5, cbar_kws={"shrink": 0.75})
ax.set_title("Numerical features correlation (Pearson's)", fontsize=20, y= 1.05);
```

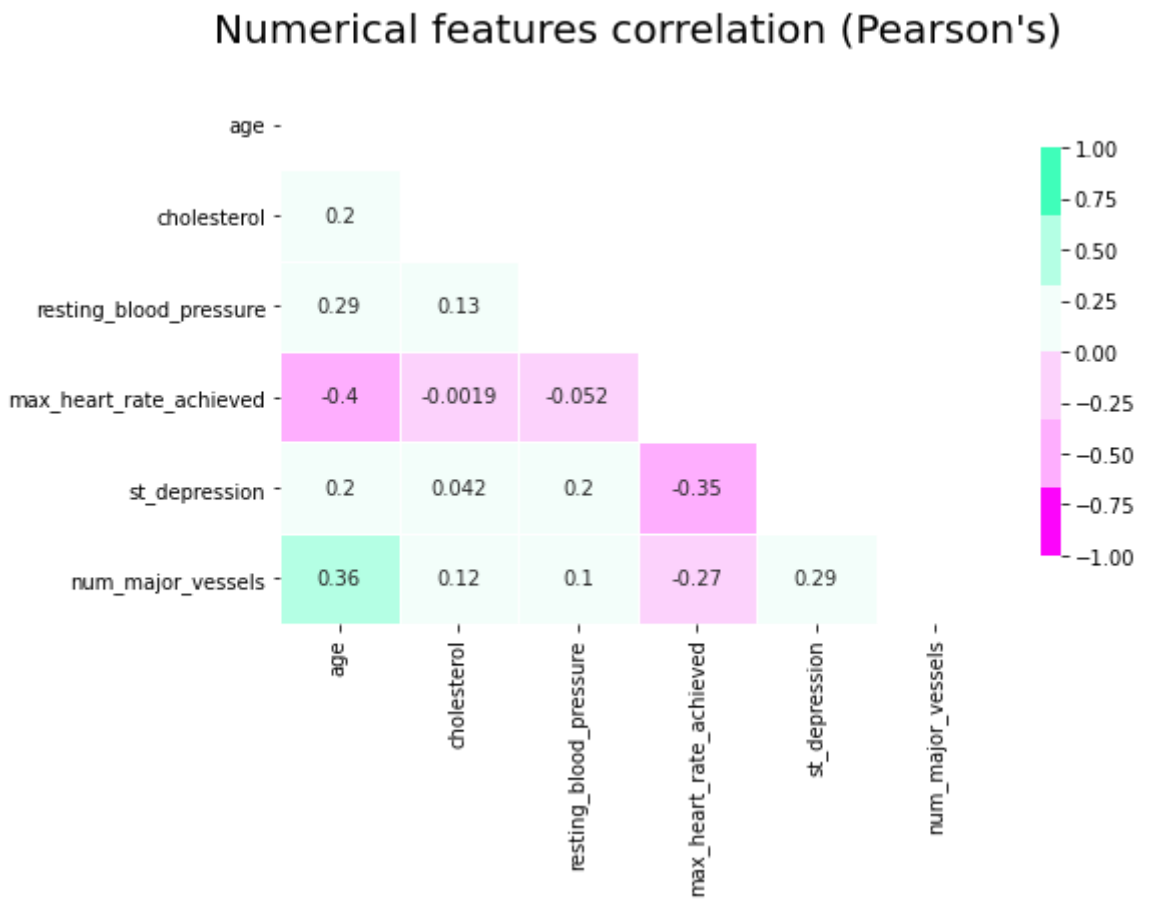


Рисунок 1.10 – Коэффициент корреляции Пирсона

Точечно-бисериальная корреляция используется для измерения силы и направления связи, существующей между одной непрерывной переменной и одной дихотомической переменной. Это частный случай корреляции произведения и момента Пирсона, которая применяется, когда есть две непрерывные переменные, тогда как в данном случае одна из переменных измеряется по дихотомической шкале. Визуальное представление точечно-бисерной корелляции представлено на рисунке 1.11.

```
feats_ = ['age', 'cholesterol', 'resting_blood_pressure', 'max_heart_rate_achieved', 'st_depression',
'num_major_vessels', 'target']
def point_biserial(x, y):
    pb = stats.pointbiserialr(x, y)
    return pb[0]
rows= []
for x in feats_:
    col = []
    for y in feats_ :
        pbs =point_biserial(data[x], data[y])
        col.append(round(pbs,2))
    rows.append(col)
pbs_results = np.array(rows)
```

```
DF = pd.DataFrame(pbs_results, columns = data[feats_].columns, index =data[feats_].columns)
mask = np.triu(np.ones_like(DF, dtype=bool))
corr = DF.mask(mask)
f, ax = plt.subplots(figsize=(8, 5), facecolor=None)
cmap = sns.color_palette(mypal, as_cmap=True)
sns.heatmap(corr, mask=mask, cmap=cmap, vmax=1.0, vmin=-1, center=0, annot=True,
            square=False, linewidths=.5, cbar_kws={"shrink": 0.75})
ax.set_title("Cont feats vs target correlation (point-biserial)", fontsize=20, y= 1.05);
```

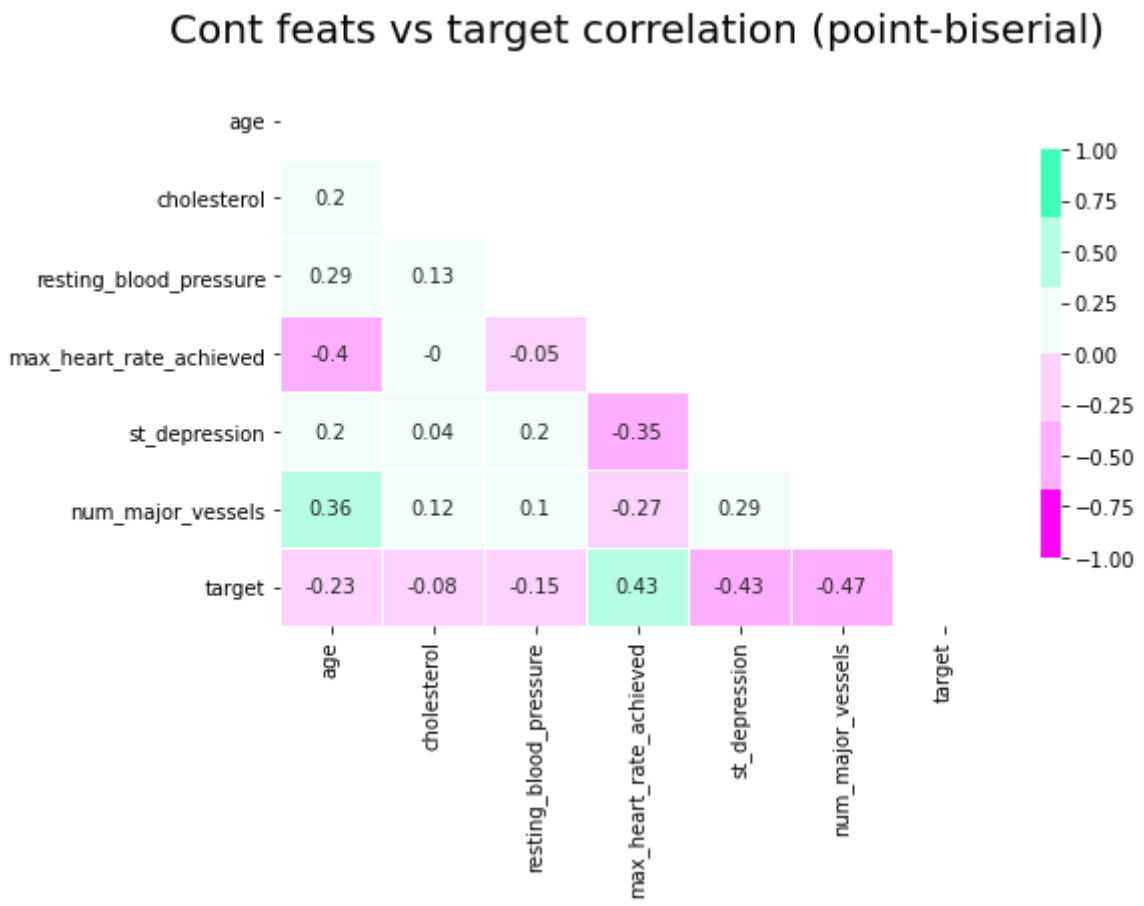


Рисунок 1.11 – Точечно-бисериальная корреляция

В статистике, корреляция V Крамера – это показатель связи между двумя номинальными переменными, который дает значение от 0 до 1 включительно. Он основан на статистике Пирсона хи-квадрат и был опубликован Харальдом Крамером в 1946 году. Визуальное представление корелляции V Крамера представлено на рисунке 1.12.

```
def cramers_v(x, y):
    confusion_matrix = pd.crosstab(x,y)
    chi2 = stats.chi2_contingency(confusion_matrix)[0]
    n = confusion_matrix.sum().sum()
    phi2 = chi2/n
    r,k = confusion_matrix.shape
    phi2corr = max(0, phi2-((k-1)*(r-1))/(n-1))
```

```
    rcorr = r-((r-1)**2)/(n-1)
    kcorr = k-((k-1)**2)/(n-1)
    return np.sqrt(phi2corr/min((kcorr-1),(rcorr-1)))
rows= []
for x in data_:
    col = []
    for y in data_ :
        cramers =cramers_v(data_[x], data_[y])
        col.append(round(cramers,2))
    rows.append(col)
cramers_results = np.array(rows)
df = pd.DataFrame(cramers_results, columns = data_.columns, index = data_.columns)
mypal_1= ['#FC05FB', '#FEAEFE', '#FCD2FC','#F3FEFA', '#B4FFE4','#3FFEBA', '#FC05FB', '#FEAEFE',
'#FCD2FC']
mask = np.triu(np.ones_like(df, dtype=bool))
corr = df.mask(mask)
f, ax = plt.subplots(figsize=(10, 6), facecolor=None)
cmap = sns.color_palette(mypal_1, as_cmap=True)
sns.heatmap(corr, mask=mask, cmap=cmap, vmax=1.0, vmin=0, center=0, annot=True,
            square=False, linewidths=.01, cbar_kws={"shrink": 0.75})
ax.set_title("Categorical Features Correlation (Cramer's V)", fontsize=20, y= 1.05);
```

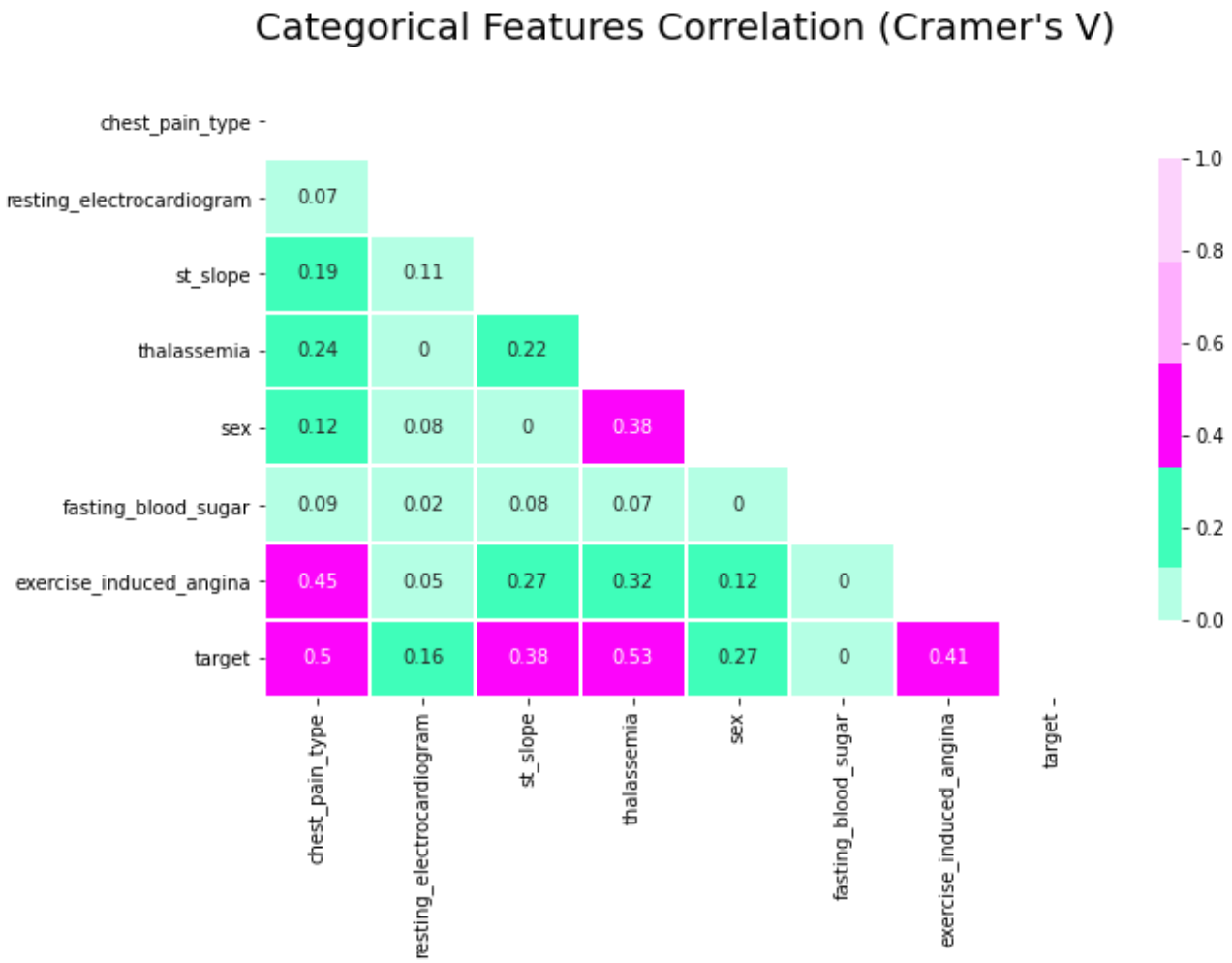


Рисунок 1.12 – Корреляция V Крамера

Таким образом объем данных составил 296 строки и 14 столбцов (13 независимых + 1 целевая переменная).

Тип данных объектов (столбцов):

– Шесть объектов являются числовыми;

– Остальные семь признаков являются категориальными переменными;

– Целевая переменная достаточно сбалансирована, от 54% случаев отсутствия заболеваний до 46% случаев наличия заболеваний.

Корреляции:

– Корреляция между признаками слабая;

– Судя по числовым характеристикам num_major_vessels, max_heart_rate_achieved и st_depression, они достаточно точно коррелируют с целевой переменной с коэффициентом корреляции -0,47, 0,43 и -0,43 соответственно;

– Из категориальных признаков chest_pain_type, num_major_vessels, талассемия и exercise_induced_angina лучше коррелируют с целевой переменной, при этом талассемия имеет самый высокий показатель - 0,52;

– Холестерин в меньшей степени связан с сердечными заболеваниями.

Scikit learn предлагает широкий спектр алгоритмов классификации и часто является отправной точкой в решении большинства традиционных задач машинного обучения. Необходимо начать с изучения нескольких алгоритмов классификации из библиотеки sklearn, таких как Логистическая регрессия, Ближайшие соседи, метод опорных векторов, Nu SVC, Случайный лес, AdaBoost, Градиент Бустинг, наивный байесовский анализ, линейный дискриминантный анализ, квадратичный дискриминантный анализ и нейронная сеть. Сначала постройте простые модели, а затем оптимизируйте их, настроив параметры.

```
def label_encode_cat_features(data, cat_features):
    label_encoder = LabelEncoder()
    data_encoded = data.copy()
```

```
    for col in cat_features:
        data_encoded[col] = label_encoder.fit_transform(data[col])
    data = data_encoded
    return data

def score_summary(names, classifiers):
    cols=["Classifier", "Accuracy", "ROC_AUC", "Recall", "Precision", "F1"]
    data_table = pd.DataFrame(columns=cols)
    for name, clf in zip(names, classifiers):
        clf.fit(X_train, y_train)
        pred = clf.predict(X_val)
        accuracy = accuracy_score(y_val, pred)
        pred_proba = clf.predict_proba(X_val)[:, 1]
        fpr, tpr, thresholds = roc_curve(y_val, pred_proba)
        roc_auc = auc(fpr, tpr)
        cm = confusion_matrix(y_val, pred)
        # TP/(TP+FN)
        recall = cm[1,1]/(cm[1,1] +cm[1,0])
        # TP/(TP+FP)
        precision = cm[1,1]/(cm[1,1] +cm[0,1])
        # TP/(TP+FP)
        f1 = 2*recall*precision/(recall + precision)
        df = pd.DataFrame([[name, accuracy*100, roc_auc, recall, precision, f1]], columns=cols)
        data_table = data_table.append(df)
    return(np.round(data_table.reset_index(drop=True), 2))

def plot_conf_matrix(names, classifiers, nrows, ncols, fig_a, fig_b):
    fig, axes = plt.subplots(nrows=nrows, ncols=ncols, figsize=(fig_a, fig_b))
    i = 0
    for clf, ax in zip(classifiers, axes.flatten()):
        clf.fit(X_train, y_train)
        plot_confusion_matrix(clf, X_val, y_val, ax=ax)
        ax.title.set_text(names[i])
        i = i + 1
    plt.tight_layout()
    plt.show()

def roc_auc_curve(names, classifiers):
    plt.figure(figsize=(12, 8))
    for name, clf in zip(names, classifiers):
        clf.fit(X_train, y_train)
        pred_proba = clf.predict_proba(X_val)[:, 1]
        fpr, tpr, thresholds = roc_curve(y_val, pred_proba)
        roc_auc = auc(fpr, tpr)
        plt.plot(fpr, tpr, lw=3, label= name +' ROC curve (area = %0.2f)' % (roc_auc))
        plt.plot([0, 1], [0, 1], color='navy', lw=1, linestyle='--')
        plt.xlim([0.0, 1.0])
        plt.ylim([0.0, 1.0])
        plt.xlabel('False Positive Rate')
        plt.ylabel('True Positive Rate')
        plt.title('Receiver operating characteristic (ROC) curves', fontsize=20)
        plt.legend(loc="lower right")
```

```
cat_features = cat_feats
data = label_encode_cat_features(data, cat_features)
seed = 0
```

```
test_size = 0.25
features = data.columns[:-1]
X = data[features]
y = data['target']
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size = test_size, random_state=seed)
names = [
    'Logistic Regression',
    'Nearest Neighbors',
    'Support Vectors',
    'Nu SVC',
    'Decision Tree',
    'Random Forest',
    'AdaBoost',
    'Gradient Boosting',
    'Naive Bayes',
    'Linear DA',
    'Quadratic DA',
    "Neural Net"
]
classifiers = [
    LogisticRegression(solver="liblinear", random_state=seed),
    KNeighborsClassifier(2),
    SVC(probability=True, random_state=seed),
    NuSVC(probability=True, random_state=seed),
    DecisionTreeClassifier(random_state=seed),
    RandomForestClassifier(random_state=seed),
    AdaBoostClassifier(random_state=seed),
    GradientBoostingClassifier(random_state=seed),
    GaussianNB(),
    LinearDiscriminantAnalysis(),
    QuadraticDiscriminantAnalysis(),
    MLPClassifier(random_state=seed),
]
```

Существует несколько показателей, которые можно использовать для оценки эффективности данного алгоритма классификации. Выбор "подходящих" показателей зависит от типа проблемы. Есть случаи, когда, например, точность может быть правильным выбором, а в некоторых других случаях могут больше соответствовать поставленной цели.

Матрица запутанности - это специальная таблица, которая позволяет визуализировать эффективность алгоритма обучения под руководством пользователя. Каждая строка матрицы представляет экземпляры в реальном классе, в то время как каждый столбец представляет экземпляры в прогнозируемом классе. Приведенная ниже таблица на рисунке 1.13 является примером матрицы запутанности для бинарной классификации.

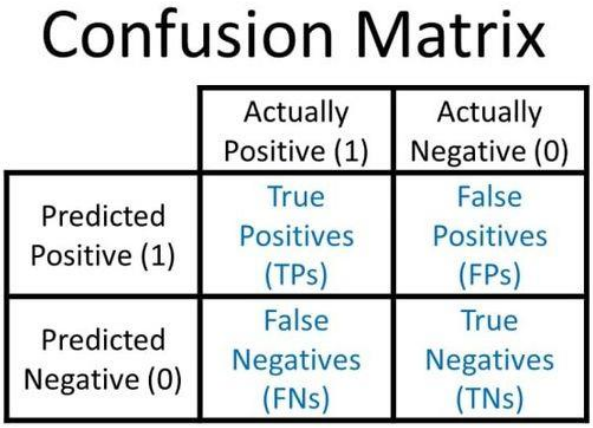


Рисунок 1.13 – Матрица запутанностей

Таблица 1.2 – Переменные

Переменная	Значение	Описание
TP	Истинно положительный	Положительные случаи, которые прогнозируются как положительные
FP	Ложно положительный	Отрицательные случаи, которые прогнозируются как положительные
TN	Истинно отрицательный	Отрицательные случаи, которые прогнозируются как отрицательные
FN	Ложно отрицательный	Положительные случаи, которые прогнозируются как отрицательные

Accuracy: измеряет, сколько случаев правильно идентифицировано/предсказано моделью, т.е. правильный прогноз делится на общий объем выборки.

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \quad (1.1)$$

Recall: измеряет частоту истинных положительных результатов, т.е. сколько реальных положительных случаев идентифицировано/предсказано моделью как положительные.

$$\text{Recall} = \frac{TP}{TP+FN} \quad (1.2)$$

Precision: измеряет, сколько положительных прогнозируемых случаев на

самом деле являются положительными.

$$\text{Precision} = \frac{TP}{TP+FP} \quad (1.3)$$

F1: сочетает в себе точность и отзывчивость модели и определяется как среднее гармоническое значение точности и отзывчивости модели.

$$F1 = 2 \cdot \frac{recall * precision}{recall + precision} \quad (1.4)$$

Ниже представлен фрагмент кода, выводящий вычисленные значения, а также результат на рисунке 1.14.

