МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Методические указания
по выполнению лабораторных работ

по дисциплине **«Большие данные»**
для студентов направления подготовки
02.03.01«Математика и компьютерные науки»
направленность (профиль)
«Анализ данных и искусственный интеллект»

Введение

Лабораторные работы - это целенаправленная форма организации педагогического процесса, направленная на углубление научно-теоретических знаний и овладение определенными методами работы, в процессе которых вырабатываются умения и навыки выполнения тех или иных учебных действий в данной сфере науки.

Лабораторные работы предназначены для углубленного изучения учебных дисциплин и играют важную роль в выработке у студентов умений и навыков применения полученных знаний для решения практических задач совместно с педагогом. Кроме того, они развивают научное мышление и речь, позволяют проверить знания студентов и выступают как средства оперативной обратной связи.

Цель лабораторных работ - углублять, расширять, детализировать знания, полученные на лекции, в обобщенной форме и содействовать выработке навыков профессиональной деятельности.

Данные методические указания к лабораторным работам разработаны в соответствии с рабочей программой по дисциплине «Большие данные» для студентов направления подготовки 02.03.01 «Математика и компьютерные науки».

Лабораторная работа 1

Тема: **Введение в BigData**.

Цель: Ознакомиться с концепцией BigData, ее характеристиками и примерами использования. Получить практический опыт работы с большими наборами данных с использованием простых инструментов (Excel, GoogleSheets). Научиться извлекать полезную информацию из данных и проводить базовый анализ.

Задание 1. Выбор открытого набора данных:

Выберите один из следующих открытых наборов данных (или найдите другой, соответствующий требованиям):

Kaggle Datasets: https://www.kaggle.com/datasets

Google Dataset Search: https://datasetsearch.research.google.com/

UCI Machine Learning Repository: http://archive.ics.uci.edu/ml/index.php

Рекомендуемые наборы данных для этой работы:

"London Bike Sharing Dataset" (Kaggle)

"Titanic - Machine Learning from Disaster" (Kaggle)

"Iris Dataset" (UCI Machine Learning Repository)

Размер набора данных должен быть достаточно большим, чтобы ощутить преимущества BigData, но при этом достаточно малым, чтобы его можно было открыть и обработать в Excel или GoogleSheets (обычно несколько мегабайт).

Задание 2. Описание выбранного набора данных: ветвление и слияние:

Опишите выбранный набор данных:

1. Название набора данных.
2. Источник данных.
3. Размер набора данных.
4. Количество строк и столбцов.
5. Описание столбцов (тип данных, что они означают).

Задание 3: Загрузка и открытие данных в Excel или GoogleSheets:

Скачайте выбранный набор данных. Откройте его в Excel или GoogleSheets. Если данные слишком велики для Excel, попробуйте использовать GoogleSheets или сократите набор данных, оставив только часть строк.

Задание 4. Очистка данных (при необходимости):

Удалите дубликаты строк. Замените пустые значения на подходящие значения (например, среднее значение для числовых столбцов). Исправьте ошибки в данных (например, опечатки).

Задание 5. Базовый анализ данных:

Рассчитайте основные статистические показатели для числовых столбцов (среднее значение, медиана, минимум, максимум, стандартное отклонение). Постройте графики и диаграммы для визуализации данных: Гистограммы для отображения распределения данных. Диаграммы рассеяния для выявления зависимостей между двумя переменными. Круговые диаграммы для отображения долей. Найдите интересные закономерности и аномалии в данных.

Контрольные вопросы:

1. Что такое BigData?
2. Перечислите и опишите основные характеристики BigData (5V).
3. Приведите примеры использования BigData в различных отраслях.
4. Какие инструменты можно использовать для работы с большими наборами данных?
5. Какие задачи можно решить с помощью анализа данных?
6. Какие проблемы могут возникнуть при работе с большими наборами данных?

Отчет должен содержать:

Титульный лист (название работы, ФИО, группа). Цель работы. Краткое описание выполненных заданий. Описание выбранного набора данных. Описание процесса загрузки и очистки данных. Результаты базового анализа данных (статистические показатели, графики, диаграммы). Выводы. Ответы на контрольные вопросы. Выводы о проделанной работе.

Лабораторная работа 2

Тема: **Основы работы с данными.**

Цель: Изучить типы данных (структурированные, неструктурированные, полуструктурированные) и форматы (CSV, JSON).Приобрести практические навыки работы с библиотекой Pandas в Python для чтения, записи и выполнения простых операций с данными. Научиться фильтровать, сортировать и группировать данные с использованием Pandas.

Задание 1. Чтение данных из CSV-файла

Установите библиотеку Pandas с помощью pip:

pipinstallpandas

Скачайте CSV-файл с данными (например, с Kaggle: https://www.kaggle.com/datasets). Можно использовать набор данных "Titanic - MachineLearningfromDisaster".

Импортируйте библиотеку Pandas в Python:

importpandasaspd

Прочитайте данные из CSV-файла в DataFrame:

df = pd.read_csv("titanic.csv")

Выведите первые несколько строк DataFrame:

print(df.head())

Выведите информацию о DataFrame (типы данных, количество строк и столбцов):

print(df.info())

Задание 2. Чтение данных из JSON-файла:

Создайте JSON-файл с данными (или скачайте готовый). Пример:

```
[
{"name": "Alice", "age": 25, "city": "New York"},
{"name": "Bob", "age": 30, "city": "London"},
{"name": "Charlie", "age": 28, "city": "Paris"}
]
```

Прочитайте данные из JSON-файла в DataFrame:

df = pd.read_json("data.json")

ВыведитеDataFrame.

Задание 3. Фильтрация данных:

Отфильтруйте DataFrame, чтобы выбрать только строки, соответствующие определенному условию. Пример:

survived_passengers = df[df["Survived"] == 1]

print(survived_passengers.head())

Примените несколько условий для фильтрации:

 # Выбрать всех пассажиров "Titanic", которые выжили и были женского пола (Sex == "female")

female_survivors = df[(df["Survived"] == 1) & (df["Sex"] == "female")]

print(female_survivors.head())

Задание 4. Сортировка данных:

Отсортируйте DataFrame по определенному столбцу:

 # Отсортировать пассажиров "Titanic" по возрасту в порядке возрастания

sorted_by_age = df.sort_values(by="Age")

print(sorted_by_age.head())

 # Отсортировать пассажиров "Titanic" по возрасту в порядке убывания

sorted_by_age_desc = df.sort_values(by="Age", ascending=False)

print(sorted_by_age_desc.head())

Задание 5. Группировка данных:

Сгруппируйте DataFrame по определенному столбцу и рассчитайте статистические показатели для каждой группы:

 # Сгруппировать пассажиров "Titanic" по полу и рассчитать средний возраст для каждой группы

grouped_by_sex = df.groupby("Sex")["Age"].mean()

print(grouped_by_sex)

Примените несколько функций агрегации:

 # Сгруппировать пассажиров "Titanic" по классу билета (Pclass) и рассчитать количество выживших и средний возраст для каждой группы

grouped_by_pclass = df.groupby("Pclass").agg({"Survived": "sum", "Age": "mean"})

print(grouped_by_pclass)

Задание 6. Запись данных в CSV-файл:

Запишите отфильтрованный или сгруппированный DataFrame в новый CSV-файл:

female_survivors.to_csv("female_survivors.csv", index=False) # index=False чтобы не записывать индексы в файл

Контрольные вопросы:

1. Какие типы данных вы знаете?
2. В чем разница между структурированными, неструктурированными и полуструктурированными данными?
3. Какие форматы данных вы знаете?
4. Что такое CSV и JSON?
5. Что такое Pandas и зачем он нужен?
6. Что такое Series и DataFrame в Pandas?
7. Как прочитать данные из CSV-файла в DataFrame?
8. Как отфильтровать, отсортировать и сгруппировать данные с использованием Pandas?
9. Как записать DataFrame в CSV-файл?

Отчет должен содержать:

Титульный лист (название работы, ФИО, группа). Цель работы. Краткое описание выполненных заданий. Описание типов данных и форматов данных. Код Python, используемый для чтения, фильтрации, сортировки и группировки данных. Результаты выполнения кода (вывод DataFrame, стати-

стические показатели). Ответы на контрольные вопросы. Выводы о проделанной работе.

Лабораторная работа 3

Тема: **Реляционные базы данных.**

Цель работы: Ознакомиться с концепцией реляционных баз данных и языком SQL. Изучить основные команды SQL (SELECT, WHERE, GROUP BY) и их применение для анализа данных. Приобрести практические навыки работы с SQLite для создания таблиц и выполнения запросов.

Задание 1. Установка SQLite:

SQLite обычно предустановлен в большинстве операционных систем (Linux, macOS). Если SQLite не установлен, скачайте и установите его с официального сайта: https://www.sqlite.org/download.html

Для работы с SQLite из Python установите библиотеку sqlite3:

pipinstall sqlite3

Задание 2. Подключение к базе данных и создание таблицы:

Импортируйте библиотеку sqlite3 в Python:

import sqlite3

Подключитесь к базе данных (или создайте новую базу данных):

conn = sqlite3.connect('mydatabase.db')

Создайте таблицу students со столбцами id, name, age, city:

```
cursor = conn.cursor()
cursor.execute('''
CREATE TABLE IF NOT EXISTS students (
id INTEGER PRIMARY KEY,
name TEXT,
age INTEGER,
city TEXT
)
''')
conn.commit()
```

Задание 3. Добавление данных в таблицу:

Добавьте несколько записей в таблицу students:

cursor.execute("INSERT INTO students (name, age, city) VALUES ('Alice', 20, 'New York')")

cursor.execute("INSERT INTO students (name, age, city) VALUES ('Bob', 22, 'London')")

cursor.execute("INSERT INTO students (name, age, city) VALUES ('Charlie', 21, 'Paris')")

cursor.execute("INSERT INTO students (name, age, city) VALUES ('David', 23, 'New York')")

conn.commit()

Задание 4.Выполнение запросов SELECT:

Выберите все данные из таблицы students:

```
cursor.execute("SELECT * FROM students")
results = cursor.fetchall()
for row in results:
print(row)
```

Выберите столбцы name и age из таблицы students:

```
cursor.execute("SELECT name, age FROM students")
results = cursor.fetchall()
for row in results:
print(row)
```

Задание 5. Фильтрация данных с помощью WHERE:

Выберите всех студентов из города "NewYork":

```
cursor.execute("SELECT * FROM students WHERE city = 'New York'")
results = cursor.fetchall()
for row in results:
print(row)
```

Выберите всех студентов старше 21 года:

```
cursor.execute("SELECT * FROM students WHERE age > 21")
```

```
results = cursor.fetchall()
for row in results:
print(row)
```

Задание 6. Сортировка данных с помощью ORDER BY:

Выберите всех студентов и отсортируйте их по возрасту в порядке возрастания:

```
cursor.execute("SELECT * FROM students ORDER BY age")
results = cursor.fetchall()
for row in results:
print(row)
```

Выберите всех студентов и отсортируйте их по возрасту в порядке убывания:

```
cursor.execute("SELECT * FROM students ORDER BY age DESC")
results = cursor.fetchall()
for row in results:
print(row)
```

Задание 7. Группировка данных с помощью GROUP BY:

Подсчитайте количество студентов в каждом городе:

```
cursor.execute("SELECT city, COUNT(*) FROM students GROUP BY city")
results = cursor.fetchall()
for row in results:
print(row)
```

Закрытие соединения с базой данных:

```
conn.close()
```

Контрольные вопросы:

1. Что такое параллельные вычисления и зачем они нужны?
2. Что такое реляционная база данных?
3. Что такое SQL?
4. Перечислите основные команды SQL.