```
score_summary(names, classifiers).sort_values(by='Accuracy' , ascending = False)\
.style.background_gradient(cmap='coolwarm')\
.bar(subset=["ROC_AUC",], color='#6495ED')\
.bar(subset=["Recall"], color='#ff355d')\
.bar(subset=["Precision"], color='lightseagreen')\
.bar(subset=["F1"], color='gold')
```

	Classifier	Accuracy	ROC_AUC	Recall	Precision	F1
0	Logistic Regression	86.490000	0.920000	0.910000	0.820000	0.860000
9	Linear DA	85.140000	0.920000	0.890000	0.820000	0.850000
10	Quadratic DA	85.140000	0.900000	0.830000	0.850000	0.840000
5	Random Forest	83.780000	0.920000	0.830000	0.830000	0.830000
4	Decision Tree	82.430000	0.820000	0.830000	0.810000	0.820000
6	AdaBoost	82.430000	0.860000	0.910000	0.760000	0.830000
7	Gradient Boosting	82.430000	0.900000	0.890000	0.780000	0.830000
8	Naive Bayes	82.430000	0.920000	0.860000	0.790000	0.820000
3	Nu SVC	81.080000	0.910000	0.910000	0.740000	0.820000
11	Neural Net	78.380000	0.880000	0.940000	0.700000	0.800000
2	Support Vectors	64.860000	0.800000	0.890000	0.580000	0.700000
1	Nearest Neighbors	55.410000	0.600000	0.310000	0.550000	0.400000

Рисунок 1.14 – Вычисленные значения

Кривая рабочих характеристик приемника (ROC) представляет собой график, который иллюстрирует эффективность алгоритма бинарной классификации в зависимости от частоты положительных и ложных срабатываний (рисунок 1.15).

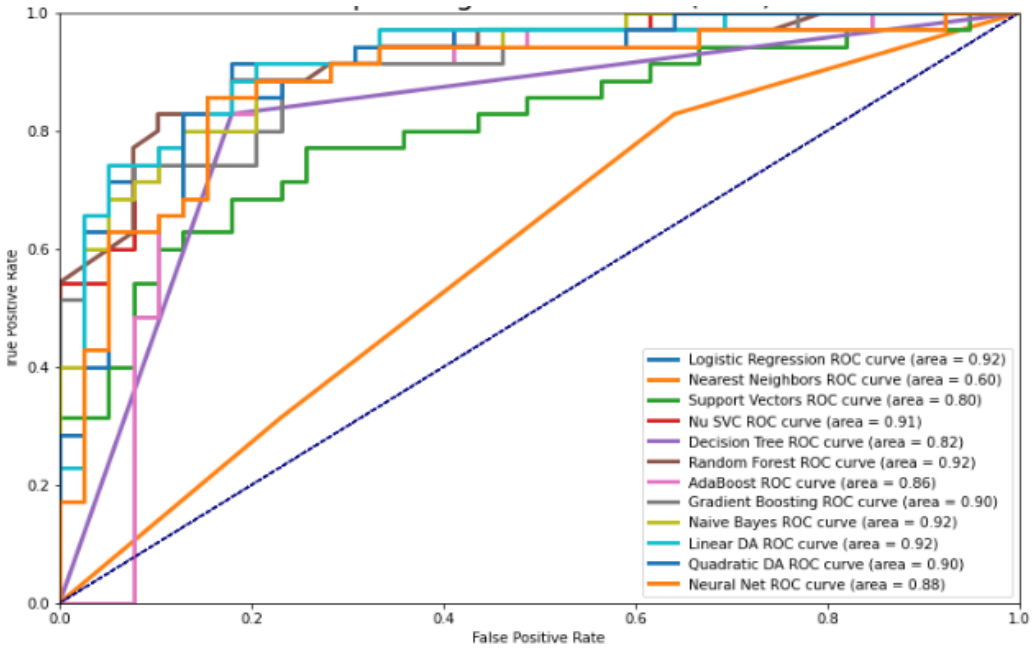
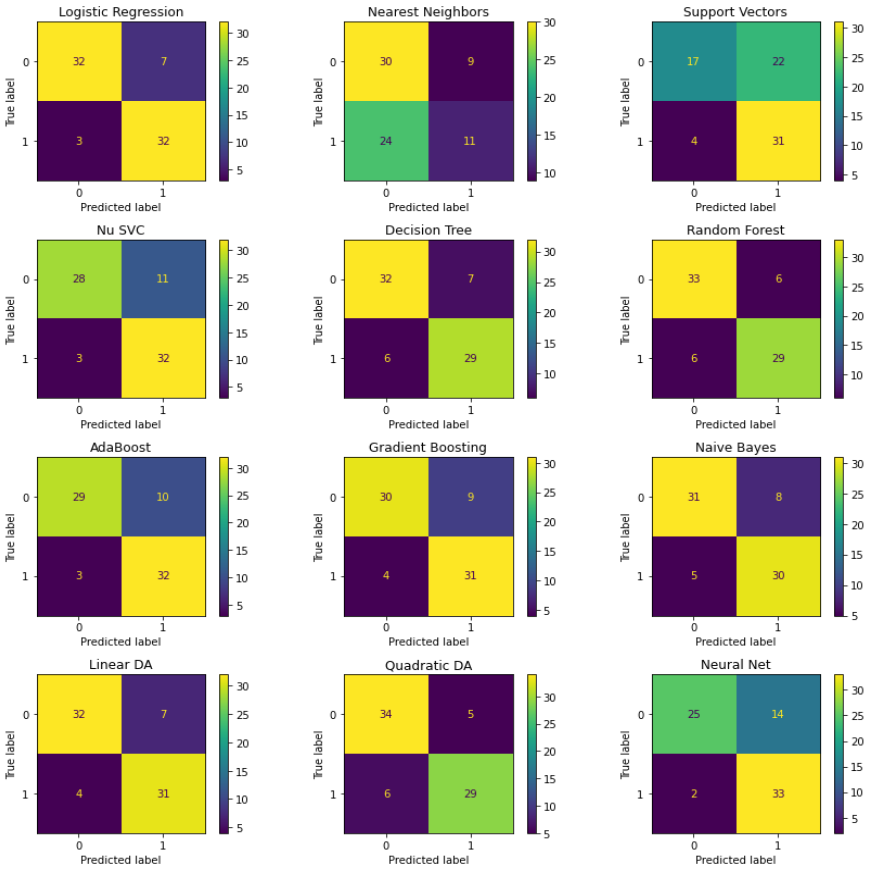


Рисунок 1.15 – Кривая рабочих характеристик

На рисунке 1.16 представлен результат построения матриц запутанностей для нескольких классификаторов.



1.16 – Матрицы запутанностей для различных классификаторов

Получилось, что у LR наилучший показатель F1, поэтому логистическая

регрессия в данном случае является лучшим классификатором. Теперь попробуем современные алгоритмы ML (boosted trees), такие как cat boost, xgboost и lgbm. Это оптимизированные алгоритмы машинного обучения, основанные на методе повышения градиента. В зависимости от конкретной задачи, один алгоритм может подойти лучше, чем другие. Для получения подробной информации можно обратиться к их документации.

```
from catboost import CatBoostClassifier
from xgboost import XGBClassifier
from lightgbm import LGBMClassifier
names_boost =[
    'Catboost',
    'xgbbost',
    'light GBM'
]
classifiers = [
    CatBoostClassifier(random_state=seed, verbose=0),
    XGBClassifier(objective= 'binary:logistic', random_state=seed),
    LGBMClassifier(random_state=seed)
   ]
```

Ниже представлен фрагмент кода сводной таблицы показателей производительности данных алгоритмов (рисунок 1.17).

```
score_summary(names_boost, classifiers).sort_values(by='Accuracy' , ascending = False)\
.style.background_gradient(cmap='coolwarm')\
.bar(subset=["ROC_AUC",], color='#6495ED')\
.bar(subset=["Recall"], color='#ff355d')\
.bar(subset=["Precision"], color='lightseagreen')\
.bar(subset=["F1"], color='gold')
```

	Classifier	Accuracy	ROC_AUC	Recall	Precision	F1
0	Catboost	82.430000	0.920000	0.830000	0.810000	0.820000
2	light GBM	82.430000	0.910000	0.860000	0.790000	0.820000
1	xgbbost	79.730000	0.920000	0.830000	0.760000	0.790000

Рисунок 1.17 – Сводная таблица показателей производительности

В данном случае (Lgbm) настройка гиперпараметра дала лучшие

результаты, чем в базовой модели. Значение отзыва было увеличено с 86% до 94%. Это означает, что снижено количество ложноотрицательных результатов с 5 до 2 в наборе проверок, а также снижено количество ложноположительных результатов на 1.

Одной из задач машинного обучения является объяснение прогнозирования модели. Модель может рассматривать некоторые функции как более важные, чем другие, для своего прогнозирования. Другая модель может оценивать другие функции как более важные. SHAP, сокращенное название SHapely Additive ExPlanations, - это метод, используемый для объяснения результатов модели машинного обучения. Он связывает оптимальное распределение баллов с локальными объяснениями, используя классические значения Шепли из теории игр и связанные с ними расширения. SHAP обладает широкими функциональными методами, с помощью которых можно интерпретировать выходные данные моделей. Важность перестановок и SHAP - это два метода, которые можно использовать, чтобы понять, какие функции были выбраны для того, чтобы оказать наибольшее влияние на прогноз модели.

```
import eli5
from eli5.sklearn import PermutationImportance

perm_imp = PermutationImportance(lgbm, random_state=seed).fit(X_train, y_train)
eli5.show_weights(perm_imp, feature_names = X_val.columns.tolist())
```

Weight	Feature
0.0901 ± 0.0127	num_major_vessels
0.0730 ± 0.0276	chest_pain_type
0.0288 ± 0.0340	st_slope
0.0099 ± 0.0175	max_heart_rate_achieved
0.0054 ± 0.0067	st_depression
0 ± 0.0000	thalassemia
0 ± 0.0000	exercise_induced_angina
0 ± 0.0000	resting_electrocardiogram
0 ± 0.0000	fasting_blood_sugar
0 ± 0.0000	resting_blood_pressure
0 ± 0.0000	sex
-0.0009 ± 0.0144	cholesterol
-0.0045 ± 0.0114	age

Рисунок 1.18 – Результат работы метода

Ниже используем shape.summary_plot(), чтобы определить влияние каждой функции на прогнозируемый результат (рисуноки 1.19 и 1.20).

```
import shap
shap.initjs()
explainer = shap.TreeExplainer(lgbm)
shap_values = explainer.shap_values(X_val)
shap.summary_plot(shap_values, X_val,
                  feature_names=features,
                  plot_type="bar",
                 )
```

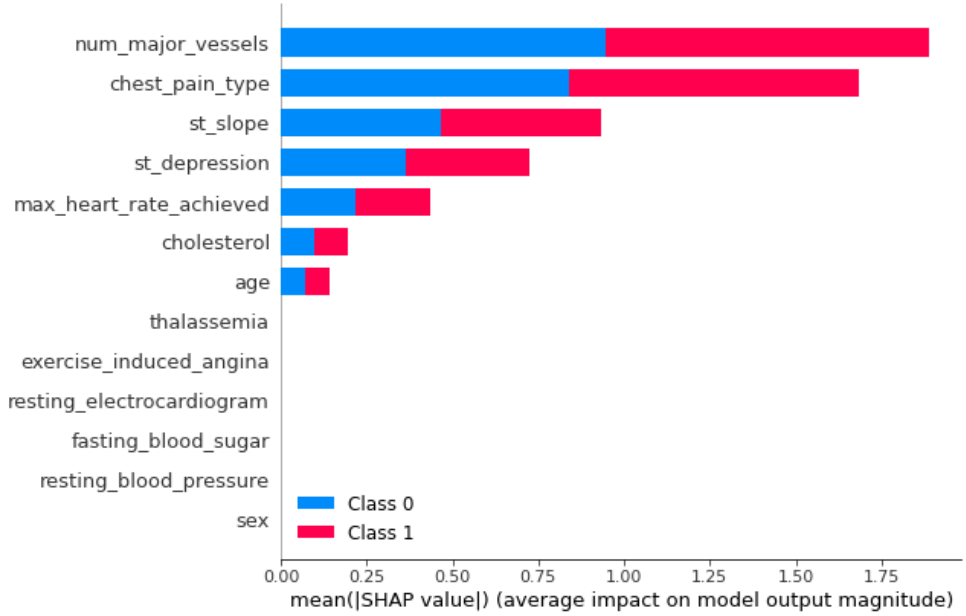


Рисунок 1.19 – Результат работы метода

```
shap.summary_plot(shap_values[0], X_val)
```

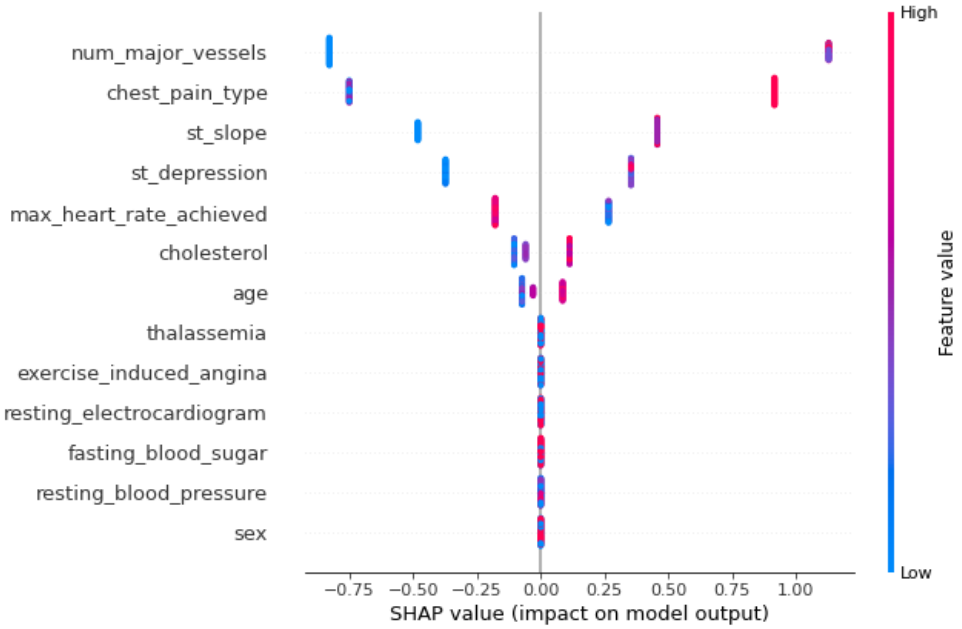


Рисунок 1.20 – Результат работы метода

Контрольные вопросы

1. Какие переменные называются целевыми?
2. Какие методы были использованы для нормализации данных?
3. Как можно получить статистическую сводку по набору данных?
4. В чем отличия точечно-бисериальной корреляции и коэффициента корреляции Пирсона?

ЛАБОРАТОРНАЯ РАБОТА № 4

УСТАНОВКА HADOOP SINGLE NODE

Целью работы является ознакомление с теоретическими и практическими основами работы с платформой Apache Hadoop, её настройка, а также применение в решении базовых задач.

Краткие теоретические сведения

С момента появления первых компьютеров люди работают с данными. Со временем объём данных начал расти, и требования к скорости обработки тоже. В 2005 году появилась новая концепция — MapReduce. Дуг Каттинг и Майк Кафарелла, разработчики программного обеспечения, решили создать на базе этой концепции программную инфраструктуру для распределённых вычислений. Уже спустя год проектом заинтересовалась компания Yahoo, в 2008 на базе технологии Hadoop они запустили поисковую машину. Так Hadoop стал верхнеуровневым проектом системы Apache Software Foundation.

Hadoop – мощный инструментарий для работы с большими данными от Apache foundation. Он работает по принципу MapReduce, то есть распределения данных. Его суть в том, что система состоит из кластеров — групп отдельных подсерверов, или узлов, которые используются как единый ресурс. Когда на кластер поступает обширная задача, Hadoop делит её на много мелких подзадач и выполняет каждую на своём узле. Это позволяет параллельно решать несколько задач и быстрее выдать конечный результат.

У Hadoop есть несколько режимов запуска: локальный, псевдо-распределённый и полностью распределённый.

Изначально Hadoop был, в первую очередь, инструментом для хранения данных, сейчас же платформа представляет собой большой стек технологий, так или иначе связанных с обработкой больших данных (не только при помощи MapReduce).

Основными компонентами Hadoop являются:

– Hadoop Distributed File System (HDFS) – распределённая файловая система, позволяющая хранить информацию практически неограниченного объёма.

– Hadoop YARN – фреймворк для управления ресурсами кластера и менеджмента задач, в том числе включает фреймворк MapReduce.

– Hadoop common – набор инструментов, которые используются для создания инфраструктуры и работы с файлами. По сути, это управляющая система для остальных модулей и связи с дополнительными инструментами.

Hadoop нужен, чтобы:

– Повысить скорость обработки данных благодаря модели MapReduce и параллельным вычислениям.

– Обеспечить устойчивость данных за счёт хранения резервных копий на других узлах.

– Работать с данными любых типов и видов, в том числе неструктурированными, например видео.

Требования к выполнению лабораторной работы

1. Ознакомиться с методическими указаниями.

2. Выполнить ход работы.

3. Составить отчёт с использованием индивидуальных параметров (к примеру, фон рабочего стола).

1 Ход работы

1.1 Установка CentOS

Перед началом работы необходимо установить и настроить виртуальную машину. В качестве примера будет использован VMware Workstation 17 Player (рисунок 1.1).

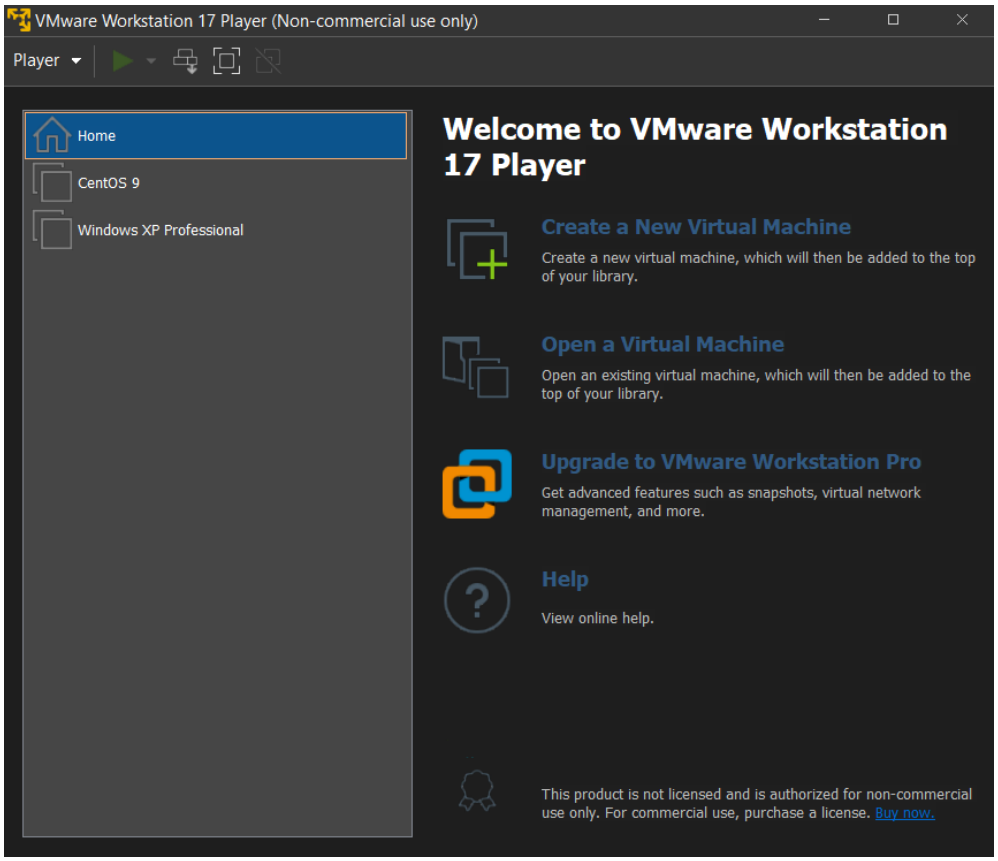


Рисунок 1.1 – Главное окно VMware Workstation 17 Player

Весь ход работы будет осуществлён на примере дистрибутива Linux – CentOS. С официального сайта необходимо скачать образ CentOS Stream 9 x86_64 (https://www.centos.org/).

С помощью кнопки «Create a New Virtual Machine» можно добавлять новые операционные системы. В поле файла образа установочного диска нужно указать путь к ранее загруженному образу дистрибутива (рисунок 1.2).

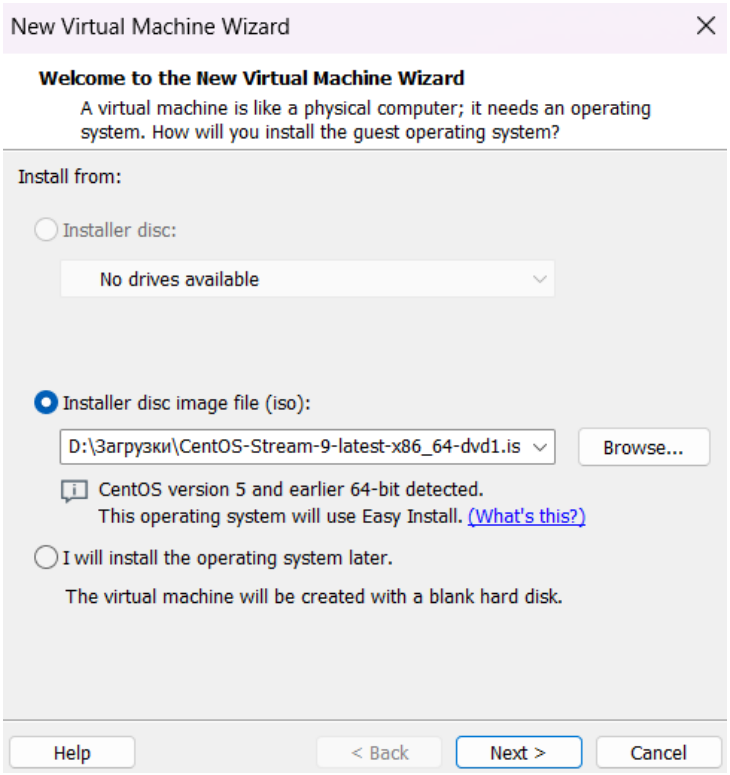


Рисунок 1.2 – Добавление CentOS

Далее заполняются поля персонализации. В строке ввода имени пользователя нельзя использовать заглавные буквы (рисунок 1.3).

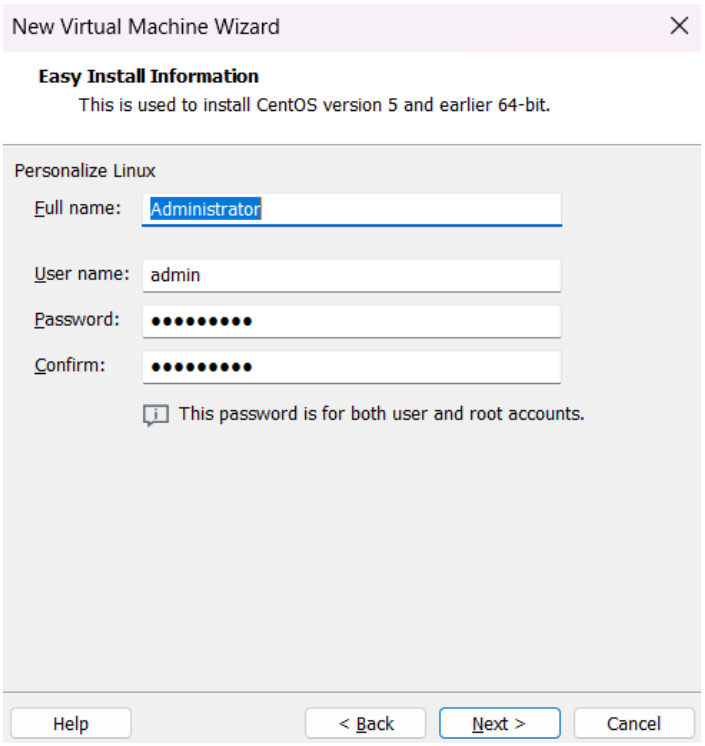


Рисунок 1.3 – Заполнение данных пользователя

Следующее окно нужно оставить без изменения и перейти далее (рисунок 1.4).

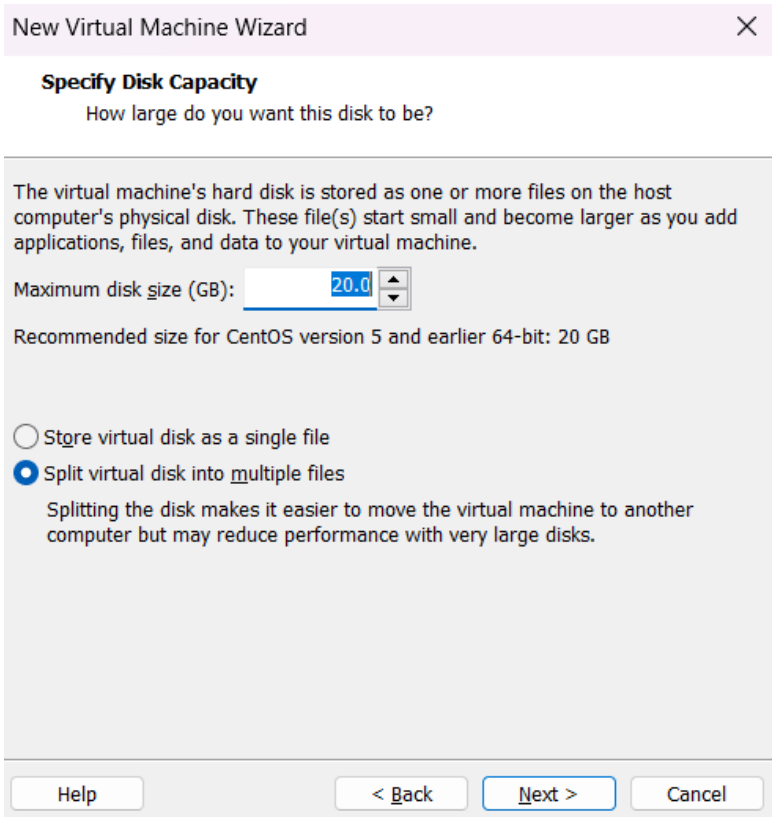


Рисунок 1.4 – Выбор максимального размера диска

В появившемся окне указаны все проведённые пользователем настройки операционной системы. Для эффективной работы CentOS необходимо поменять последний параметр с помощью кнопки «Customize Hardware…» (рисунок 1.5).

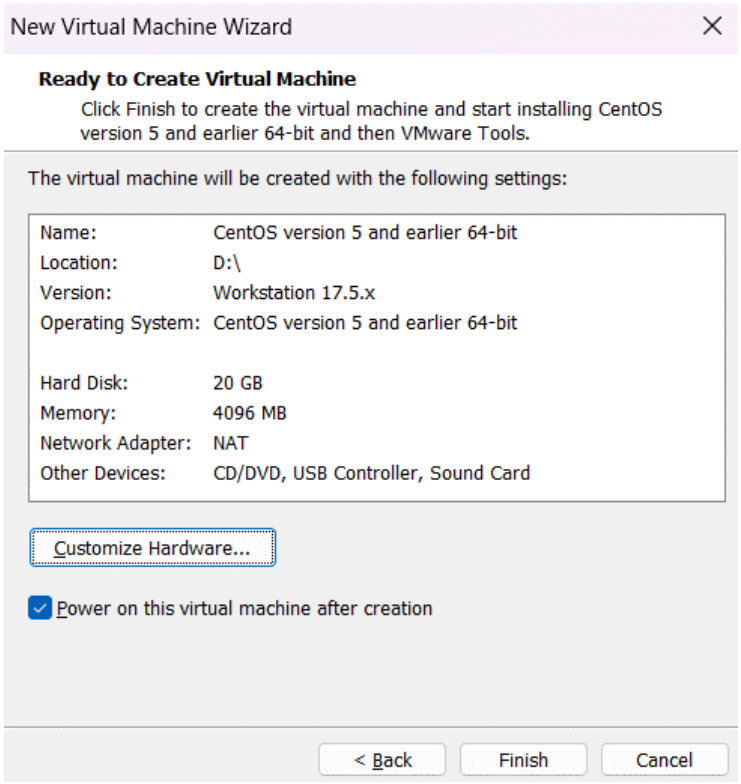


Рисунок 1.5 – Заключительная настройка операционной системы

В данном пункте для корректной работы дистрибутива следует выделить не менее 6 Гб (рисунок 1.6).

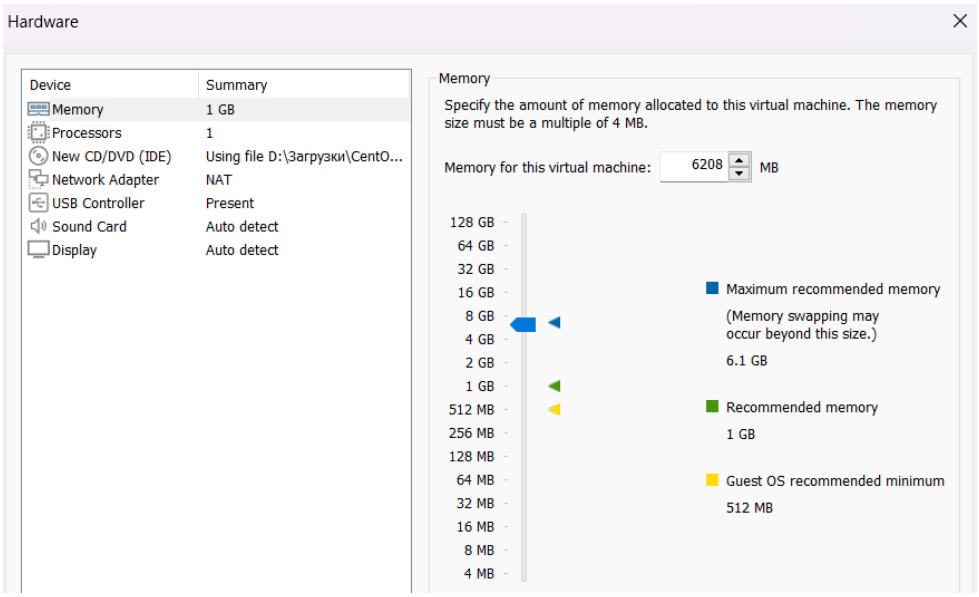


Рисунок 1.6 – Выделение нужного объёма памяти

На этом шаге настройка операционной системы завершена. После нажатия соответствующей кнопки начинается запуск самого установщика CentOS (рисунок 1.7).

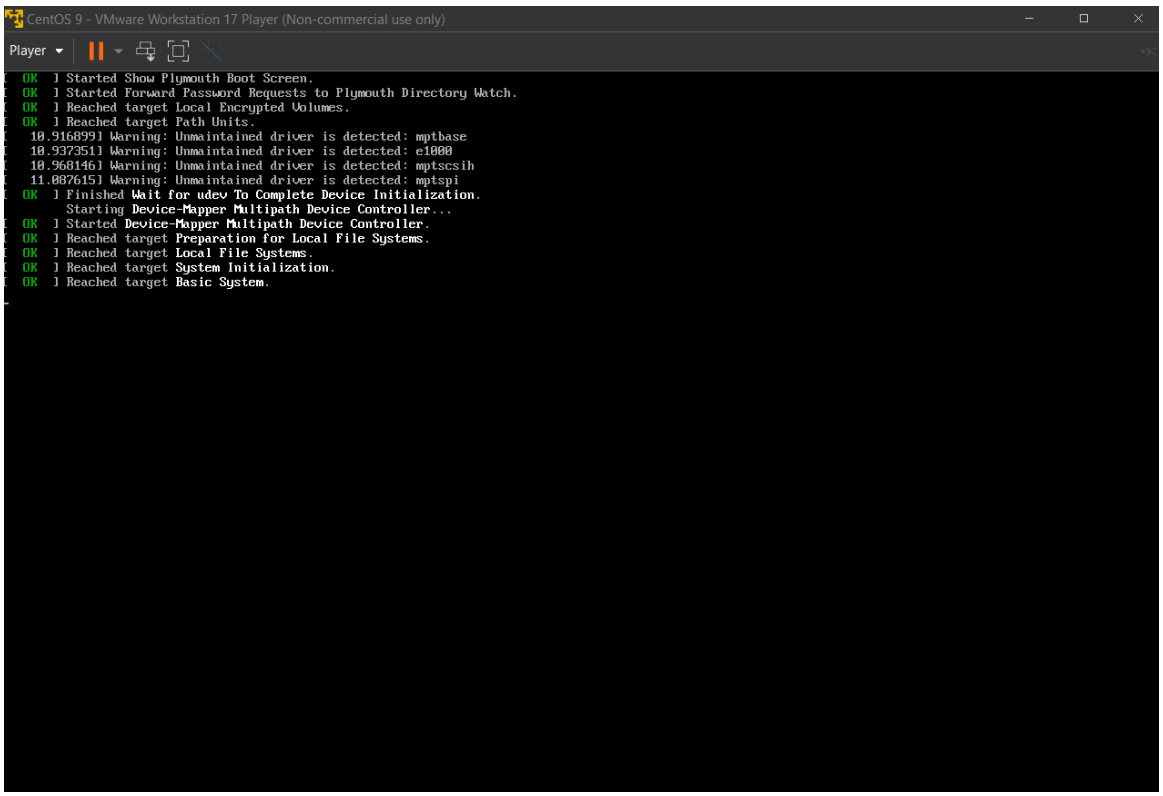


Рисунок 1.7 – Запуск компонентов CentOS

После успешного запуска всех компонентов открывается графический интерфейс установки CentOS Stream 9. На рисунке 1.8 происходит выбор языка, который будет использоваться в процессе установки.

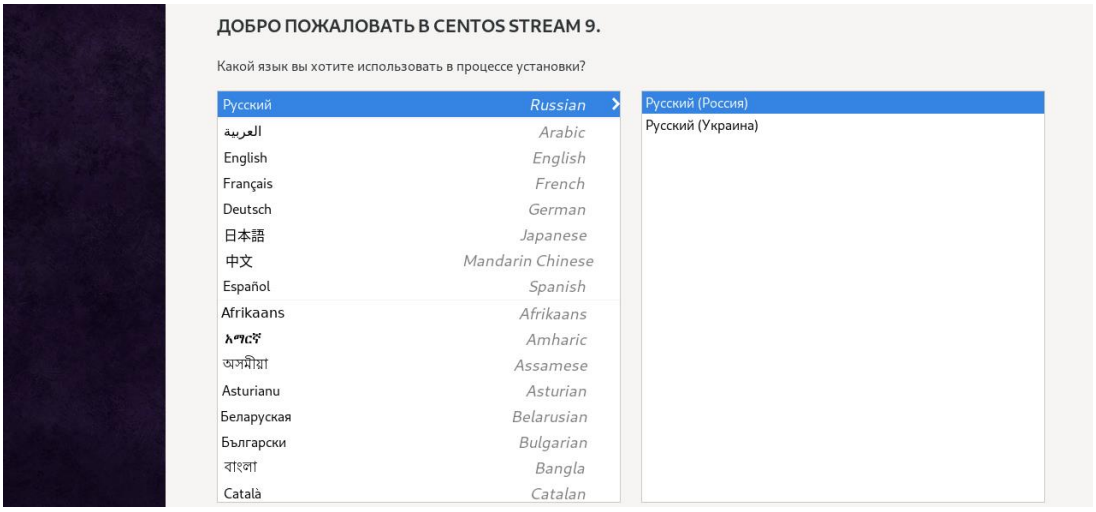


Рисунок 1.8 – Выбор языка

Последующие шаги установки происходят в окне обзора параметров установки. Некоторые пункты требуют особого внимания и выделены ярким цветом (рисунок 1.9).

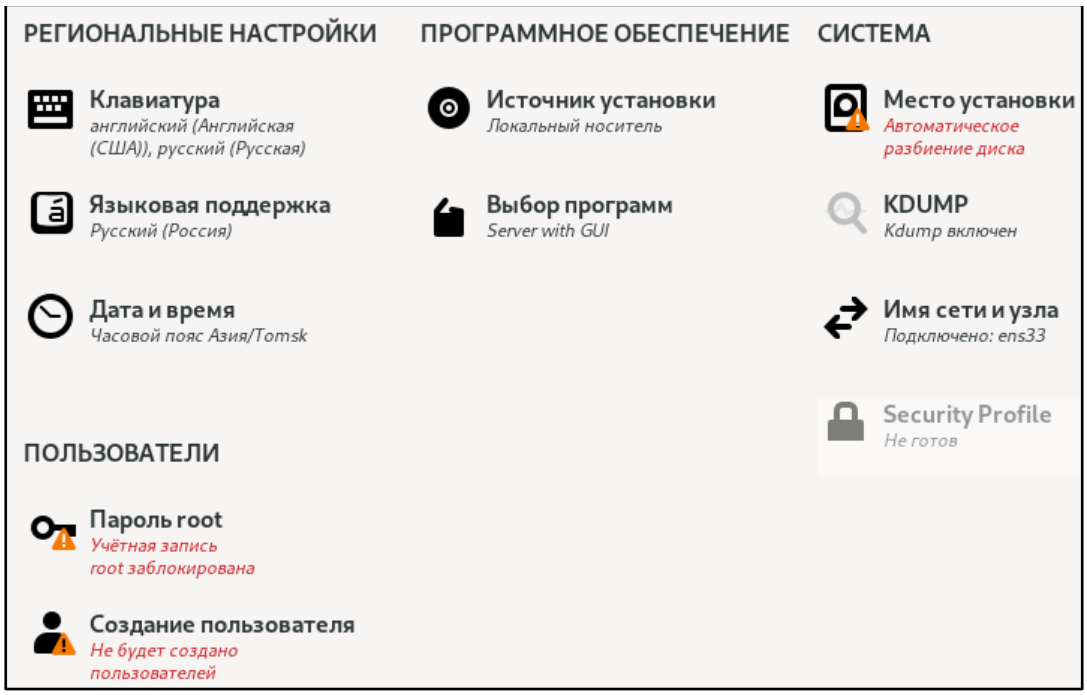


Рисунок 1.9 – Изначальный вид окна установки

Первое, что стоит сделать, – проверить подключение сети Ethernet (рисунок 1.10).

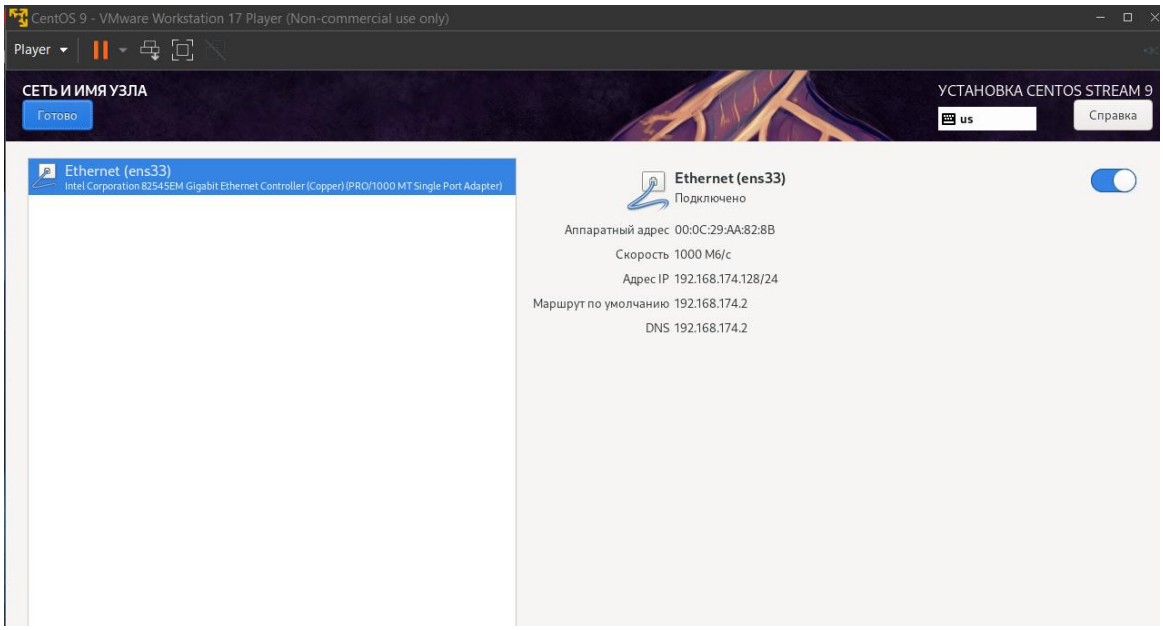


Рисунок 1.10 – Настройка «Сеть и имя узла»