5. Для чего используется команда SELECT?
6. Для чего используется команда WHERE?
7. Для чего используется команда GROUP BY?
8. Что такое SQLite?
9. Как подключиться к базе данных SQLite из Python?
10. Как создать таблицу в SQLite?
11. Как добавить, изменить и удалить данные в таблице SQLite?

Отчет должен содержать:

Титульный лист (название работы, ФИО, группа). Цель работы. Краткое описание выполненных заданий. Описание реляционных баз данных и языка SQL. Код Python, используемый для создания базы данных, таблиц и выполнения запросов. Результаты выполнения запросов (вывод данных). Описание выполненных операций с базой данных. Ответы на контрольные вопросы. Выводы о проделанной работе.

Лабораторная работа 4

Тема: **NoSQL базы данных.**

Цель работы: Ознакомиться с концепцией NoSQL баз данных, их преимуществами и типами. Приобрести практические навыки работы с MongoDB: создание коллекций, добавление и поиск данных.

Задание 1. Установка MongoDB:

Скачайте и установите MongoDB с официального сайта: https://www.mongodb.com/try/download/community

Установите MongoDBCompass (GUI для управления MongoDB).

Запустите MongoDB сервер.

Задание 2. Подключение к MongoDB из Python:

Установите библиотеку pymongo:

pipinstallpymongo

Импортируйте библиотеку pymongo в Python:

importpymongo

Подключитесь к MongoDB серверу:

client = pymongo.MongoClient("mongodb://localhost:27017/")

Задание 3. Создание базы данных и коллекции:

Создайте новую базу данных с именем mydatabase:

db = client["mydatabase"]

Создайте новую коллекцию с именем students:

collection = db["students"]

Задание 4. Добавление данных в коллекцию:

Добавьте несколько документов в коллекцию students:

student1 = {"name": "Alice", "age": 20, "city": "New York"}

student2 = {"name": "Bob", "age": 22, "city": "London"}

student3 = {"name": "Charlie", "age": 21, "city": "Paris"}

student4 = {"name": "David", "age": 23, "city": "New York"}

collection.insert_many([student1, student2, student3, student4])

Задание 5. Поиск данных в коллекции:

Найдите все документы в коллекции students:

```
for student in collection.find():
    print(student)
```

Найдите студентов из города "NewYork":

```
for student in collection.find({"city": "New York"}):
    print(student)
```

Найдите студентов старше 21 года:

```
for student in collection.find({"age": {"$gt": 21}}): # $gt - greater than
    print(student)
```

Найдите студентов старше 21 года и из города "NewYork":

```
for student in collection.find({"age": {"$gt": 21}, "city": "New York"}):
    print(student)
```

Задание 6. Обновление данных в коллекции:

Обновите возраст студента с именем "Alice" на 25:

```
collection.update_one({"name": "Alice"}, {"$set": {"age": 25}})
```

Проверьте изменения:

```
for student in collection.find({"name": "Alice"}):
    print(student)
```

Задание 7. Удаление данных из коллекции:

Удалите студента с именем "Charlie":

```
collection.delete_one({"name": "Charlie"})
```

Проверьте изменения:

```
for student in collection.find():
    print(student)
```

Закрытие соединения с MongoDB:

```
client.close()
```

Контрольные вопросы:

1. Что такое NoSQL база данных?
2. Перечислите преимущества NoSQL баз данных.
3. Какие типы NoSQL баз данных вы знаете?
4. Что такое MongoDB?
5. Какие форматы данных используются в MongoDB?
6. Как подключиться к MongoDB из Python?
7. Как создать базу данных и коллекцию в MongoDB?
8. Как добавить, найти, обновить и удалить данные в коллекции MongoDB?

Отчет должен содержать:

Титульный лист (название работы, ФИО, группа). Цель работы. Краткое описание выполненных заданий. Описание NoSQL баз данных и MongoDB. Код Python, используемый для подключения к MongoDB, создания базы данных и коллекции, добавления, поиска, обновления и удаления данных. Результаты выполнения кода (вывод данных).Описание выполненных операций с базой данных. Ответы на контрольные вопросы. Выводы о проделанной работе.

Лабораторная работа 5

Тема: **Введение в Hadoop**.

Цель работы: Ознакомиться с Hadoop как одной из ключевых технологий для работы с большими данными. Изучить архитектуру HDFS (Hadoop Distributed File System) и принципы распределенного хранения данных. Приобрести практические навыки работы с Hadoop и загрузки данных в HDFS.

Задание 1. Установка Hadoop (SingleNodeSetup):

Скачайте и установите Hadoop с официального сайта Apache: https://hadoop.apache.org/releases.html

Следуйте инструкциям по установке Hadoop в режиме SingleNodeSetup (StandaloneMode или Pseudo-DistributedMode).