Следующая настройка – «Место установки». Необходимо проверить, что в поле «Локальные диски» выбран виртуальный диск, и нажать кнопку «Готово» (рисунок 1.11).

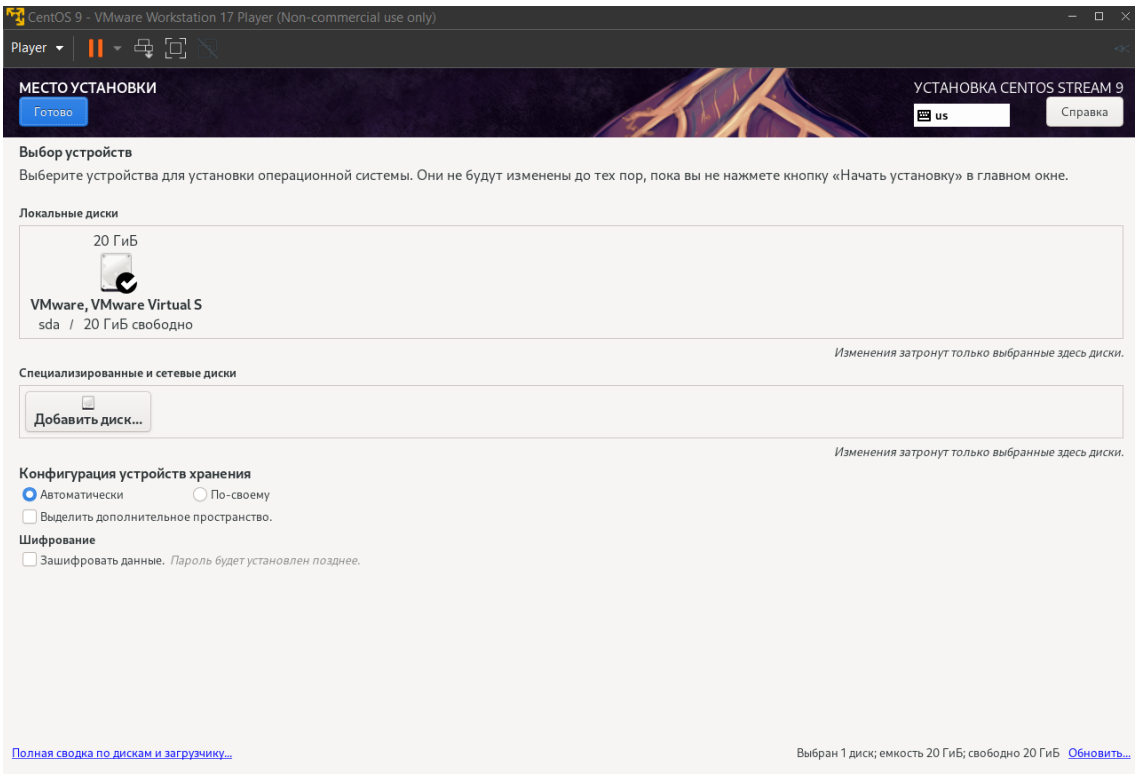


Рисунок 1.11 – Настройка «Место установки»

В окне «Выбор программ» происходит определение вида интерфейса, в котором будет работать пользователь. В базовом окружении необходимо выбрать «Server», в качестве дополнительного ПО – «GNOME» (рисунок 1.12).

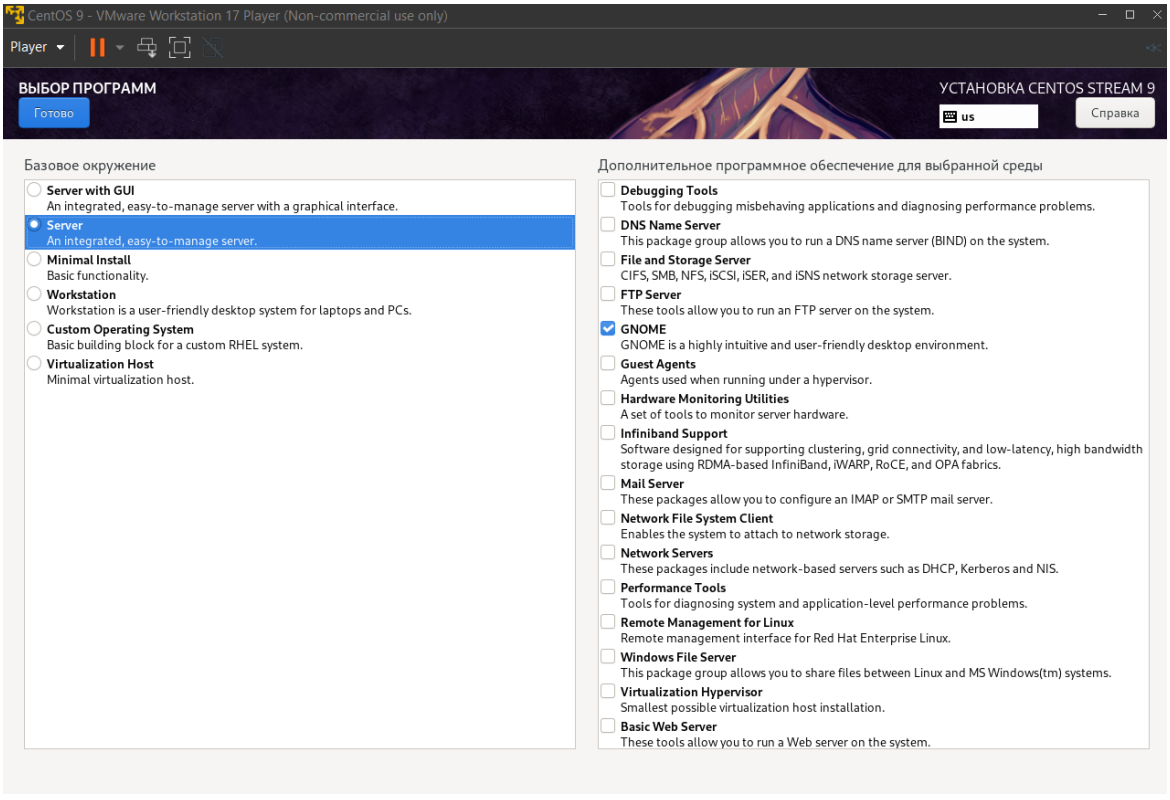


Рисунок 1.12 – Настройка «Выбор программ»

Последними двумя действиями осуществляется добавление двух учётных записей: администратора (root) и пользователя (рисунок 1.13).

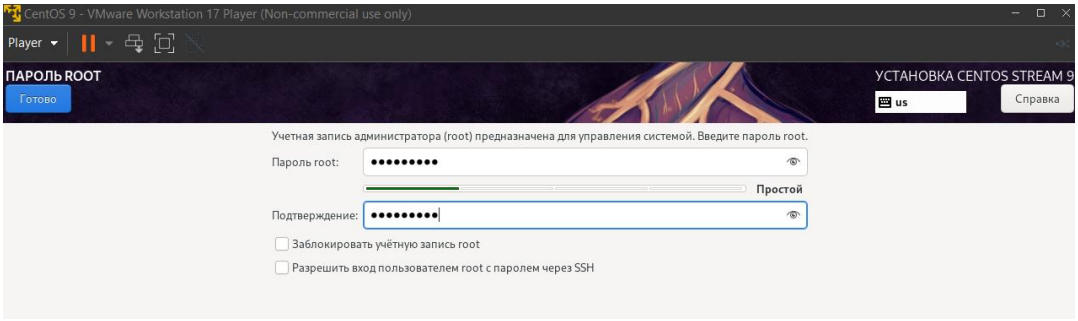


Рисунок 1.13 – Создание учётной записи root

После всех вышеописанных действий можно завершать настройку. Окно обзора установки должно выглядеть, как представлено на рисунке 1.14. После можно переходить к установке.

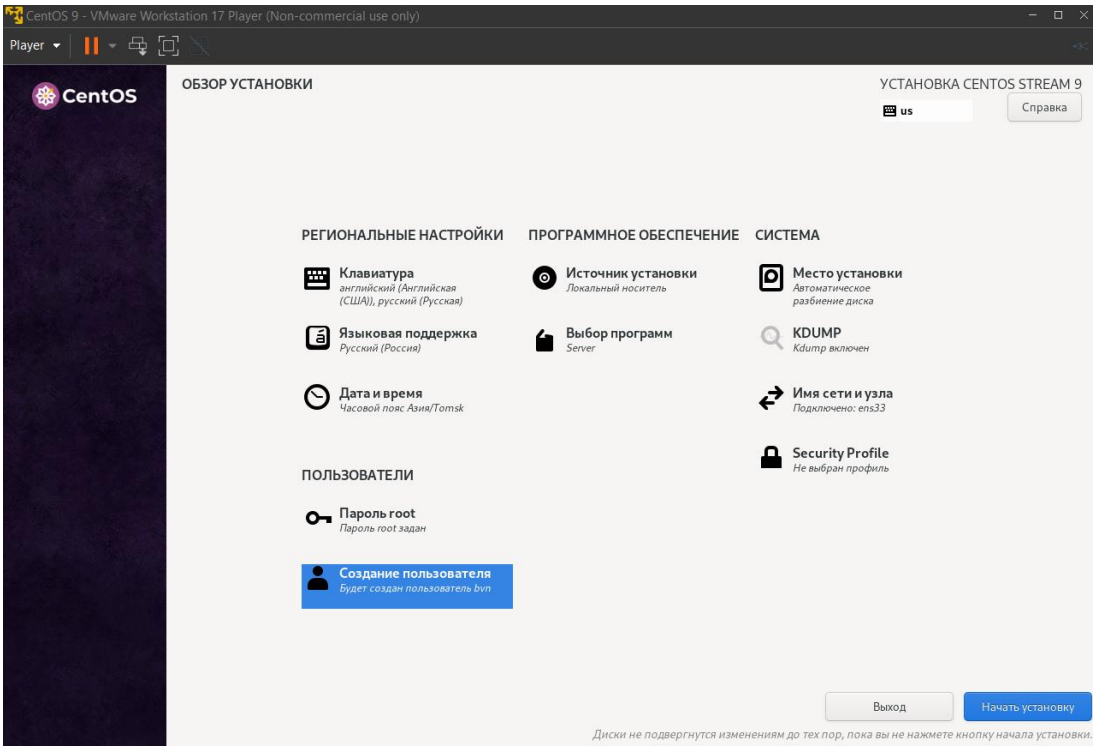


Рисунок 1.14 – Завершение настройки

Установка происходит без вмешательства человека. В конце машина перезапускается. По умолчанию root не отображается в списке пользователей, поэтому нужно выбрать «Нет в списке», ввести данную учётную запись и пароль от неё (рисунок 1.15).

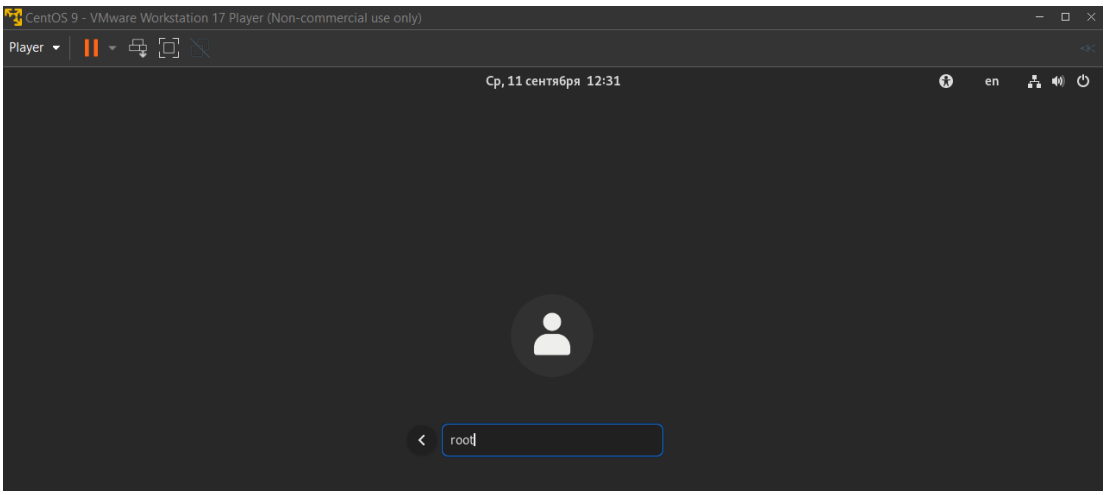


Рисунок 1.15 – Вход под учётной записью root

После входа в систему запускается рабочий стол. На этом первый этап лабораторной работы завершается.

1.2 Установка Hadoop

Перед началом работы с Hadoop, необходимо войти в «Центр приложений» на рабочем столе виртуальной машины и установить веб-браузер. В рамках данной лабораторной установлен Firefox (рисунки 1.16).

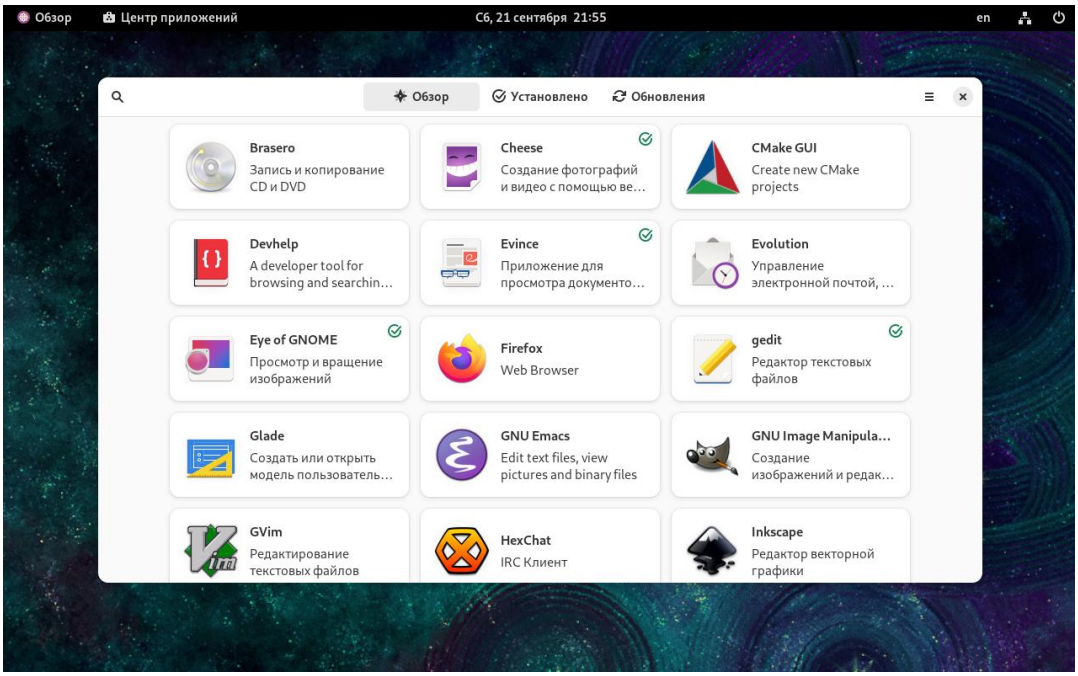


Рисунок 1.16 – Центр приложений

В рамках курса понадобится среда разработки Intellij IDEA, её можно найти в любом веб-браузере, перейти на официальный сайт (https://www.jetbrains.com/ru-ru/idea/) и загрузить community редакцию (рисунок 1.17).

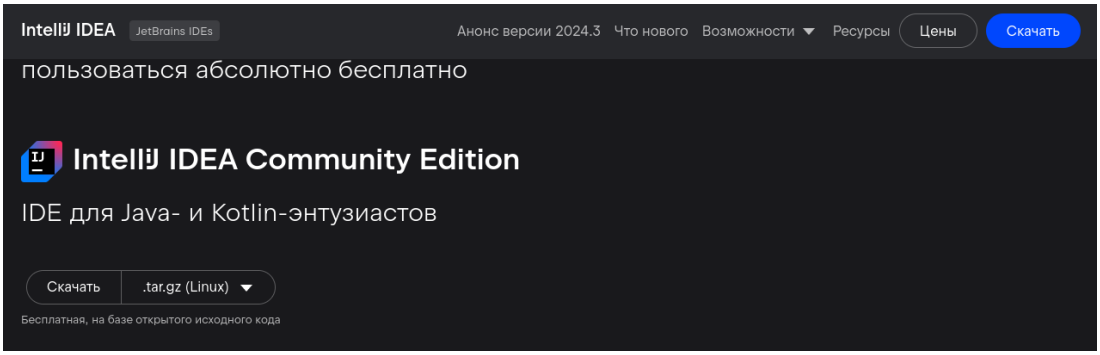


Рисунок 1.17 – Окно загрузки Community Edition

После того, как архив загрузится, его нужно разархивировать (рисунок 1.18). Для этого открывается утилита командной строки и используется команда tar с опциями для разархивирования.

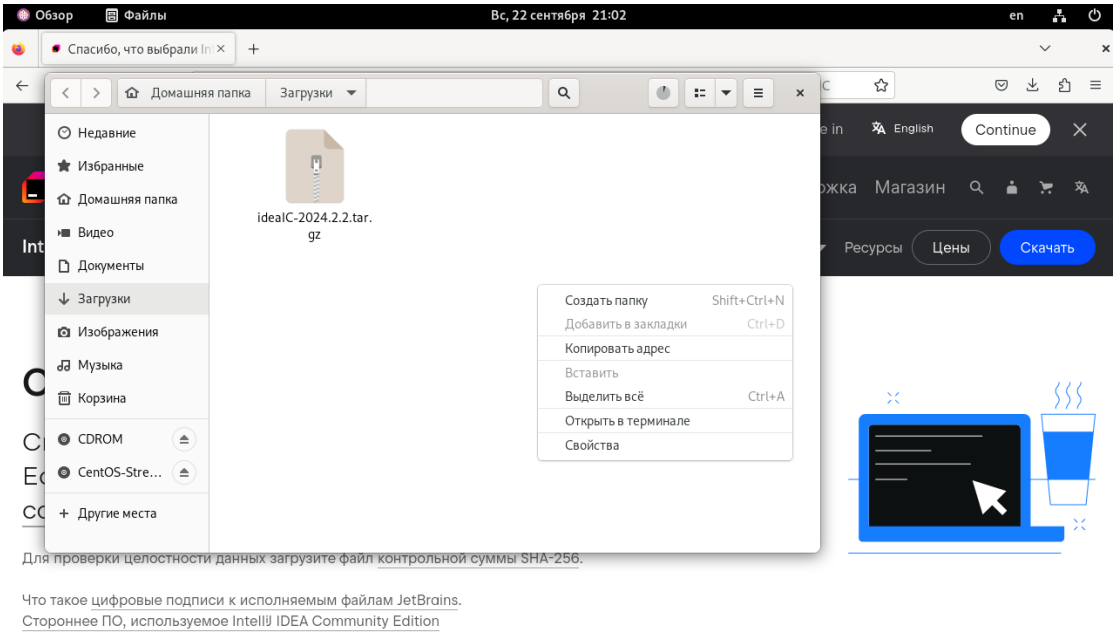


Рисунок 1.18 – Открытие терминала в папке «Загрузки»

После утилиты tar используется набор опций: -x – извлечение файлов из архива (разархивация); -z – распаковка архива, сжатого с помощью gzip; -f –

указание имени архива для обработки. В качестве целевой директории укажите каталог /opt (рисунок 1.19).

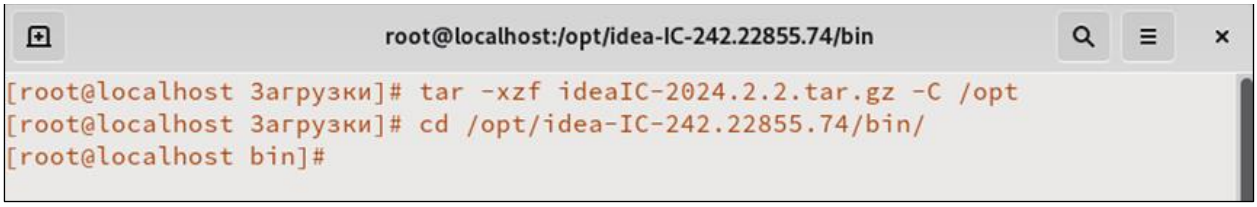


Рисунок 1.19 – Разархивация и запуск IDEA

После чего перейдите в папку с IDEA и запустите файл idea.sh (рисунок 1.20).

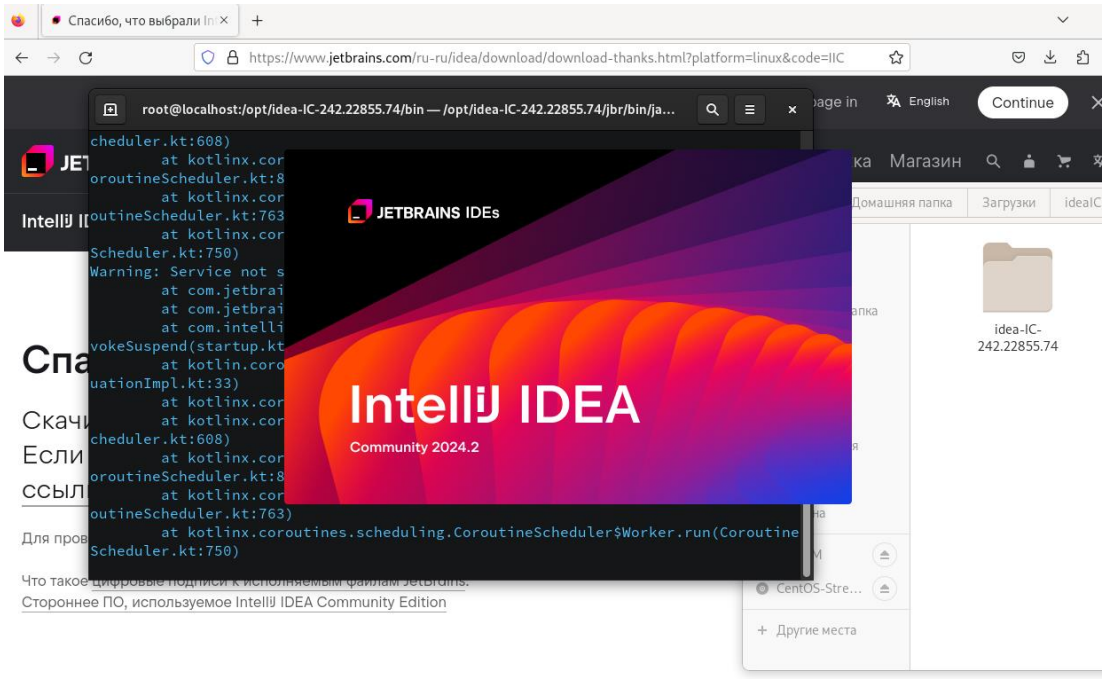


Рисунок 1.20 – Запуск приложения

При первом запуске IDEA не отображается в списке приложений операционной системы. Для того, чтобы она отображалась, можно создать пустой проект (рисунок 1.21).

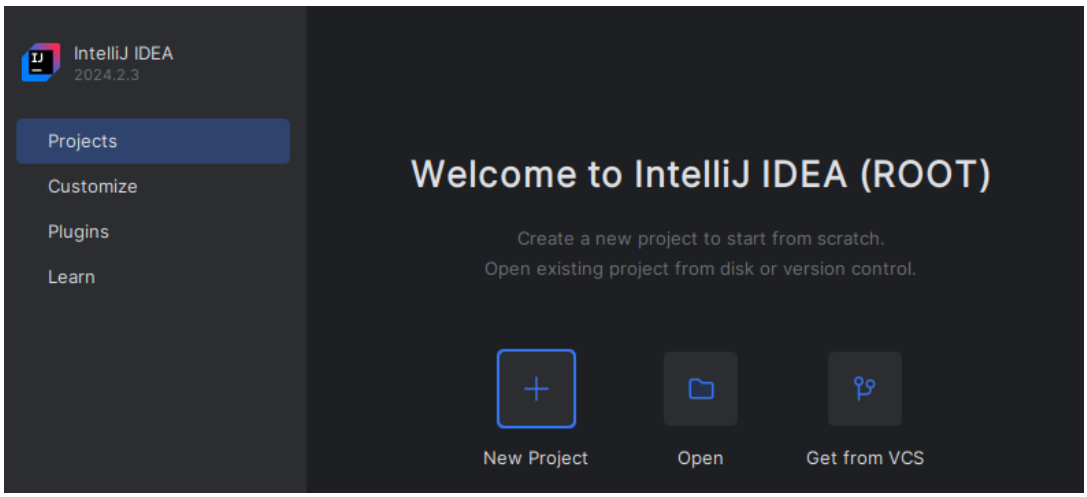


Рисунок 1.21 – Создание нового проекта

Открывается окно IDEA, и в меню «Tools» есть специальная функция «Create Desktop Entry», которая добавляет IDEA в список приложений для всех пользователей (рисунок 1.22).

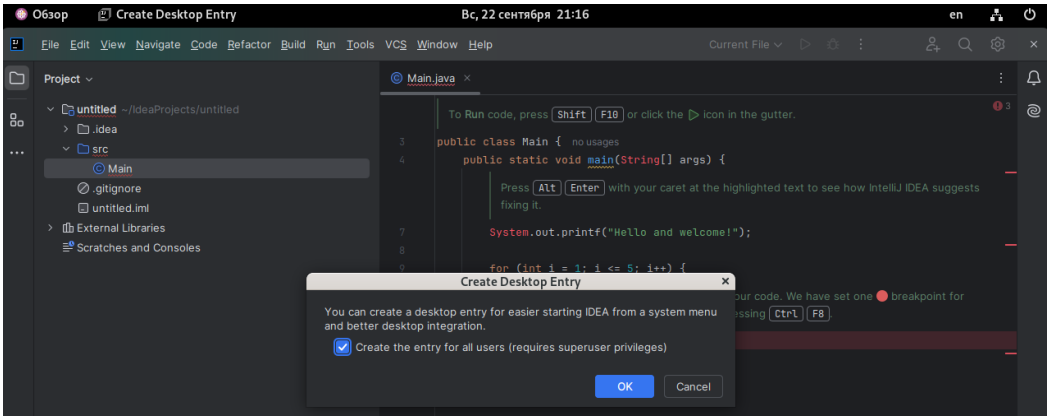


Рисунок 1.22 – Добавление IDEA в список приложений

После этого проект можно закрыть и выполнить проверку, открыв список приложений (рисунок 1.23).

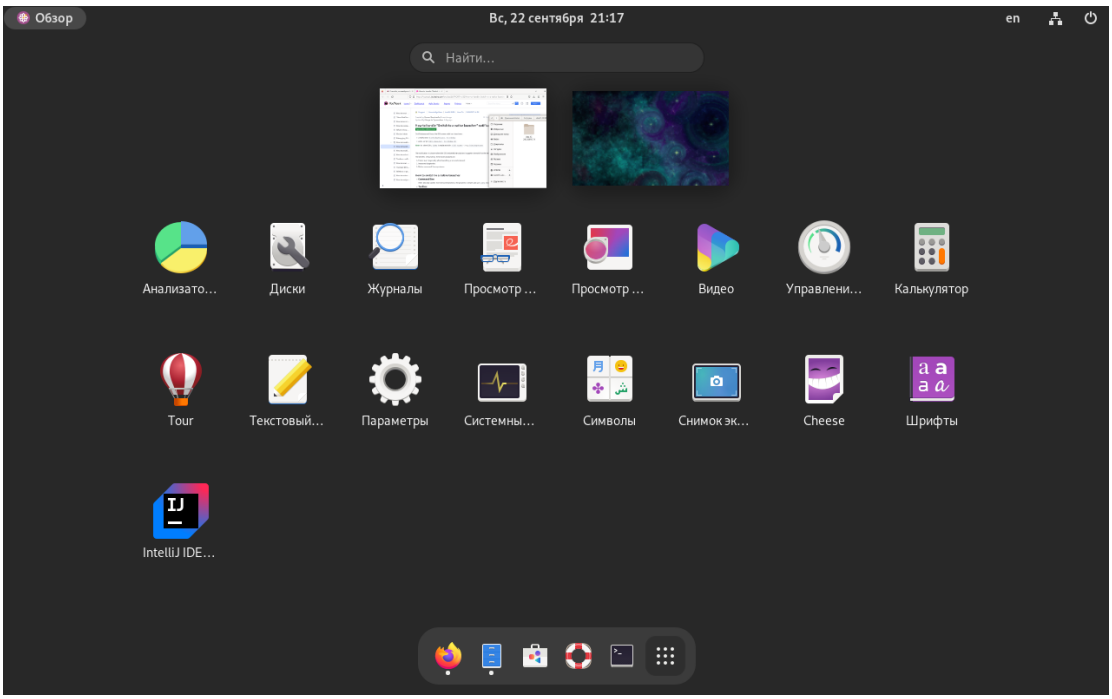


Рисунок 1.23 – Проверка отображения IDEA в списке

При запуске приложения запускается новый проект. При этом созданный ранее проект нужно удалить. Для этого открытый проект закрывается с помощью опции «Close Project» в списке меню (рисунок 1.24).

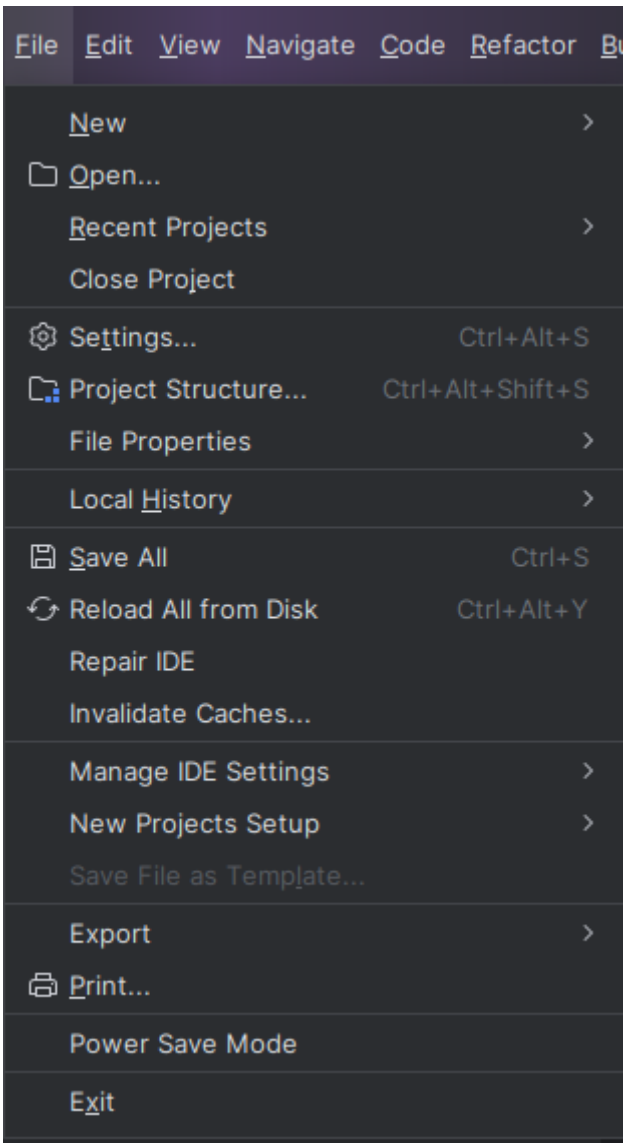


Рисунок 1.24 – Закрытие проекта

А далее в главном меню удалить его из недавних проектов используя пункт «Remove from Recent Projects» (рисунок 1.25).

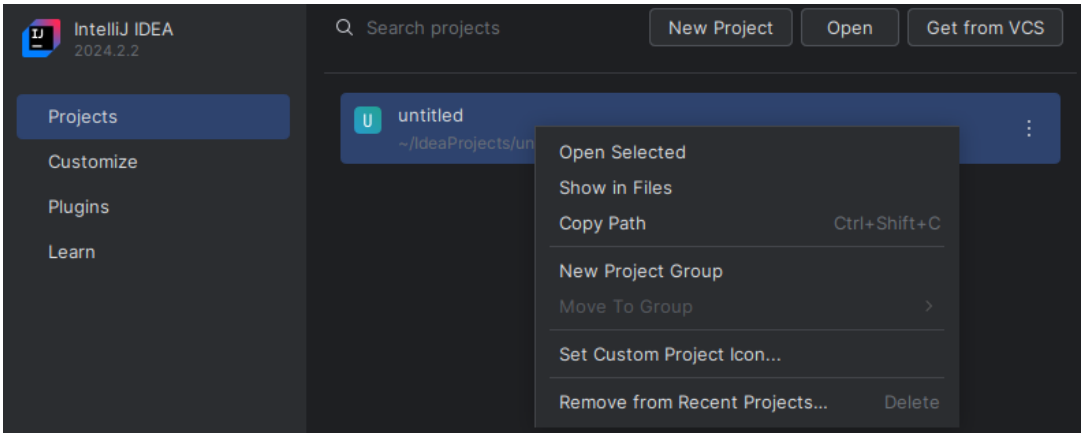


Рисунок 1.25 – Удаление проекта

Теперь можно перейти к установке и настройке непосредственно фреймворка Hadoop. Для этого понадобятся бинарные файлы Hadoop с официального сайта: https://hadoop.apache.org/ (рисунок 1.26).

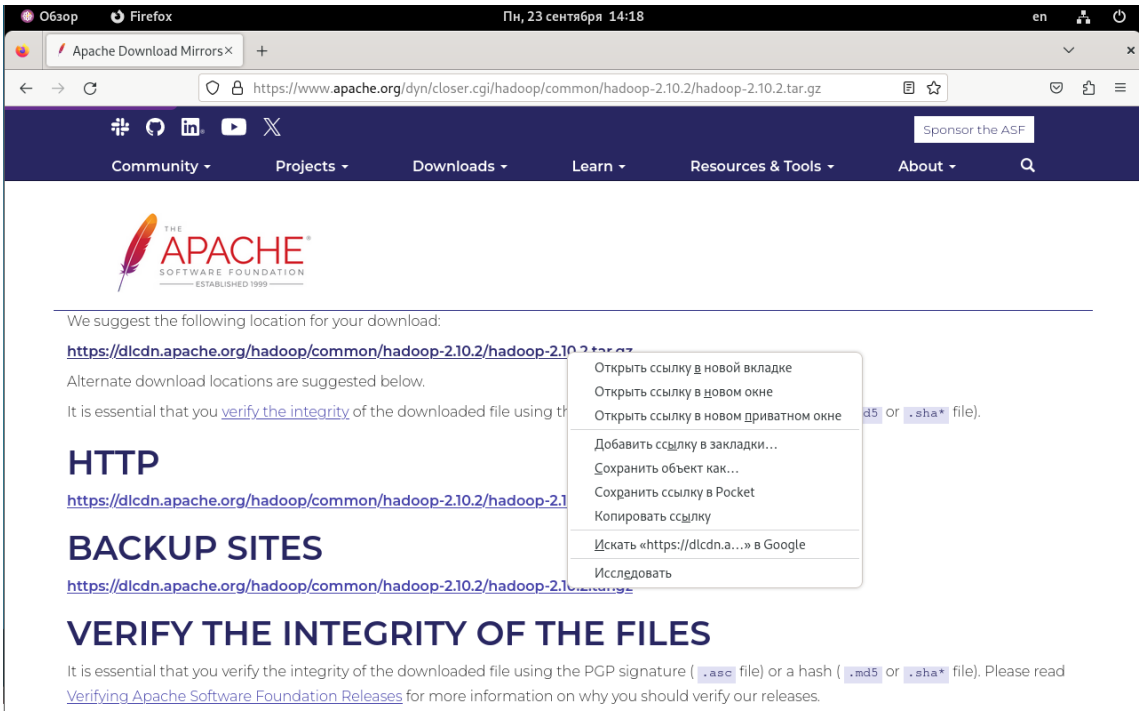


Рисунок 1.26 – Установка Hadoop версии 2.10.2

Через утилиту командной строки «wget» можно с использованием ссылки на зеркало с дистрибутивом Hadoop выполнить загрузку. Ссылка на зеркало: https://dlcdn.apache.org/hadoop/common/hadoop-2.10.2/hadoop-2.10.2.tar.gz (рисунок 1.27).

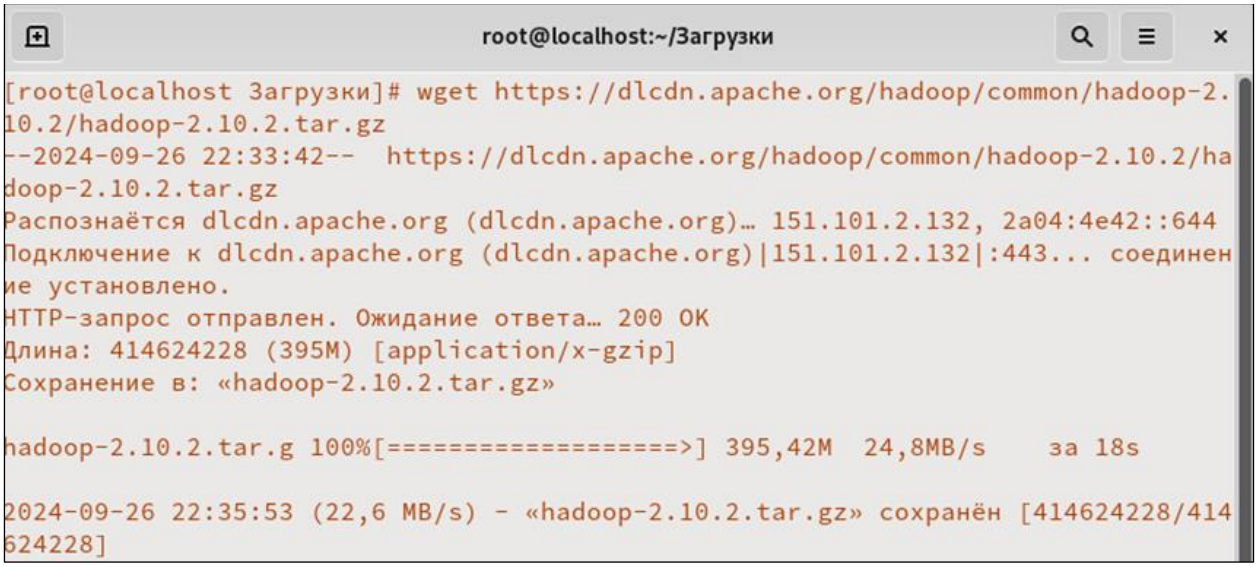


Рисунок 1.27 – Загрузка Hadoop с помощью утилиты wget

После загрузки нужно разархивировать файл в директорию /opt/. В терминале выполняется команда: tar -zxvf hadoop-2.10.2.tar.gz -C /opt/. Опция -v в команде tar включает подробный режим, при котором tar выводит в терминал список всех файлов, которые извлекаются или добавляются в архив.

Файлы разархивированы, следующим шагом нужно провести минимальную настройку Hadoop. В настоящий момент единственное, что нужно установить, чтобы фреймворк работал, – это JAVA_HOME (рисунок 1.28). Для установки в данной лабораторной работе используется редактор vi: vi /opt/hadoop-2.10.2/etc/hadoop/hadoop-env.sh. В результате выполнения команды открывается файл «hadoop-env.sh».

Этот файл является конфигурационным скриптом для Apache Hadoop и используется для настройки переменных среды, которые Hadoop будет использовать при запуске. В частности, в этом файле часто настраивают переменные, такие как JAVA_HOME (указывает путь к Java), и другие параметры, связанные с производительностью и конфигурацией среды Hadoop.

```
# regarding copyright ownership.  The ASF licenses this file
# to you under the Apache License, Version 2.0 (the
# "License"); you may not use this file except in compliance
# with the License.  You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.

# Set Hadoop-specific environment variables here.

# The only required environment variable is JAVA_HOME.  All others are
# optional.  When running a distributed configuration it is best to
# set JAVA_HOME in this file, so that it is correctly defined on
# remote nodes.

# The java implementation to use.
export JAVA_HOME=${JAVA_HOME}
```

Рисунок 1.28 – Файл «hadoop-env.sh» до изменения

Далее необходимо прописать в открывшемся файле значение этой переменной. Используются специальные команды для установки (рисунок 1.29). Стоит убедиться, что изменения сохранены и выйти с помощью :wq.

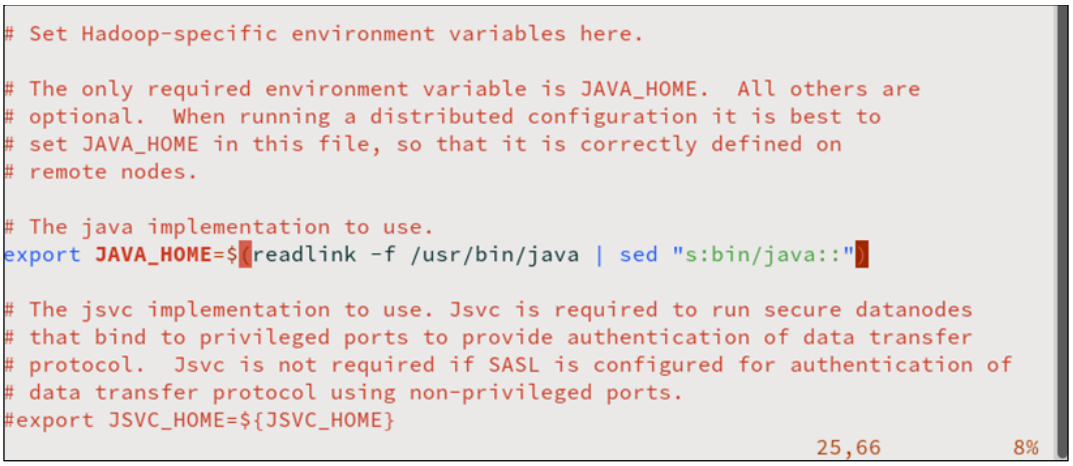


Рисунок 1.29 – Редактирование значения переменной

Чтобы не писать каждый раз путь к исполняемым файлам Hadoop вручную, тем самым сокращая время выполнения дальнейших действий лабораторной работы, следует добавить директорию с Hadoop в PATH (рисунок 1.30).

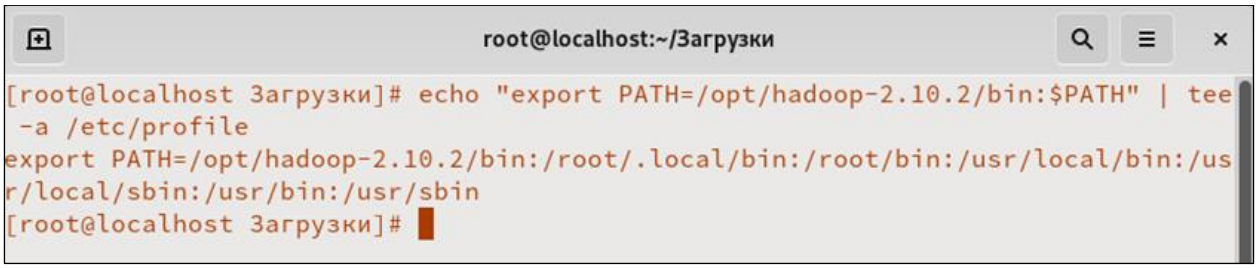


Рисунок 1.30 – Добавление Hadoop в PATH

Для того, чтобы в текущем сеансе применились настройки для переменной PATH, нужно этот файл «profile» проверить (cat /etc/profile) и применить к нему команду «source». В самом конце файла должна отображаться строка экспорта (рисунок 1.31). В результате будет получена возможность работать с командами Hadoop из любой директории.

Теперь для того, чтобы проверить корректность настройки, необходимо создать тестовую директорию и скопировать туда файл с лицензией Hadoop. Далее запускается фреймворк в локальном режиме. Все команды представлены на рисунке 1.31.

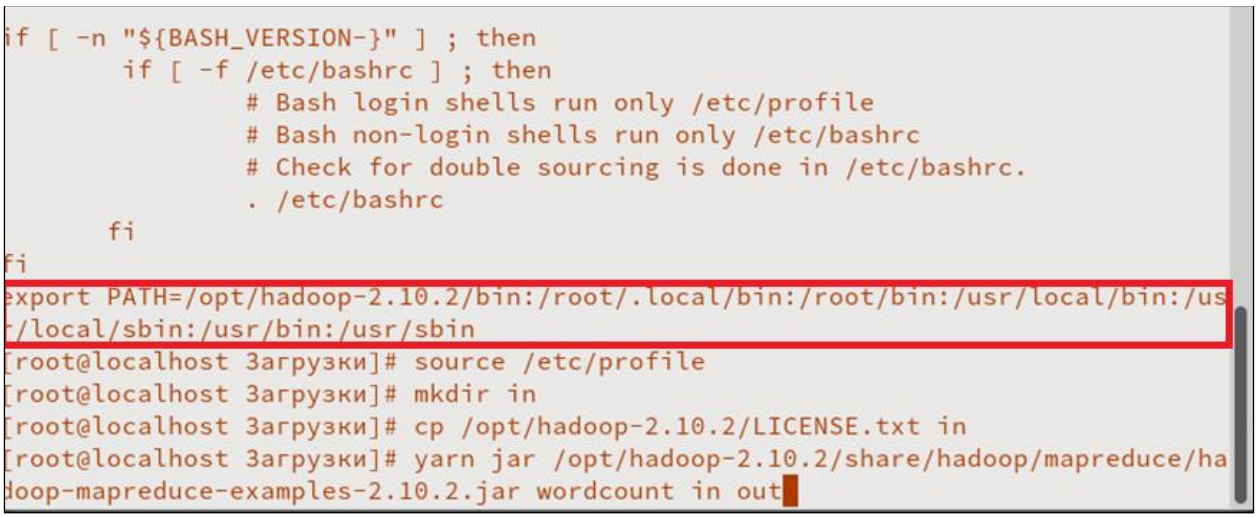


Рисунок 1.31 – Проверка корректности настройки

После того, как Hadoop запустился, произошёл расчёт и появилась директория «out». В ней находятся два файла: Part-r-00000 содержит в себе результат работы программы; _SUCCESS является индикатором успешного выполнения задания (рисунок 1.32).

Рисунок 1.32 – Запуск Hadoop и проверка новой директории

Далее для того, чтобы можно было разрабатывать Hadoop-программы, понадобится Java Development Kit. В данной работе используется Java версии 8, для установки используйте команду «yum install» (рисунок 1.33).

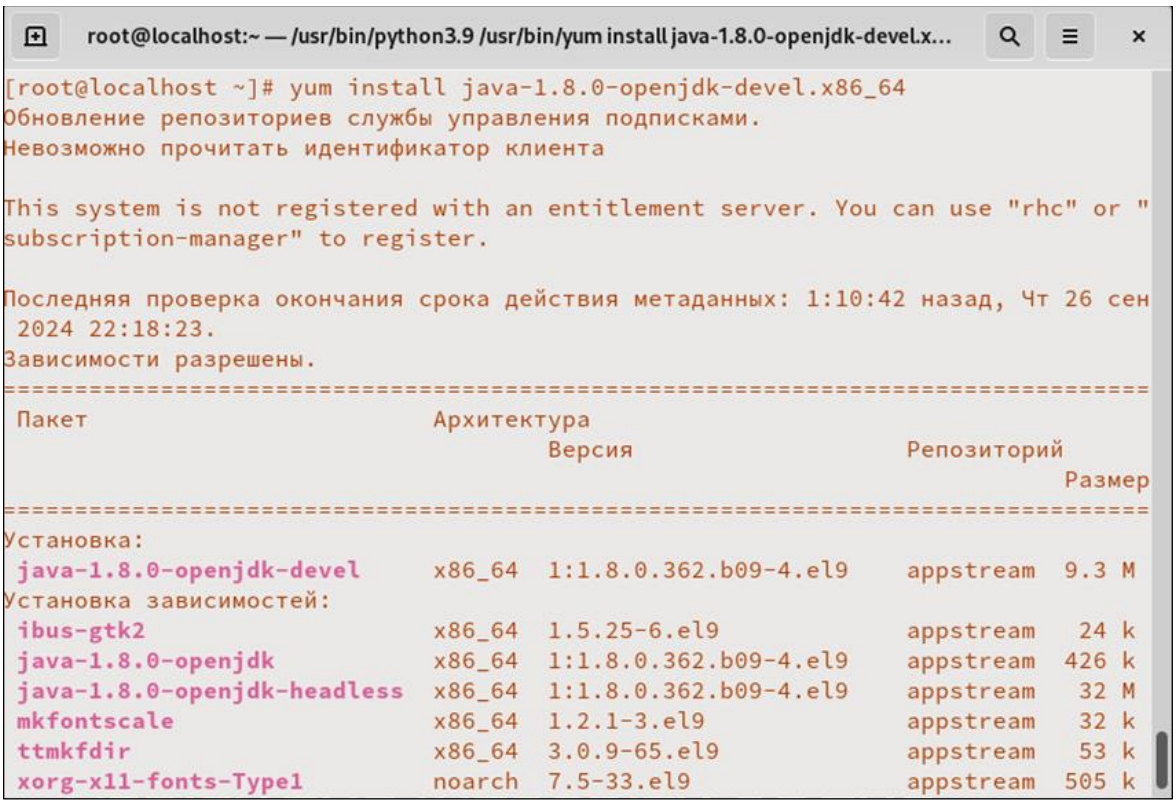


Рисунок 1.33 – Установка Java Development Kit

Следующий шаг – это конфигурирование Hadoop таким образом, чтобы он мог работать в псевдо-распределённом режиме. То есть, когда все сервисы кластеров запущены на одном узле. В документации на официальном сайте есть специальный пункт «Single Node Cluster», где написано, что в некоторых файлах с конфигурацией Hadoop необходимо задать настройки.

Документация предназначена для версии 2.10.2 (рисунок 1.34). Необходимо убедиться, что используемая документация соответствует установленной версии Hadoop, так как данные из нее для других версий могут содержать отличия и не быть применимыми.

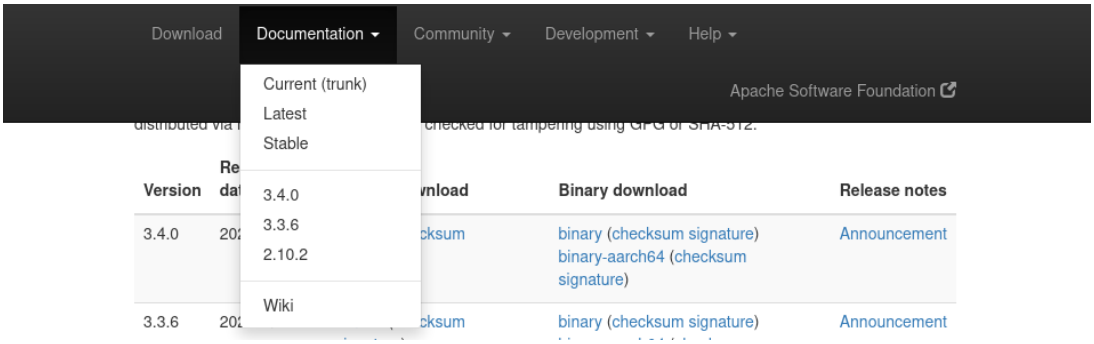


Рисунок 1.34 – Выбор документации по версиям

После перехода в окно с документами (рисунок 1.35), необходимо выбрать пункт «Single Node Setup» в содержании справа.

Рисунок 1.35 – Раздел документации для версии 2.10.2

Далее будет использоваться код, представленный в разделе «Pseudo-Distributed Operation» (рисунок 1.36). Элементы кода, которые представлены подразделе «Configuration» будут нужны в дальнейших действиях работы.

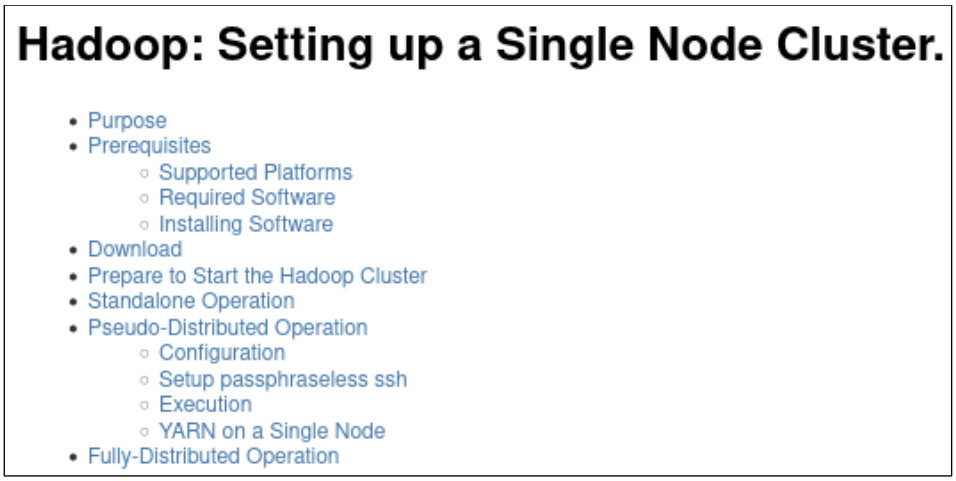


Рисунок 1.36 – Список разделов

В настройках Hadoop указан порт для работы с файловой системой. На рисунке 1.37 представлен переход в директорию hadoop и просмотр её содержимого. Здесь перечислены файлы конфигурации Hadoop, такие как core-site.xml, hdfs-site.xml, yarn-site.xml, и скрипты, такие как hadoop-env.sh, mapred-env.sh, и другие файлы, которые используются для настройки различных аспектов системы Hadoop, включая MapReduce, YARN, и HDFS.