Например, можно использовать руководство с сайта Apache: https://hadoop.apache.org/docs/stable/hadoop-project-dist/single-node-setup.html

Настройте переменные окружения (JAVA_HOME, HADOOP_HOME, PATH).

Отредактируйте конфигурационные файлы (hadoop-env.sh, core-site.xml, hdfs-site.xml, mapred-site.xml, yarn-site.xml).

Форматируйте HDFS:

hdfsnamenode -format

Запустите Hadoop:

start-dfs.sh

start-yarn.sh

Задание 2. Проверка работоспособности Hadoop:

Откройте веб-интерфейс NameNode (по умолчанию http://localhost:9870) и убедитесь, что HDFS запущен.

Откройте веб-интерфейс ResourceManager (по умолчанию http://localhost:8088) и убедитесь, что YARN запущен.

Задание 3. Создание директории в HDFS:

Создайте директорию /user/<username> в HDFS:

hadoop fs -mkdir /user/<username>

hadoop fs -mkdir /user/<username>/input

Замените <username> на имя вашего пользователя в системе.

Задание 4. Загрузка данных в HDFS:

Создайте текстовый файл с данными (например, data.txt).

Загрузите файл data.txt в директорию /user/<username>/input в HDFS:

hadoop fs -put data.txt /user/<username>/input

Задание 5. Просмотр содержимого HDFS:

Просмотрите список файлов и каталогов в HDFS:

hadoop fs -ls /user/<username>/input

Просмотрите содержимое файла data.txt в HDFS:

hadoop fs -cat /user/<username>/input/data.txt

Задание 6. Анализ репликации данных (опционально):

Изучите веб-интерфейс NameNode и посмотрите информацию о блоках данных и их репликах.

Попробуйте остановить один из DataNode и убедитесь, что данные все еще доступны благодаря репликации.

Задание 7. Остановка Hadoop:

Остановите Hadoop:

stop-dfs.sh

stop-yarn.sh

Контрольные вопросы:

1. Что такое Hadoop?
2. Какие основные компоненты Hadoop вы знаете?
3. Что такое HDFS?
4. Какова архитектура HDFS?
5. Что такое NameNode и DataNode?
6. Как данные хранятся в HDFS?
7. Что такое репликация данных и зачем она нужна?

8. Как создать директорию в HDFS?
9. Как загрузить данные в HDFS?
10. Как просмотреть содержимое HDFS?

Отчет должен содержать:

Титульный лист (название работы, ФИО, группа). Цель работы. Краткое описание выполненных заданий. Описание Hadoop и HDFS. Инструкции по установке и настройке Hadoop. Команды Hadoop, используемые для создания директории, загрузки данных и просмотра содержимого HDFS. Скриншоты, иллюстрирующие работу с Hadoop и веб-интерфейсом NameNode. Ответы на контрольные вопросы. Выводы о проделанной работе.

Лабораторная работа 6

Тема: **Основы MapReduce.**

Цель работы: Изучить принцип работы MapReduce и его применения для обработки больших объемов данных. Приобрести практические навыки написания и запуска простого MapReduce-задания (например, подсчета слов в тексте) на платформе Hadoop.

Задание 1. Настройка Hadoop (если еще не настроен):

Убедитесь, что Hadoop установлен и настроен в режиме SingleNodeSetup (StandaloneMode или Pseudo-DistributedMode). См. Лабораторную работу №5.

Запустите Hadoop:

start-dfs.sh

start-yarn.sh

Задание 2. Написание MapReduce-задания (WordCount) на Java:

Создайте Java-проект в IDE (например, IntelliJ IDEA или Eclipse).

Добавьтезависимости Hadoop MapReduce впроект (Hadoop Common, Hadoop Client, Hadoop MapReduce Client Core).

Создайте три класса:

WordCountMapper: Класс, реализующий функцию Map.

WordCountReducer: Класс, реализующий функцию Reduce.

WordCountDriver: Класс, содержащий метод main, который запускает MapReduce-задание.

```
import java.io.IOException;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
```

```
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class WordCountDriver {

   public static void main(String[] args) throws Exception {
      Configuration conf = new Configuration();
      Job job = Job.getInstance(conf, "word count");
job.setJarByClass(WordCountDriver.class);
job.setMapperClass(WordCountMapper.class);
job.setReducerClass(WordCountReducer.class);
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(IntWritable.class);
FileInputFormat.addInputPath(job, new Path(args[0]));
FileOutputFormat.setOutputPath(job, new Path(args[1]));
System.exit(job.waitForCompletion(true) ? 0 : 1);
   }
}

public class WordCountMapper extends Mapper<Object, Text, Text,
IntWritable>{

   private final static IntWritable one = new IntWritable(1);
   private Text word = new Text();

   public void map(Object key, Text value, Context context) throws IOExcep-
tion, InterruptedException {
      String[] words = value.toString().split("\\s+"); // Split by whitespace
      for (String w : words) {
```

```
word.set(w.toLowerCase().replaceAll("[^a-zA-Z0-9]", "")); // Convert to lowercase and remove non-alphanumeric characters
            if (!word.toString().isEmpty()) {
context.write(word, one);
            }
        }
    }
}

public class WordCountReducer extends Reducer<Text,IntWritable,Text,IntWritable> {
    private IntWritable result = new IntWritable();

    public void reduce(Text key, Iterable<IntWritable> values, Context context) throws IOException, InterruptedException {
        int sum = 0;
        for (IntWritableval : values) {
            sum += val.get();
        }
result.set(sum);
context.write(key, result);
    }
}
```

<u>Задание 3. Создание входного файла. Загрузка входного файла в HDFS. Экспорт проекта в JAR-файл.</u>

Создайте текстовый файл input.txt с текстом, который нужно проанализировать.

```
hadoop fs -mkdir /user/<username>/wordcount_input
hadoop fs -put input.txt /user/<username>/wordcount_input
```

Экспортируйте Java-проект в JAR-файл.