```
[root@localhost ~]# cd /opt/hadoop-2.10.2/etc/
[root@localhost etc]# ls -al
итого 4
drwxr-xr-x. 3 nika nika   20 мая 25  2022 .
drwxr-xr-x. 9 nika nika  149 мая 25  2022 ..
drwxr-xr-x. 2 nika nika 4096 окт 17 17:18 hadoop
[root@localhost etc]# cd hadoop/
[root@localhost hadoop]# ls
capacity-scheduler.xml      httpfs-env.sh            mapred-env.sh
configuration.xsl           httpfs-log4j.properties  mapred-queues.xml.template
container-executor.cfg      httpfs-signature.secret  mapred-site.xml.template
core-site.xml               httpfs-site.xml          slaves
hadoop-env.cmd              kms-acls.xml             ssl-client.xml.example
hadoop-env.sh               kms-env.sh               ssl-server.xml.example
hadoop-metrics2.properties  kms-log4j.properties     yarn-env.cmd
hadoop-metrics.properties   kms-site.xml             yarn-env.sh
hadoop-policy.xml           log4j.properties         yarn-site.xml
hdfs-site.xml               mapred-env.cmd
[root@localhost hadoop]#
```

Рисунок 1.37 – Пункт с элементами кода для настройки

На этот раз в лабораторной редактирование будет осуществляться с применением редактора «nano». Необходимо открыть файл конфигурации «core-site.xml». Изначально конфигурация является пустой, поэтому нужно скопировать туда элемент кода (рисунок 1.38), который находится в браузере в соответствующем пункте ранее открытой документации. Аналогичным образом происходит настройка файла конфигурации «hdfs-site.xml».

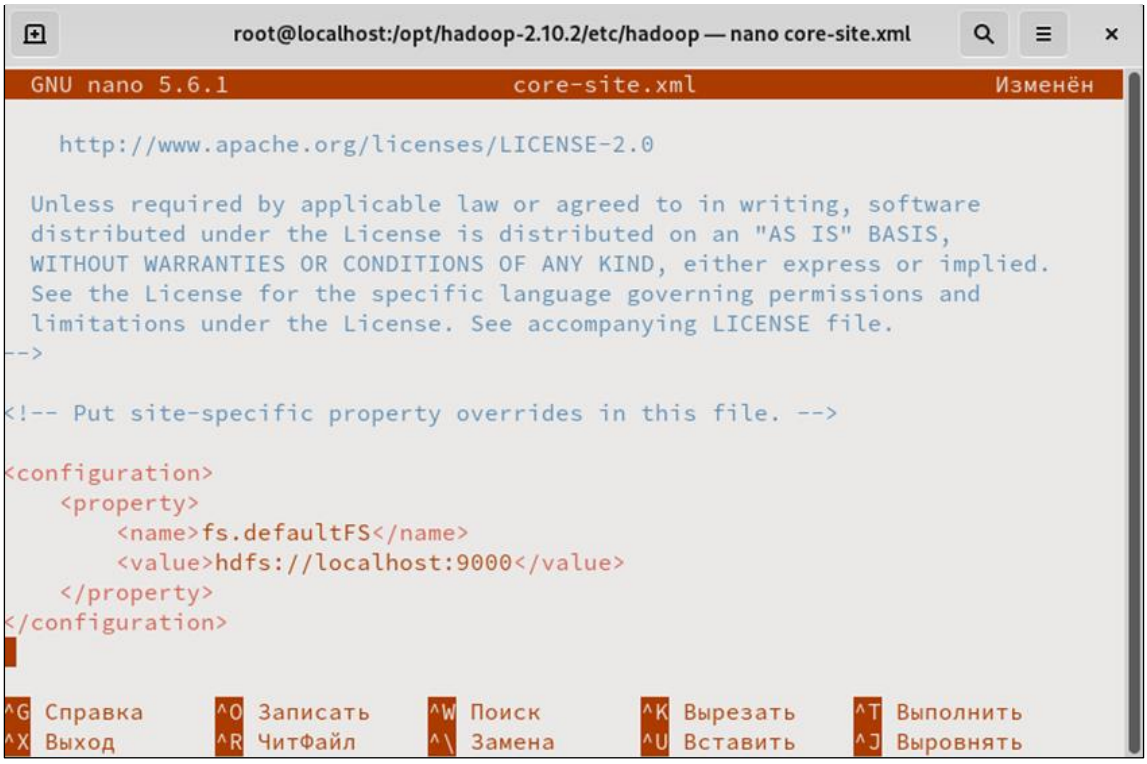


Рисунок 1.38 – Настройка «core-site.xml»

Далее для того, чтобы настроенные сервисы запустились, нужно обеспечить беспарольный доступ на узлы (в данном случае он один). Применяется команда «ssh localhost», после которой запрашивается пароль у пользователя. Чтобы такого больше не происходило, нужно выпустить пару ключей, добавить открытый ключ в список авторизованных, выдать владельцу файла права чтения и записи к этому списку авторизованных ключей и проверить, что войти в систему можно без пароля (рисунок 1.39).

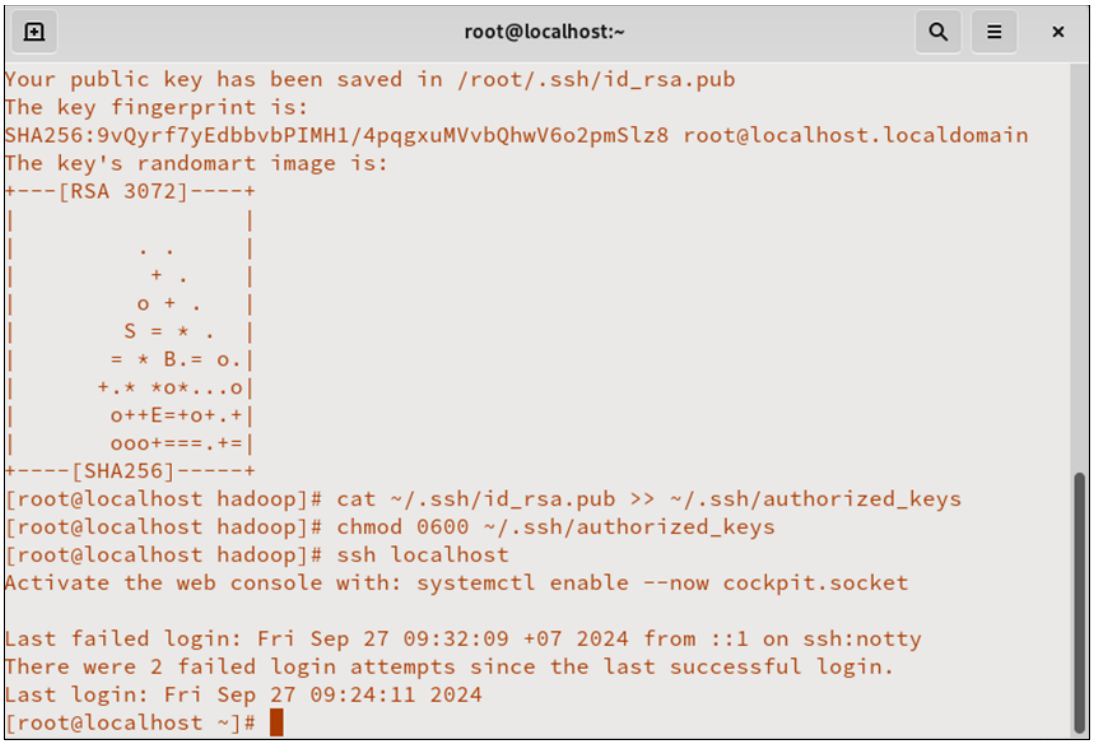


Рисунок 1.39 – Обеспечение беспарольного доступа

Таким логином пользуется Hadoop, когда запускает свои сервисы на узлах, тут он всего один – локальный, поэтому нужен беспарольный доступ.

Далее необходимо отформатировать файловую систему Hadoop, так как инициализация ещё не была проведена. Если терминал перезапускался, нужно ещё раз применить команду «source» к файлу «/etc/profile», чтобы можно было напрямую пользоваться командами.

Следующим шагом надо удостовериться, что пользователь находится в домашней директории Hadoop (/opt/hadoop-2.10.2/), чтобы запустить скрипт: sbin/start-dfs.sh (рисунок 1.40).

Рисунок 1.40 – Запуск скрипта «dfs»

В процессе запуска скрипта используется беспарольное соединение. При вводе команды jps отображается информация о работающих процессах Java на локальной машине. В результате выводится список запущенных сервисов Hadoop: DataNode, NameNode и SecondaryNameNode (рисунок 1.41).

Рисунок 1.41 – Проверка запуска сервисов

У NameNode и DataNode есть свои интерфейсы, которые работают на старых технологиях Java, однако они позволяют выполнять основные операции. В подразделе «Execution» имеется ссылка для перехода в интерфейс NameNode (рисунок 1.42).

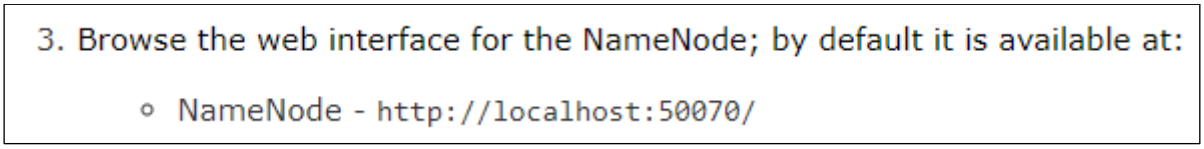


Рисунок 1.42 – Ссылка для перехода в интерфейс

После перехода по ссылке открывается главная страница интерфейса (рисунок 1.43).

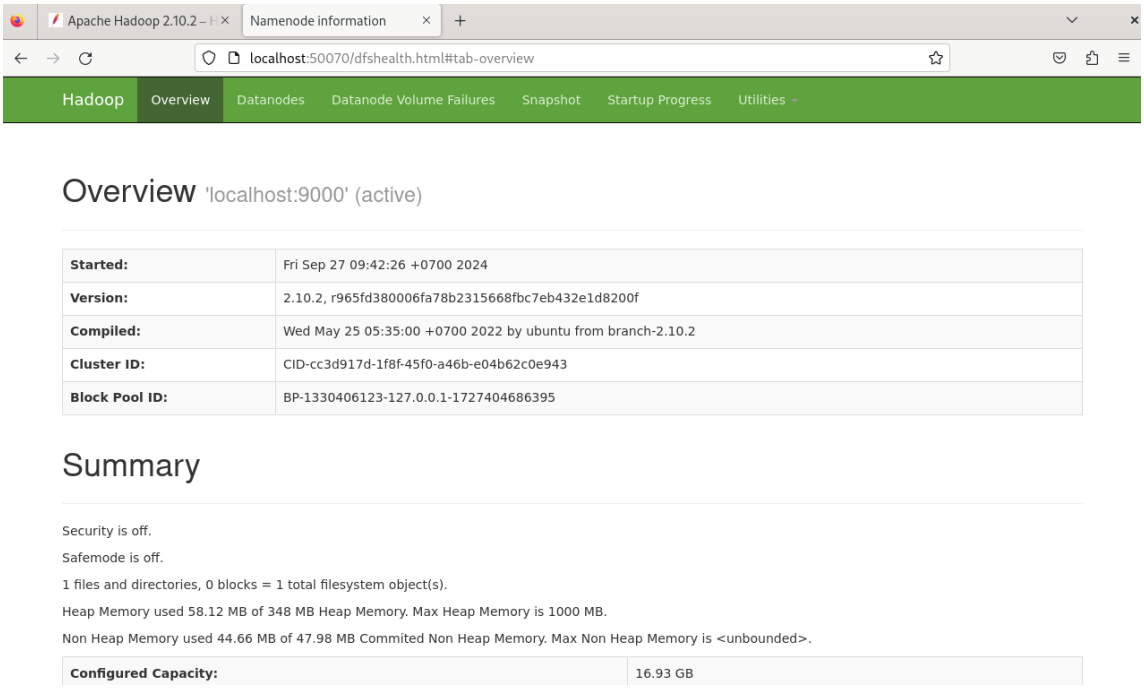


Рисунок 1.43 – Главная страница

Раздел «Live Nodes», в котором представлена информация о кластере (рисунок 1.44).

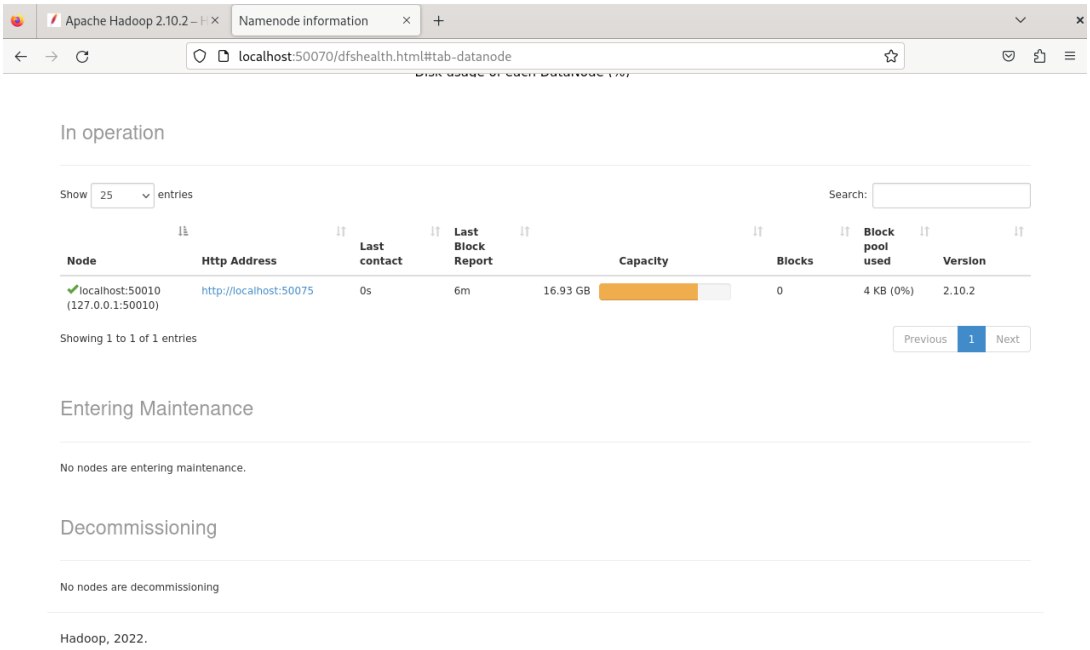


Рисунок 1.44 – Информация о кластере

При нажатии на представленный http-адрес появляется окно с более подробной информацией (рисунок 1.45).

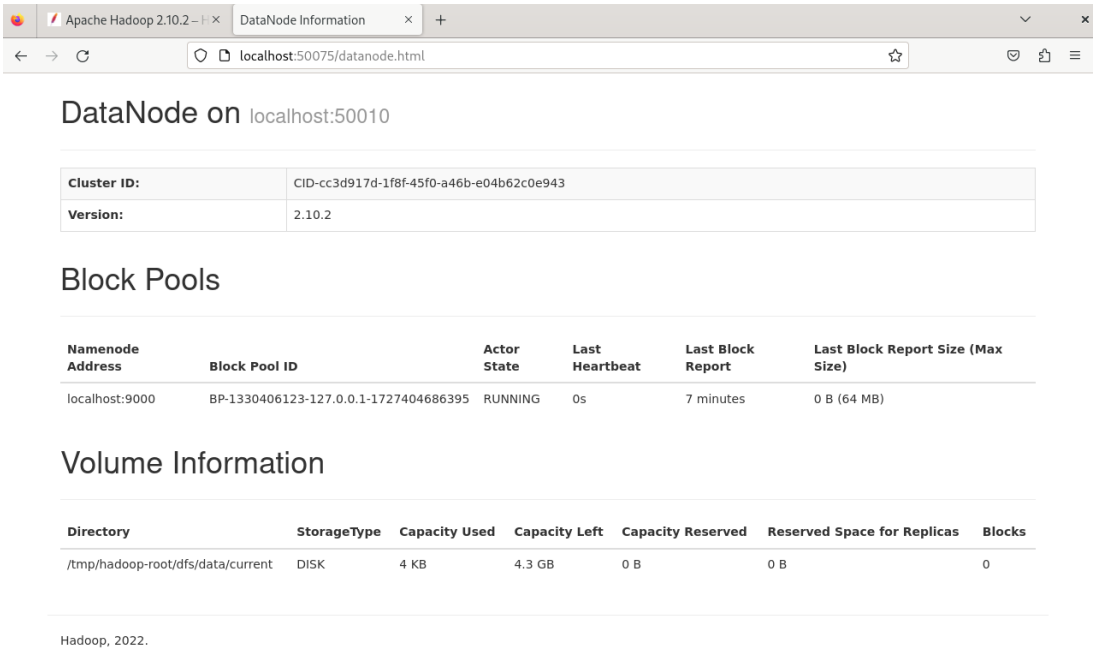


Рисунок 1.45 – Более подробная информация об узлах

В графическом интерфейсе легко можно осуществлять навигацию по файловой системе (рисунок 1.46).

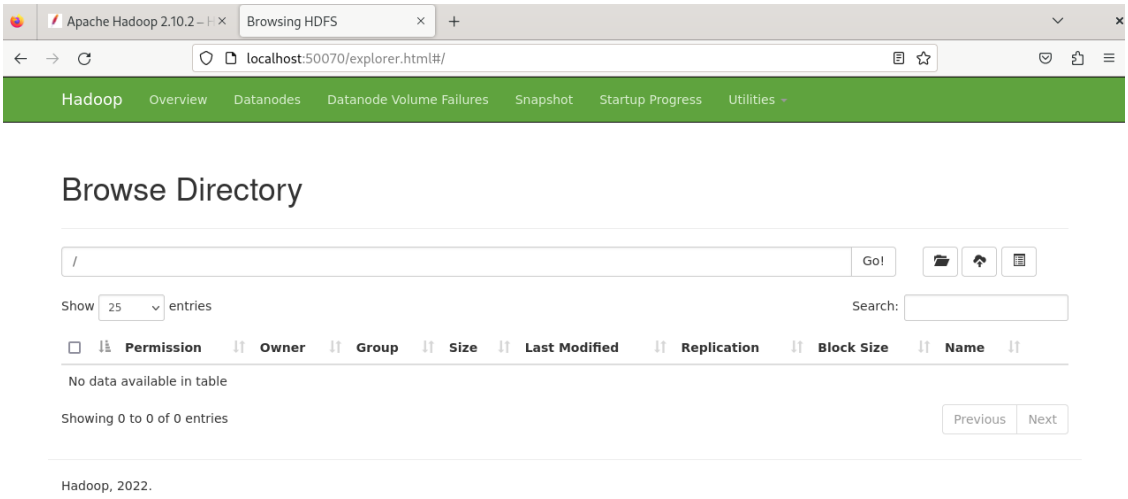


Рисунок 1.46 – Пустая файловая система

По открытой ранее документации имеется инструкция, по которой можно создать систему каталогов. Создается сначала папка «user», затем папка авторизованного пользователя внутри с помощью команды: hdfs dfs –mkdir

/user/root, а далее происходит перемещение директории in в hdfs под новым именем input (рисунок 1.47).

HDFS (Hadoop Distributed File System) — это распределенная файловая система, входящая в состав экосистемы Hadoop. Она предназначена для хранения больших объемов данных, разделенных на множество узлов в кластере, и обеспечивает высокую отказоустойчивость и доступность данных.

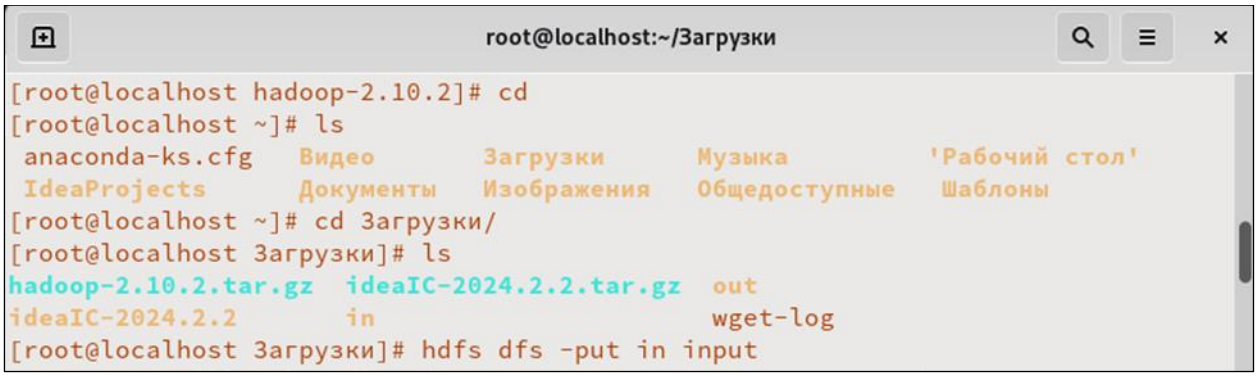


Рисунок 1.47 – Перемещение директории

Теперь, если перейти в файловый менеджер, можно увидеть, что создалась система каталогов (рисунки 1.48 – 1.49).

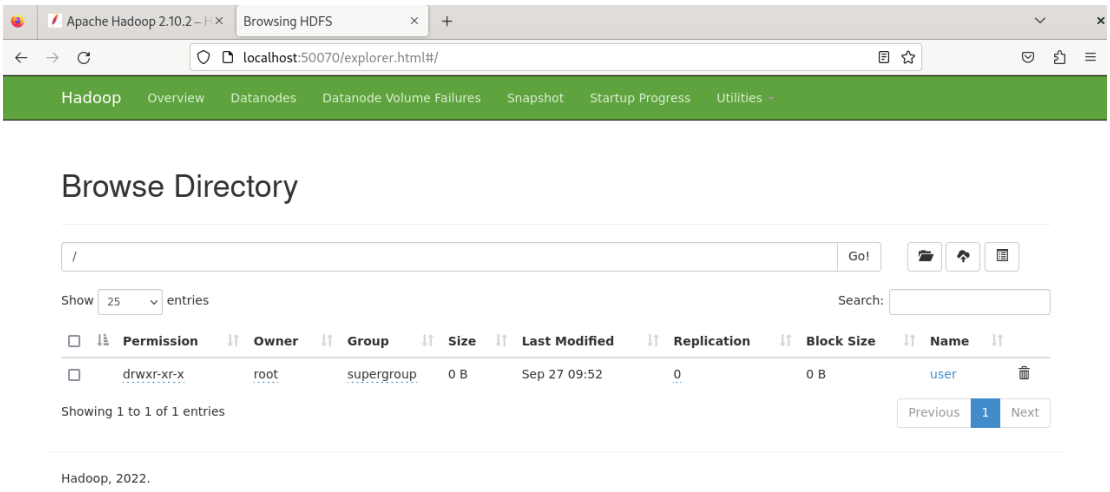


Рисунок 1.48 – Отображение созданного каталога

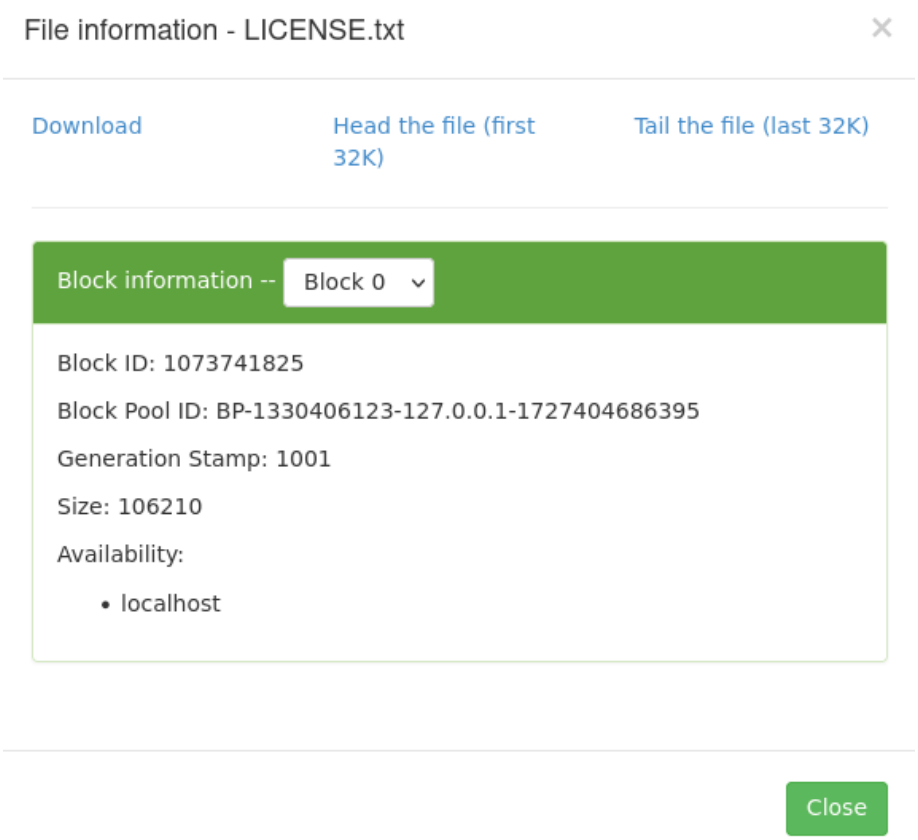


Рисунок 1.49 – Просмотр файла

Далее необходимо вновь запустить расчёт и указать путь к перемещенной в hdfs директории (рисунок 1.50).

```
[root@localhost Загрузки]# hadoop jar /opt/hadoop-2.10.2/share/hadoop/mapreduce/
hadoop-mapreduce-examples-2.10.2.jar wordcount input output
```

Рисунок 1.50 – Путь к директории через hdfs

Можно обновить вкладку с менеджером файлов и увидеть там новый каталог (рисунки 1.51 – 1.52).

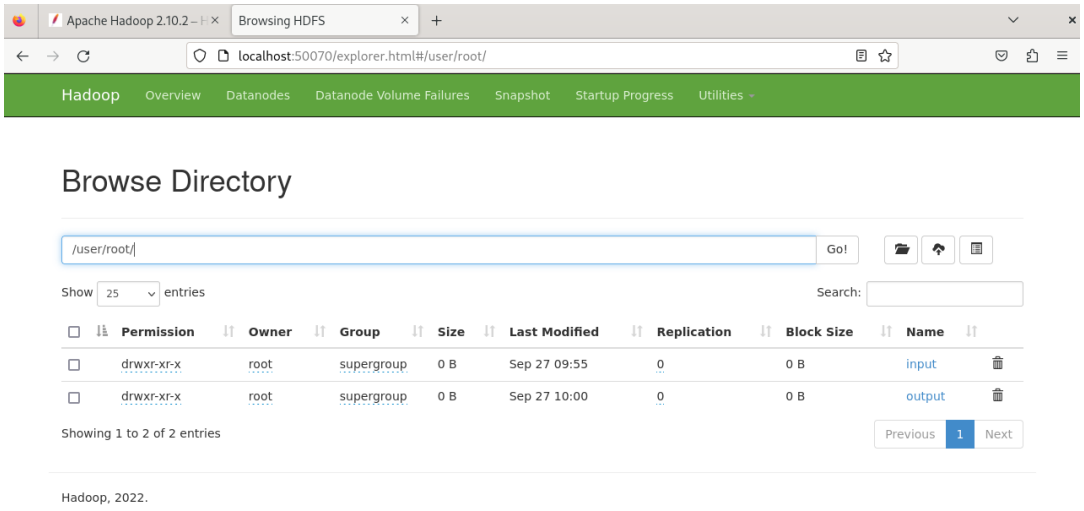


Рисунок 1.51 – Новая директория

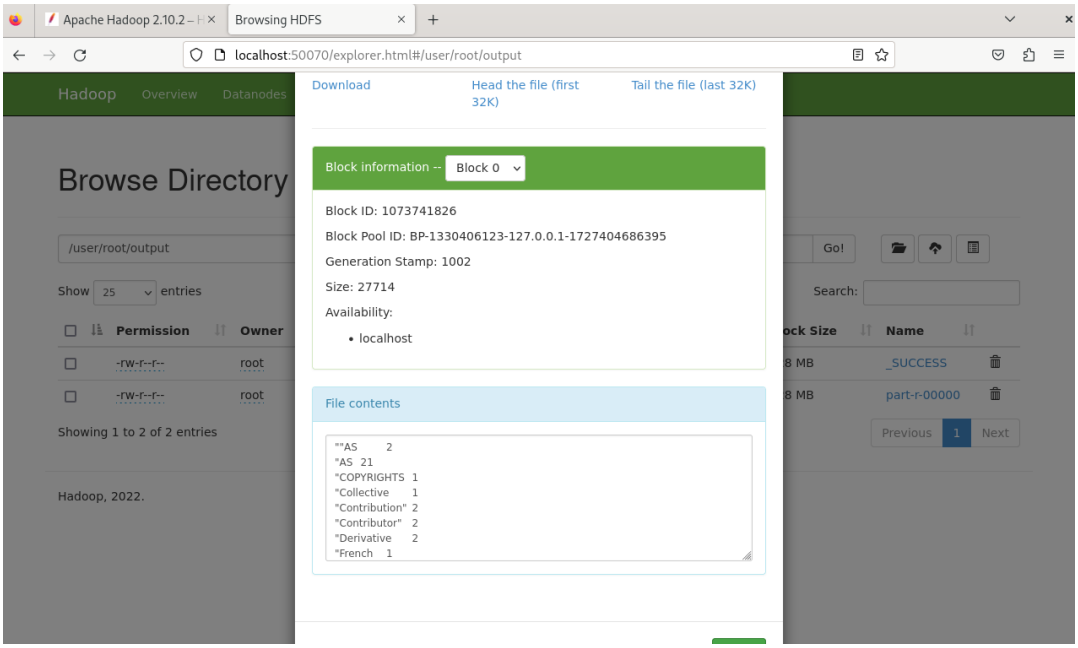


Рисунок 1.52 – Просмотр файла

Также можно воспользоваться командой «get», которая позволяет выгрузить директорию из hdfs в локальную файловую систему (рисунок 1.53).

```
[root@localhost Загрузки]# ls
hadoop-2.10.2.tar.gz  ideaIC-2024.2.2.tar.gz  out
ideaIC-2024.2.2       in                      wget-log
[root@localhost Загрузки]# hdfs dfs -get output output
```

Рисунок 1.53 – Использование специальной команды

В результате можно увидеть те же самые файлы, что лежали в hdfs (рисунок 1.54).

```
[root@localhost Загрузки]# ls
hadoop-2.10.2.tar.gz  ideaIC-2024.2.2.tar.gz  out      wget-log
ideaIC-2024.2.2       in                      output
[root@localhost Загрузки]# ls -al output/
итого 28
drwxr-xr-x. 2 root root    42 сен 27 10:05 .
drwxr-xr-x. 6 root root   138 сен 27 10:05 ..
-rw-r--r--. 1 root root 27714 сен 27 10:05 part-r-00000
-rw-r--r--. 1 root root     0 сен 27 10:05 _SUCCESS
[root@localhost Загрузки]#
```

Рисунок 1.54 – Результат использования команды «get»

С помощью команды «cat» напрямую осуществляется просмотр содержимое файлов в консоли (рисунок 1.55).

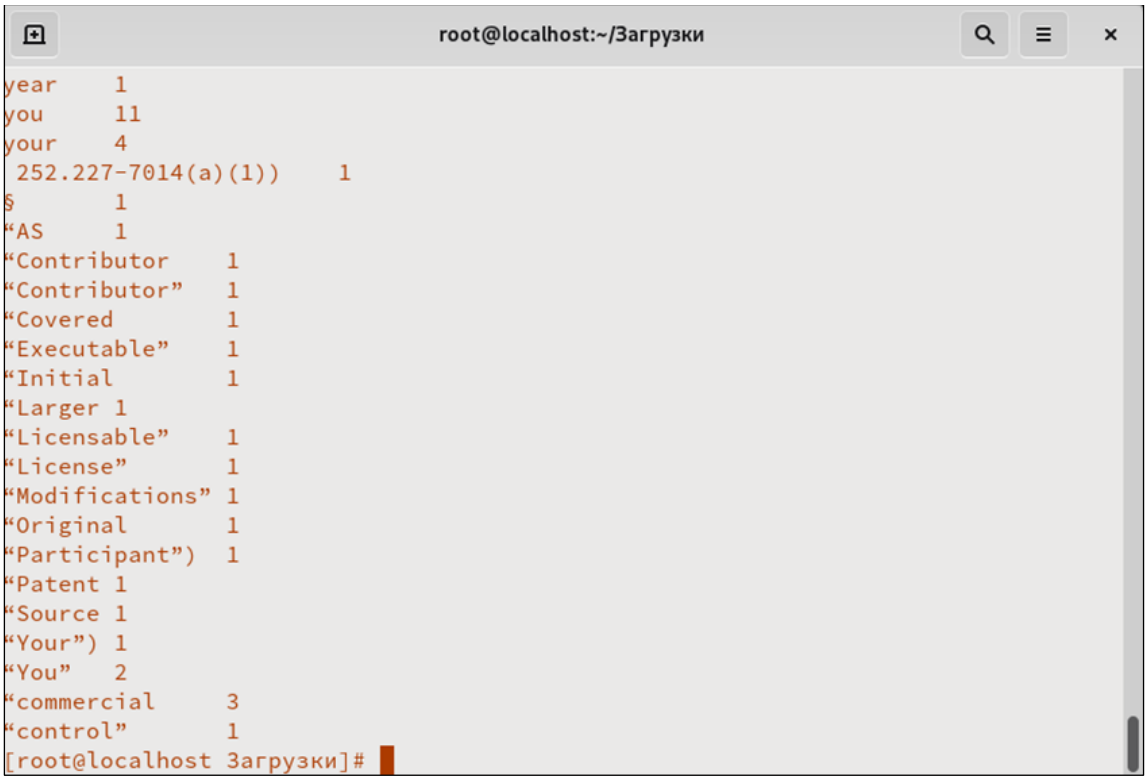


Рисунок 1.55 – Просмотр содержимого файлов

Чтобы остановить все сервисы, нужно воспользоваться скриптом: /opt/Hadoop-2.10.2/sbin/stop-dfs.sh (рисунок 1.56).

Рисунок 1.56 – Завершение работы сервисов и проверка

Контрольные вопросы

1. Что такое Hadoop?
2. Назовите основные компоненты Hadoop.
3. Сколько режимов запуска есть у Hadoop?
4. Что такое HDFS?

ЛАБОРАТОРНАЯ РАБОТА № 5

РАСЧЕТЫ В РАСПРЕДЕЛЕННОМ РЕЖИМЕ

Целью работы является знакомство и освоение навыков работы с новыми сервисами Hadoop 2.0.

В данной работе используется система, настроенная по методическим указаниям четвертой лабораторной работы «Установка hadoop single node».

Краткие теоретические сведения

По сравнению с изначальной в Hadoop версии 2.0 основные изменения коснулись компонента выполнения распределенных вычислений Hadoop MapReduce. Классический Hadoop MapReduce представлял собой один процесс JobTracker и произвольное количество процессов TaskTracker. В новой версии Hadoop MapReduce функции JobTracker по управлению ресурсами и планированию/координации жизненного цикла выполнения заданий были разделены на 2 отдельных компонента: менеджер ресурсов ResourceManager; планировщик и координатор приложения ApplicationMaster.

ResourceManager (RM) – глобальный менеджер ресурсов, чьей задачей является распределение ресурсов, затребованных приложениями и наблюдение за вычислительными узлами, на которых эти приложения выполняются.

Ресурсы у RM запрашиваются для ключевой абстракции – Container, которому можно задать такие параметры как требуемое процессорное время, объем оперативной памяти, необходимая пропускная способность сети. С декабря 2012 года и по настоящее время поддерживается только параметр «объем RAM». Введение RM позволяет относиться к узлам кластера как к вычислительным ресурсам, что качественно повышает утилизацию ресурсов кластера.

ApplicationMaster (AM) – компонент, ответственный за планирование жизненного цикла, координацию и отслеживание статуса выполнения

распределенного приложения. Каждое приложение имеет свой экземпляр ApplicationMaster.

YARN (Yet Another Resource Negotiator) – это программный фреймворк выполнения распределенных приложений (каким экземпляр ApplicationMaster и является). YARN предоставляет компоненты и API, необходимые для разработки распределенных приложений различных типов. Сам фреймворк берет на себя ответственность по распределению ресурсов в ответ на запросы ресурсов от выполняемых приложений и ответственность за отслеживанием статуса выполнения приложений.

Благодаря YARN на Hadoop-кластере возможно запускать не только «map/reduce»-приложения, но и распределенные приложения, созданные с использованием: Open MPI, Spark, Apache HAMA, Apache Giraph, и т.д. Есть возможность реализовать и другие распределенные алгоритмы.

Разделение ответственности по управлению ресурсами и планированию/координации жизненного цикла приложения между компонентами ResourceManager и ApplicationMaster придали платформе Hadoop более распределенный характер. Что, в свою очередь, положительно сказалось на масштабируемости платформы.

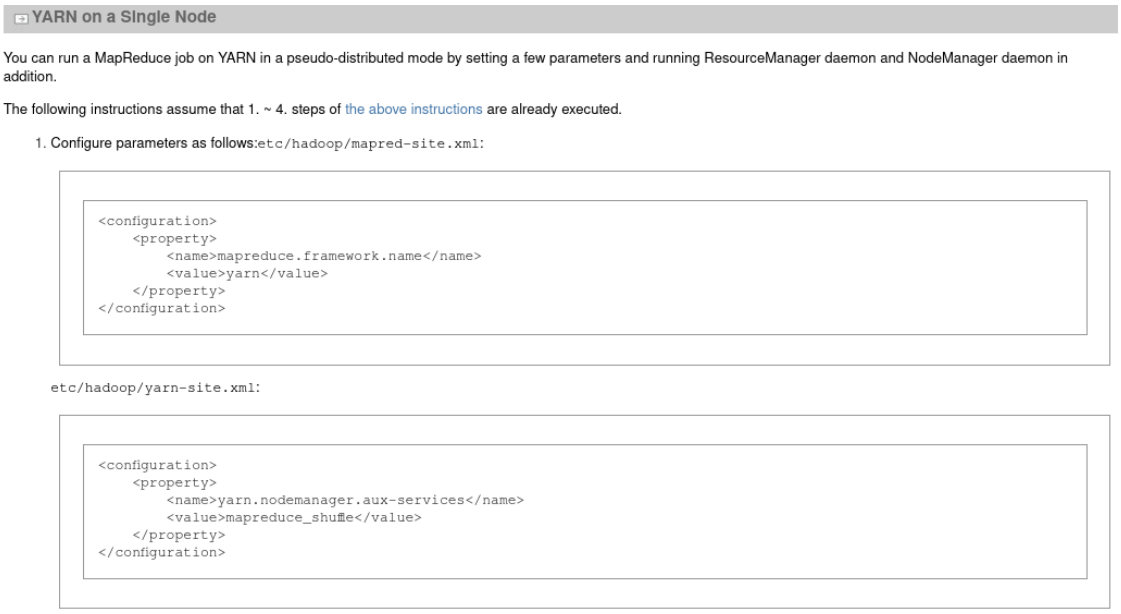
Требования к выполнению лабораторной работы

1. Ознакомиться с методическими указаниями.
2. Выполнить ход работы.
3. Составить отчёт с использованием индивидуальных параметров (к примеру, фон рабочего стола).

1 Ход работы

Перед началом работы необходимо открыть код конфигурации для файлов, отвечающих за выполнение расчетов в Hadoop. Для этого на официальном сайте «Apache Hadoop» (https://hadoop.apache.org/) существует раздел с документацией. В подразделе версии, которая используется в ходе лабораторной работы (версия 2.10.2), выбирается пункт «Single Node Cluster».

В разделе «YARN on a Single Node» представлены шаги для реализации задания MapReduce: изменение файлов конфигурации путем копирования в них элементов, представленных в документации (рисунок 1.1).



YARN on a Single Node

You can run a MapReduce job on YARN in a pseudo-distributed mode by setting a few parameters and running ResourceManager daemon and NodeManager daemon in addition.

The following instructions assume that 1. ~ 4. steps of the above instructions are already executed.

1. Configure parameters as follows:`etc/hadoop/mapred-site.xml`:

```
<configuration>
    <property>
        <name>mapreduce.framework.name</name>
        <value>yarn</value>
    </property>
</configuration>
```

`etc/hadoop/yarn-site.xml`:

```
<configuration>
    <property>
        <name>yarn.nodemanager.aux-services</name>
        <value>mapreduce_shuffle</value>
    </property>
</configuration>
```

Рисунок 1.1 – Раздел документации «YARN on a Single Node»

Следующим шагом выполняется возврат в папку etc Hadoop с помощью команд, представленных на рисунке 1.2.