Задание4. Запуск MapReduce-задания:

hadoop jar <jar_file_name>.jar WordCountDriver /user/<username>/wordcount_input /user/<username>/wordcount_output

Просмотрите результаты работы MapReduce-задания в HDFS:

hadoop fs -cat /user/<username>/wordcount_output/part-r-00000

Остановка Hadoop:

stop-dfs.sh

stop-yarn.sh

Контрольные вопросы:

1. Что такое MapReduce?
2. Какие основные фазы MapReduce вы знаете?
3. Опишите принцип работы MapReduce.
4. Что такое Mapper и Reducer?
5. Какова роль функции Map?
6. Какова роль функции Reduce?
7. Как запустить MapReduce-задание на Hadoop?
8. Как просмотреть результаты MapReduce-задания?

Отчет должен содержать:

Титульный лист (название работы, ФИО, группа). Цель работы. Краткое описание выполненных заданий. Описание MapReduce. Код Java, используемый для реализации MapReduce-задания (WordCountMapper, WordCountReducer, WordCountDriver). Входной файл (input.txt). Команды Hadoop, используемые для запуска MapReduce-задания и просмотра результатов. Результаты работы MapReduce-задания (вывод данных). Ответы на контрольные вопросы. Выводы о проделанной работе.

Лабораторная работа 7

Тема: **Введение в ApacheSpark**.

Цель работы: Ознакомиться с ApacheSpark как технологией для быстрой обработки данных. Изучить основные компоненты Spark (RDD, DataFrame) и их применение для выполнения операций с данными. Приобрести практические навыки работы со Spark для фильтрации и группировки данных.

Задание 1. Установка Spark:

Скачайте и установите ApacheSpark с официального сайта: https://spark.apache.org/downloads.htmlУстановите Java (JDK 8 или выше).Установите Python (Python 3.6 или выше).Настройте переменные окружения (SPARK_HOME, JAVA_HOME, PATH).

Задание 2. Запуск SparkShell. Работа с RDD:

Запустите SparkShell (интерактивная оболочка Spark):

```
./bin/spark-shell
```

Создайте RDD из списка чисел:

```
val data = Array(1, 2, 3, 4, 5)
valrdd = spark.sparkContext.parallelize(data)
```

Отфильтруйте RDD, чтобы выбрать только четные числа:

```
valevenNumbers = rdd.filter(x => x % 2 == 0)
```

Выведите содержимое отфильтрованного RDD:

```
evenNumbers.foreach(println)
```

Преобразуйте RDD, чтобы получить квадрат каждого числа:

```
valsquaredNumbers = rdd.map(x => x * x)
```

Выведите содержимое преобразованного RDD:

```
squaredNumbers.foreach(println)
```

Сосчитайте сумму чисел в RDD:

```
val sum = rdd.reduce((x, y) => x + y)
println(s"Sum: ${sum}")
```

Задание 3Работа с DataFrame:

Создайте DataFrame из списка словарей:

```
from pyspark.sql import SparkSession

spark = SparkSession.builder.appName("DataFrameExample").getOrCreate()

data = [
    {"name": "Alice", "age": 25, "city": "New York"},
    {"name": "Bob", "age": 30, "city": "London"},
    {"name": "Charlie", "age": 28, "city": "Paris"},
    {"name": "David", "age": 23, "city": "New York"}
]

df = spark.createDataFrame(data)
df.show()
```

Выведите схему DataFrame:

```
df.printSchema()
```

Отфильтруйте DataFrame, чтобы выбрать только строки, где city = "NewYork":

```
filtered_df = df.filter(df.city == "New York")
filtered_df.show()
```

Сгруппируйте DataFrame по городу и рассчитайте средний возраст для каждой группы:

```
from pyspark.sql.functions import avg

grouped_df = df.groupBy("city").agg(avg("age").alias("average_age"))
grouped_df.show()
```

Сохраните DataFrame в формате CSV:

grouped_df.write.csv("output.csv", header=True)

Выйдите из Spark Shell:

:quit

Контрольные вопросы:

1. Что такое ApacheSpark?
2. Какие основные компоненты Spark вы знаете?
3. Что такое RDD?
4. Что такое DataFrame?
5. В чем разница между RDD и DataFrame?
6. Как создать RDD и DataFrame?
7. Как отфильтровать и сгруппировать данные в Spark?

Отчет должен содержать:

Титульный лист (название работы, ФИО, группа). Цель работы. Краткое описание выполненных заданий. Описание ApacheSpark, RDD и DataFrame. Код, используемый для работы с RDD и DataFrame (на Scala или Python). Результаты выполнения кода (вывод данных).Описание выполненных операций с данными. Ответы на контрольные вопросы. Выводы о проделанной работе.

Лабораторная работа 8

Тема: **Анализ данных с помощью Pandas.**

Цель работы: Углубить знания по работе с библиотекой Pandas для анализа данных в Python. Изучить основные функции Pandas (фильтрация, группировка, агрегация) и их применение для работы с реальными наборами данных. Приобрести практические навыки создания простых графиков с использованием библиотеки Matplotlib.