```
[root@localhost ~]# cd /opt/hadoop-2.10.2/etc/
[root@localhost etc]# ls -al
итого 4
drwxr-xr-x. 3 nika nika   20 мая 25  2022 .
drwxr-xr-x. 9 nika nika  149 мая 25  2022 ..
drwxr-xr-x. 2 nika nika 4096 окт 17 17:18 hadoop
[root@localhost etc]# cd hadoop/
```

Рисунок 1.2 – Путь в папку Hadoop

Для просмотра содержимого папки используется команда «ls -al». Соответственно в директории hadoop находится файл «mapred-site.xml.template» (рисунок 1.3). Настроив параметры данного файла, можно оптимизировать работу MapReduce и адаптировать систему под конкретные потребности.

```
-rw-r--r--. 1 nika nika   758 мая 25  2022 mapred-site.xml.template
-rw-r--r--. 1 nika nika    10 мая 25  2022 slaves
-rw-r--r--. 1 nika nika  2316 мая 25  2022 ssl-client.xml.example
-rw-r--r--. 1 nika nika  2697 мая 25  2022 ssl-server.xml.example
-rw-r--r--. 1 nika nika  2250 мая 25  2022 yarn-env.cmd
-rw-r--r--. 1 nika nika  4876 мая 25  2022 yarn-env.sh
-rw-r--r--. 1 nika nika   690 мая 25  2022 yarn-site.xml
[root@localhost hadoop]#
```

Рисунок 1.3 – Файл «mapred-site.xml.template»

Для того, чтобы файл, как представлено в документации, назывался «mapred-site.xml», его нужно переименовать новым именем в открытой папке, используя команду «mv mapred-site.xml.template mapred-site.xml».

После чего происходит редактирование файла с помощью уже известного редактора nano. В секцию «configuration» необходимо скопировать элемент «property» из ранее открытой на сайте документации (рисунок 1.4).

```
  GNU nano 5.6.1                    mapred-site.xml                    Изменён
  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>
    <property>
        <name>mapreduce.framework.name</name>
        <value>yarn</value>
    </property>
</configuration>

^G Справка   ^O Записать  ^W Поиск     ^K Вырезать  ^T Выполнить ^C Позиция
^X Выход     ^R ЧитФайл   ^\ Замена    ^U Вставить  ^J Выровнять ^_ К строке
```

Рисунок 1.4 – Редактирование файла «mapred-site.xml»

Аналогичным способом проводится настройка файла «yarn-site.xml» (рисунок 1.5).

```
<configuration>

<!-- Site specific YARN configuration properties -->
    <property>
        <name>yarn.nodemanager.aux-services</name>
        <value>mapreduce_shuffle</value>
    </property>
</configuration>
```

Рисунок 1.5 – Редактирование файла «yarn-site.xml»

Далее из домашней директории Hadoop (/opt/hadoop-2.10.2/) происходит запуск сервисов распределенной файловой системы «dfs» и сервисов «yarn» с помощью скриптов «sbin/start-dfs.sh» и «sbin/start-yarn.sh» соответственно (рисунок 1.6). Аналогично запускаются сервисы «yarn».

```
[root@localhost hadoop]# cd ../..
[root@localhost hadoop-2.10.2]# sbin/start-dfs.sh
```

Рисунок 1.6 – Запуск сервисов «dfs»

Таким образом при использовании команды «jps» можно будет увидеть все запущенные сервисы (рисунок 1.7). К ранее использованным сервисам добавилось два новых: NodeManager и ResourceManager. Они принадлежат Yarn.

NodeManager (NM) – агент, запущенный на вычислительном узле, в чьи обязанности входит:

– отслеживание используемых вычислительных ресурсов (CPU, RAM, network, etc.);

– отправка отчетов по используемым ресурсам планировщику менеджера ресурсов ResourceManager/Scheduler.

```
[root@localhost hadoop-2.10.2]# jps
42453 NodeManager
42535 Jps
41768 NameNode
42345 ResourceManager
41899 DataNode
42075 SecondaryNameNode
[root@localhost hadoop-2.10.2]#
```

Рисунок 1.7 – Список запущенных сервисов

После запуска сервисов стал доступен графический интерфейс, управляющий ResourceManager (рисунок 1.8).

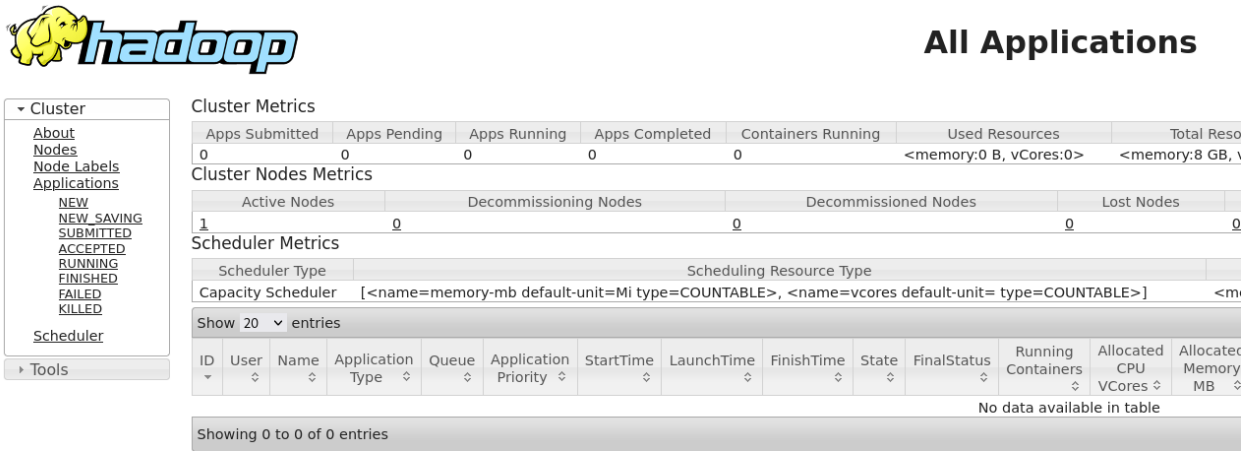


Рисунок 1.8 – Графический интерфейс, позволяющий просматривать задачи

Перейти в него можно с помощью адреса, представленного так же на сайте документации (рисунок 1.9).

Рисунок 1.9 – Адрес для перехода в графический интерфейс

Так как сайт сейчас только стартовал, запущенных задач на нем нет. Для демонстрации его работы, нужно запустить пример с «wordcount» на свежераспределенном кластере Hadoop. В рамках данной работы выполняется переход в директорию «Загрузки». Переход не является обязательным для работы с HDFS и выполнялся строго для удобства.

Перед запуском примера стоит проверить загруженность узла при выполнении тех или иных операций с файловой системой Hadoop (рисунок 1.10).

```
[root@localhost hadoop-2.10.2]# cd
[root@localhost ~]# cd Загрузки/
[root@localhost Загрузки]# hdfs dfs -ls
Found 2 items
drwxr-xr-x   - root supergroup          0 2024-10-17 17:50 input
drwxr-xr-x   - root supergroup          0 2024-10-17 17:54 output
[root@localhost Загрузки]#
```

Рисунок 1.10 – Команда, выполняющая листинг директории hdfs

После выполнения операций, представленных на рисунке 1.10, необходимо открыть «Системный монитор» и перейти в раздел «Ресурсы». Это системная утилита, которая отображает загруженность процессора и состояния памяти, работу с сетью.

Далее в рамках работы проводится исследование. Нужно выполнить листинг той директории, которой на самом деле не существует, и проверить состояние загруженности процессора. Аналогичные действия проводятся с уже существующей директорией для сравнения загруженности (рисунок 1.11).

```
[root@localhost Загрузки]# hdfs dfs -ls /user/output
ls: `/user/output': No such file or directory
[root@localhost Загрузки]# hdfs dfs -ls /user/root/output
```

Рисунок 1.11 – Листинг двух директорий

В результате получается два схожих скачка, которые говорят о том, что процессор реагирует на каждое действие пользователя с файловой системой, которое существенно нагружает процессор на общем фоне и обладает большой задержкой (рисунки 1.12 – 1.13).

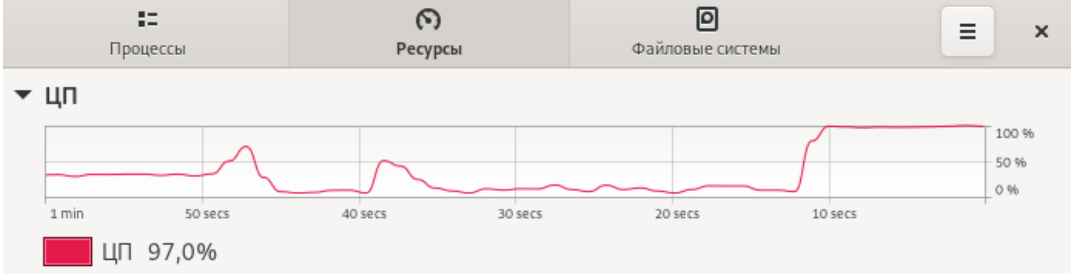


Рисунок 1.12 – Загруженность при листинге несуществующей директории

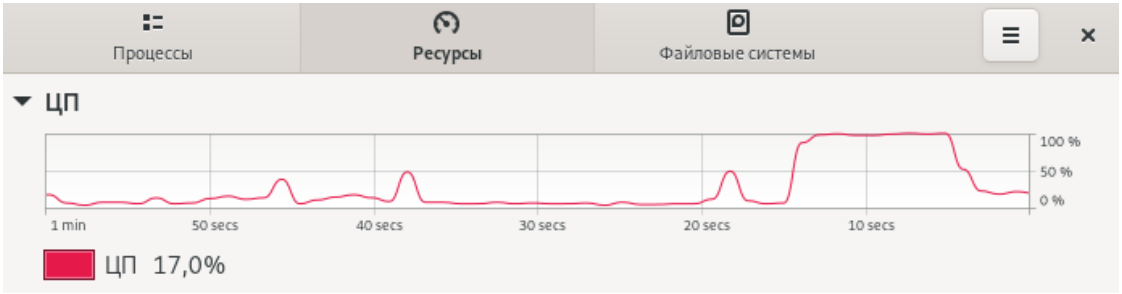


Рисунок 1.13 – Загруженность при листинге существующей директории

Следующим шагом происходит рекурсивное удаление директории «output» с помощью команды «-rm -r». Аналогично удаляется эта директория из локальной файловой системы (рисунки 1.14), и практически не наблюдается нагрузка на процессор.

```
[root@localhost Загрузки]# hdfs dfs -rm -r output
Deleted output
[root@localhost Загрузки]# ls
hadoop-2.10.2.tar.gz  ideaIC-2024.2.3.tar.gz  in  out  output
[root@localhost Загрузки]# rm -rf output/
[root@localhost Загрузки]# yarn jar /opt/hadoop-2.10.2/share/hadoop/mapreduce/ha
doop-mapreduce-examples-2.10.2.jar wordcount input output
```

Рисунок 1.14 – Рекурсивное удаление директории «output»

Наконец можно запустить пример «wordcount», придав в качестве входных параметров директорию «input» и имя выходной директории «output», которой больше в распределенной файловой системе Hadoop нет. Полная команда: «yarn jar /opt/hadoop-2.10.2/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.10.2.jar wordcount input output».

После запуска задачи можно воспользоваться графическим интерфейсом ResourceManager и увидеть изменения (рисунок 1.15).

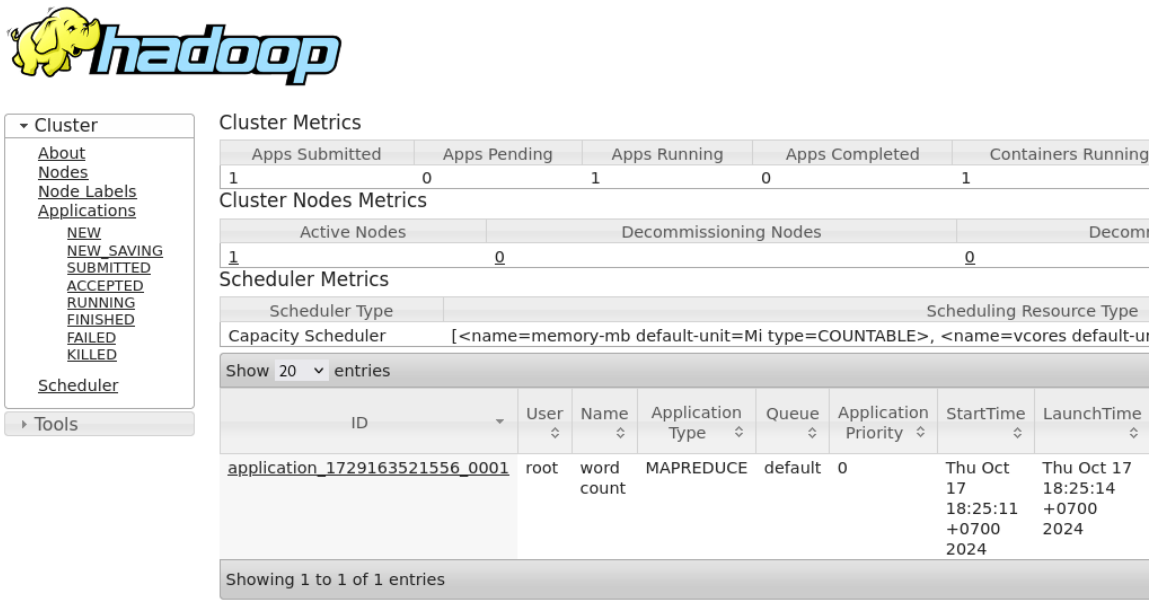


Рисунок 1.15 – ID запущенной задачи

После перехода по появившемуся индикатору задачи можно узнать подробную информацию (рисунок 1.16).

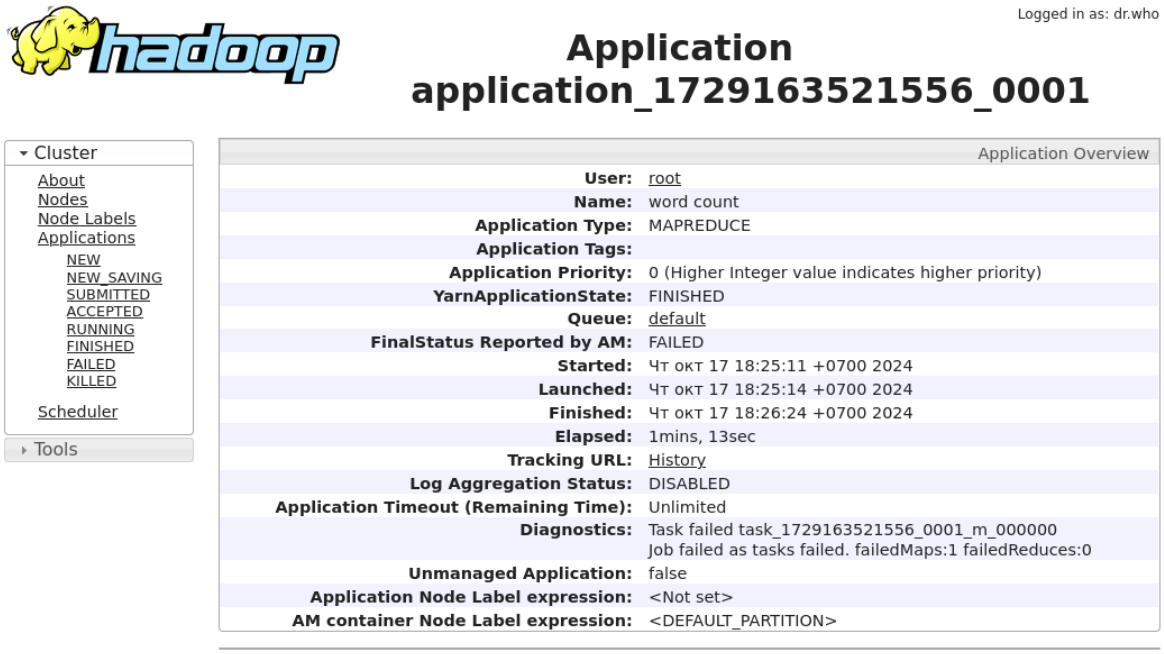


Рисунок 1.16 – Информация о выполненной задаче

После завершения задачи она перемещается из раздела запущенных задач в раздел выполненных (рисунок 1.17).

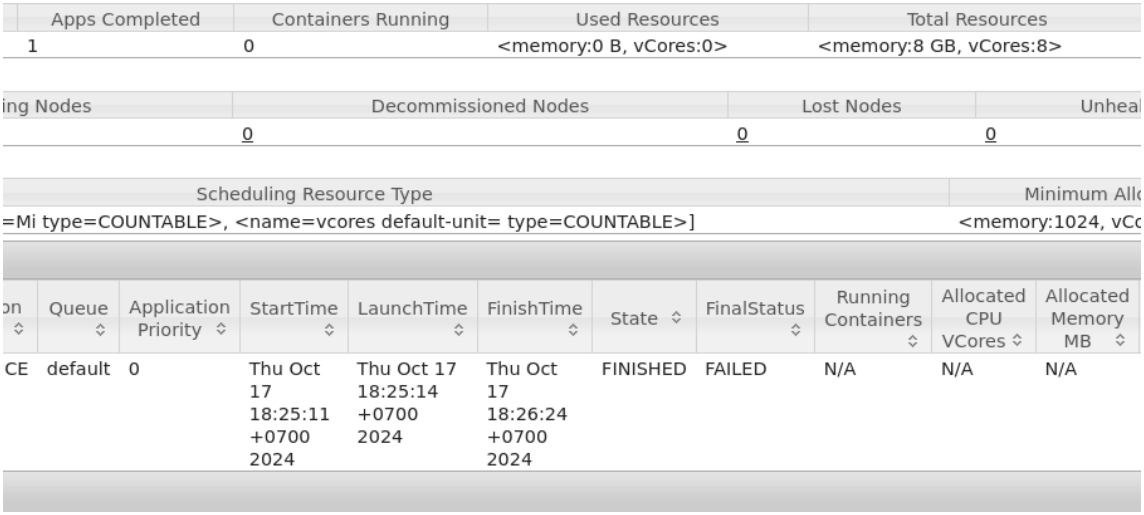


Рисунок 1.17 – Задача в разделе завершенных

В данном случае задача не была выполнена успешно из-за недостаточности ресурсов. Подобный результат допускается в работе.

Во время работы с файловой системой была определенная нагрузка на CPU (рисунок 1.18).

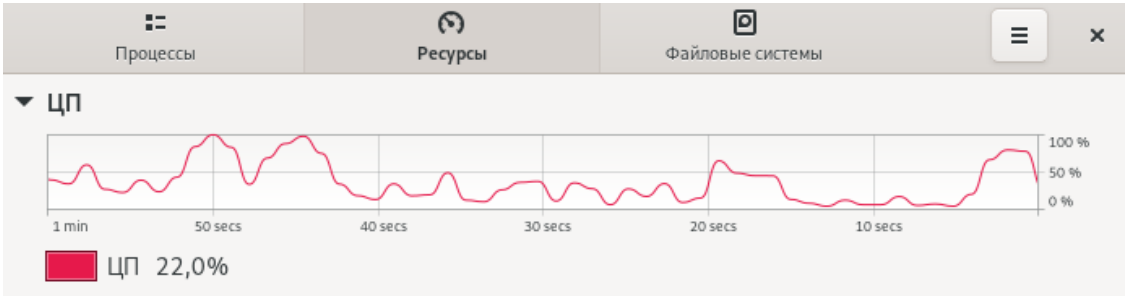


Рисунок 1.18 – Нагрузка на процессор

Контрольные вопросы

1. Какие два основных компонента выделили в новой архитектуре Hadoop из функции JobTracker?

2. Что такое ResourceManager?

3. Что такое ApplicationMaster?

4. Назовите особенности программного фреймворка Yarn.