Задание1.Установка библиотек. Чтение данных из CSV-файла. Преобразование типов данных.

Установите библиотеки Pandas и Matplotlib с помощью pip:

pipinstallpandasmatplotlib

Скачайте CSV-файл с данными (например, с Kaggle: https://www.kaggle.com/datasets). Можно использовать набор данных "LondonBikeSharingDataset".

Импортируйте библиотеки Pandas и Matplotlib в Python:

```
import pandas as pd
import matplotlib.pyplot as plt
```

Прочитайте данные из CSV-файла в DataFrame:

```
df = pd.read_csv("london_merged.csv")
```

Выведите первые несколько строк DataFrame:

```
print(df.head())
```

Выведите информацию о DataFrame (типы данных, количество строк и столбцов):

```
print(df.info())
```

Преобразуйте столбец timestamp в формат datetime:

```
df['timestamp'] = pd.to_datetime(df['timestamp'])
```

Создайте новые столбцы на основе столбца timestamp (год, месяц, день недели, час):

```
df['year'] = df['timestamp'].dt.year
df['month'] = df['timestamp'].dt.month
```

df['day_of_week'] = df['timestamp'].dt.day_name()

df['hour'] = df['timestamp'].dt.hour

Задание 2. Фильтрация данных. Группировка данных. Агрегация данных.

Отфильтруйте DataFrame, чтобы выбрать только данные за определенный год (например, 2015):

df_2015 = df[df['year'] == 2015]

print(df_2015.head())

Сгруппируйте DataFrame по дню недели и рассчитайте среднее количество велосипедных поездок (cnt) для каждого дня недели:

df_grouped = df.groupby('day_of_week')['cnt'].mean()

print(df_grouped)

Рассчитайте основные статистические показатели для столбца cnt (среднее значение, медиана, минимум, максимум, стандартное отклонение):

print(df['cnt'].describe())

Задание 3. Визуализация данных с помощью Matplotlib:

Постройте столбчатую диаграмму для отображения среднего количества велосипедных поездок по дням недели:

plt.figure(figsize=(10, 6))

plt.bar(df_grouped.index, df_grouped.values)

plt.xlabel("Day of Week")

plt.ylabel("Average Bike Rides")

plt.title("Average Bike Rides per Day of Week")

plt.xticks(rotation=45)

plt.tight_layout()

plt.show()

Постройте линейный график для отображения количества велосипедных поездок по часам в течение дня:

df_hourly = df.groupby('hour')['cnt'].mean()

plt.figure(figsize=(10, 6))

plt.plot(df_hourly.index, df_hourly.values)

plt.xlabel("Hour of Day")

plt.ylabel("Average Bike Rides")

plt.title("Average Bike Rides per Hour of Day")

plt.grid(True)

plt.show()

Постройте гистограмму для отображения распределения количества велосипедных поездок:

plt.figure(figsize=(10, 6))

plt.hist(df['cnt'], bins=30)

plt.xlabel("Bike Rides")

plt.ylabel("Frequency")

plt.title("Distribution of Bike Rides")

plt.show()

Контрольные вопросы:

1. Что такое Pandas?
2. Что такое DataFrame?
3. Как прочитать данные из CSV-файла в DataFrame?
4. Как отфильтровать данные в Pandas?
5. Как сгруппировать данные в Pandas?
6. Как рассчитать статистические показатели в Pandas?
7. Что такое Matplotlib?
8. Как построить столбчатую диаграмму, линейный график и гистограмму с использованием Matplotlib?

Отчет должен содержать:

Титульный лист (название работы, ФИО, группа). Цель работы. Краткое описание выполненных заданий. Описание Pandas и Matplotlib. Код

Python, используемый для чтения, фильтрации, группировки и агрегации данных, а также для создания графиков. Графики, созданные с использованием Matplotlib. Описание выполненных операций с данными и интерпретация полученных результатов. Ответы на контрольные вопросы. Выводы о проделанной работе.

Учебно-методическое дисциплины

Перечень основной литературы:

1. МакКинни У. Python для анализа данных / У. МакКинни. – 2-е изд. – СПб.: Питер, 2020. – 544 с. – ISBN 978-5-4461-1345-3.
2. Теейт Р. М. П. SQL для Data Scientists: Руководство для начинающих по созданию наборов данных для анализа / Р. М. П. Теейт. – М.: ДМК Пресс, 2021. – 280 с. – ISBN 978-5-97060-912-7.
3. Брэдшоу Ш. Mongo DB: Полное руководство / Ш. Брэдшоу, Э. Бразил, К. Чодоров. – М.: Вильямс, 2020. – 720 с. – ISBN 978-5-9909445-8-7.
4. Уайт Т. Hadoop: Подробное руководство / Т. Уайт. – 4-е изд. – СПб.: Питер, 2019. – 768 с. – ISBN 978-5-4461-1157-2.
5. Дамджи Дж. Изучаем Spark / Дж. Дамджи, Б. Уэниг, Т. Дас, Д. Ли. – М.: ДМК Пресс, 2021. – 480 с. – ISBN 978-5-97060-899-1.

Перечень дополнительной литературы:

1. Парк Э. Data Science для начинающих / Э. Парк. – М.: ДМК Пресс, 2020. – 320 с. – ISBN 978-5-97060-835-9.
2. Марц Н. Большие данные: Принципы и практика построения масштабируемых систем обработки данных в реальном времени / Н. Марц, Дж. Уоррен. – М.: Вильямс, 2015. – 480 с. – ISBN 978-5-8459-1992-2.
3. Клеппман М. Высоконагруженные приложения: Программирование, масштабирование, поддержка / М. Клеппман. – М.: Питер, 2019. – 592 с. – ISBN 978-5-4461-1108-4.
4. Чарльз Д. Искусство обработки данных: Как извлечь знания из больших данных / Д. Чарльз. – М.: ДМК Пресс, 2018. – 360 с. – ISBN 978-5-97060-567-9.
5. Рамачандра К. Основы Data Science и Big Data: Python и наука о данных / К. Рамачандра. – М.: Вильямс, 2020. – 400 с. – ISBN 978-5-9909445-7-0.
6. Майер-Шёнбергер В. Большие данные: Революция, которая изменит то, как мы живем, работаем и мыслим / В. Майер-Шёнбергер, К. Кукьер. – М.: Манн, Иванов и Фербер, 2014. – 240 с. – ISBN 978-5-91657-947-9.

Перечень ресурсов информационно-телекоммуникационной сети «Интернет», необходимых для освоения дисциплины

1. https://hadoop.apache.org/docs/current/ - документация Hadoop.
2. https://spark.apache.org/docs/latest/ - документация Apache Spark.
3. https://docs.mongodb.com/ - документация MongoDB.
4. https://pandas.pydata.org/docs/ - документация Pandas.
5. https://scikit-learn.org/stable/ - документация Scikit-learn.
6. https://www.kaggle.com/ - Платформа для работы с реальными наборами данных и участия в соревнованиях по Data Science.
7. https://datasetsearch.research.google.com/ - Поиск открытых наборов данных для анализа.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Методические указания
по организации и проведению самостоятельной работы
по дисциплине
«БОЛЬШИЕ ДАННЫЕ»
для студентов направления подготовки
02.03.01 Математика и компьютерные науки
направленность (профиль)
«Анализ данных и искусственный интеллект»

Общие положения

Самостоятельная работа - планируемая учебная, учебно-исследовательская, научно-исследовательская работа студентов, выполняемая во внеаудиторное (аудиторное) время по заданию и при методическом руководстве преподавателя, но без его непосредственного участия (при частичном непосредственном участии преподавателя, оставляющем ведущую роль за работой студентов).

Самостоятельная работа студентов (СРС) в ВУЗе является важным видом учебной и научной деятельности студента. Самостоятельная работа студентов играет значительную роль в рейтинговой технологии обучения.

К основным видам самостоятельной работы студентов относятся:

– формирование и усвоение содержания конспекта лекций на базе рекомендованной лектором учебной литературы, включая информационные образовательные ресурсы (электронные учебники, электронные библиотеки и др.);

– написание докладов;

– подготовка к семинарам, практическим и лабораторным работам, их оформление;

– составление аннотированного списка статей из соответствующих журналов по отраслям знаний (педагогических, психологических, методических и др.);

– выполнение учебно-исследовательских работ, проектная деятельность;

– подготовка практических разработок и рекомендаций по решению проблемной ситуации;

– выполнение домашних заданий в виде решения отдельных задач, проведения типовых расчетов, расчетно-компьютерных и индивидуальных работ по отдельным разделам содержания дисциплин и т.д.;

– компьютерный текущий самоконтроль и контроль успеваемости на базе электронных обучающих и аттестующих тестов;

– выполнение курсовых работ (проектов) в рамках дисциплин;

– выполнение выпускной квалификационной работы и др.

Методика организации самостоятельной работы студентов зависит от структуры, характера и особенностей изучаемой дисциплины, объема часов на ее изучение, вида заданий для самостоятельной работы студентов, индивидуальных качеств студентов и условий учебной деятельности.

Процесс организации самостоятельной работы студентов включает в себя следующие этапы:

- подготовительный (определение целей, составление программы, подготовка методического обеспечения, подготовка оборудования);
- основной (реализация программы, использование приемов поиска информации, усвоения, переработки, применения, передачи знаний, фиксирование результатов, самоорганизация процесса работы);
- заключительный (оценка значимости и анализ результатов, их систематизация, оценка эффективности программы и приемов работы, выводы о направлениях оптимизации труда).

Самостоятельная работа по дисциплине **«Большие данные»** направлена на формирование следующих **компетенций**:

- **ПК-1.** Способен демонстрировать базовые знания математических и естественных наук, основ программирования и информационных технологий
- **ПК-7.** Способен подготавливать данные и проводить аналитические работы по исследованию больших данных

1. Цель и задачи самостоятельной работы

Ведущая цель организации и осуществления СРС совпадает с целью обучения студента – формирование соответствующего набора компетенций будущего бакалавра по направлению подготовки «Математика и компьютерные науки».

При организации СРС важным и необходимым условием становятся формирование умения самостоятельной работы для приобретения знаний, навыков и возможности организации учебной и научной деятельности. Целью самостоятельной работы студентов является овладение фундаментальными знаниями, профессиональными умениями и навыками деятельности по профилю, опытом творческой, исследовательской деятельности. Самостоятельная работа студентов способствует развитию самостоятельности, ответственности и организованности, творческого подхода к решению проблем учебного и профессионального уровня.

Задачами СРС являются:

- систематизация и закрепление полученных теоретических знаний и практических умений студентов;
- углубление и расширение теоретических знаний;
- формирование умений использовать нормативную, правовую, справочную документацию и специальную литературу;
- развитие познавательных способностей и активности студентов: творческой инициативы, самостоятельности, ответственности и организованности;
- формирование самостоятельности мышления, способностей к саморазвитию, самосовершенствованию и самореализации;
- развитие исследовательских умений;
- использование материала, собранного и полученного в ходе самостоятельных занятий на семинарах, на практических и лабораторных занятиях, при написании курсовых и выпускной квалификационной работ, для эффективной подготовки к итоговым зачетам и экзаменам.

2. Технологическая карта самостоятельной работы студента

Коды реализуемых компетенций	Вид деятельности студентов	Средства и технологии оценки	Объем часов, в том числе (акад.)		
			СРС	Контактная работа с преподавателем	Всего
6 семестр					
ПК-1, ПК-7	Самостоятельное изучение литературы	Собеседование Написание реферата	38	2	40
ПК-1, ПК-7	Подготовка к лабораторным работам	Отчет	18	2	20
Итого за 6 семестр			**56**	**4**	**60**
Итого			**56**	**4**	**60**

3. Порядок выполнения самостоятельной работы студентом

3.1. Методические рекомендации по работе с учебной литературой

При работе с книгой необходимо подобрать литературу, научиться правильно ее читать, вести записи. Для подбора литературы в библиотеке используются алфавитный и систематический каталоги.

Важно помнить, что рациональные навыки работы с книгой - это всегда большая экономия времени и сил.

Правильный подбор учебников рекомендуется преподавателем, читающим лекционный курс. Необходимая литература может быть также указана в методических разработках по данному курсу.

Изучая материал по учебнику, следует переходить к следующему вопросу только после правильного уяснения предыдущего, описывая на бумаге все выкладки и вычисления (в том числе те, которые в учебнике опущены или на лекции даны для самостоятельного вывода).

При изучении любой дисциплины большую и важную роль играет самостоятельная индивидуальная работа.

Особое внимание следует обратить на определение основных понятий курса. Студент должен подробно разбирать примеры, которые поясняют такие определения, и уметь строить аналогичные примеры самостоятельно. Нужно добиваться точного представления о том, что изучаешь. Полезно составлять опорные конспекты. При изучении материала по учебнику полезно в тетради (на специально отведенных полях) дополнять конспект лекций. Там же следует отмечать вопросы, выделенные студентом для консультации с преподавателем.

Выводы, полученные в результате изучения, рекомендуется в конспекте выделять, чтобы они при перечитывании записей лучше запоминались.

Опыт показывает, что многим студентам помогает составление листа опорных сигналов, содержащего важнейшие и наиболее часто употребляемые формулы и понятия. Такой лист помогает запомнить формулы, основные положения лекции, а также может служить постоянным справочником для студента.

Чтение научного текста является частью познавательной деятельности. Ее цель – извлечение из текста необходимой информации. От того на сколько осознанна читающим собственная внутренняя установка при обращении к печатному слову (найти нужные сведения, усвоить информацию полностью или частично, критически проанализировать материал и т.п.) во многом зависит эффективность осуществляемого действия.

Выделяют ***четыре основные установки в чтении научного текста***:

информационно-поисковый (задача – найти, выделить искомую информацию)

усваивающая (усилия читателя направлены на то, чтобы как можно полнее осознать и запомнить как сами сведения излагаемые автором, так и всю логику его рассуждений)

аналитико-критическая (читатель стремится критически осмыслить материал, проанализировав его, определив свое отношение к нему)

творческая (создает у читателя готовность в том или ином виде – как отправной пункт для своих рассуждений, как образ для действия по аналогии и т.п. – использовать суждения автора, ход его мыслей, результат наблюдения, разработанную методику, дополнить их, подвергнуть новой проверке).

Основные виды систематизированной записи прочитанного:

Аннотирование – предельно краткое связное описание просмотренной или прочитанной книги (статьи), ее содержания, источников, характера и назначения;

Планирование – краткая логическая организация текста, раскрывающая содержание и структуру изучаемого материала;

Тезирование – лаконичное воспроизведение основных утверждений автора без привлечения фактического материала;

Цитирование – дословное выписывание из текста выдержек, извлечений, наиболее существенно отражающих ту или иную мысль автора;

Конспектирование – краткое и последовательное изложение содержания прочитанного.

Конспект – сложный способ изложения содержания книги или статьи в логической последовательности. Конспект аккумулирует в себе предыдущие виды записи, позволяет всесторонне охватить содержание книги, статьи. Поэтому умение составлять план, тезисы, делать выписки и другие записи определяет и технологию составления конспекта.

Методические рекомендации по составлению конспекта:

1. Внимательно прочитайте текст. Уточните в справочной литературе непонятные слова. При записи не забудьте вынести справочные данные на поля конспекта;
2. Выделите главное, составьте план;
3. Кратко сформулируйте основные положения текста, отметьте аргументацию автора;
4. Законспектируйте материал, четко следуя пунктам плана. При конспектировании старайтесь выразить мысль своими словами. Записи следует вести четко, ясно.
5. Грамотно записывайте цитаты. Цитируя, учитывайте лаконичность, значимость мысли.

В тексте конспекта желательно приводить не только тезисные положения, но и их доказательства. При оформлении конспекта необходимо стремиться к емкости каждого предложения. Мысли автора книги следует излагать кратко, заботясь о стиле и выразительности написанного. Число дополнительных элементов конспекта должно быть логически обоснованным, записи должны распределяться в определенной последовательности, отвечающей логической структуре произведения. Для уточнения и дополнения необходимо оставлять поля.

Овладение навыками конспектирования требует от студента целеустремленности, повседневной самостоятельной работы.

3.2. Методические рекомендации по подготовке к практическим и лабораторным занятиям

Для того чтобы практические и лабораторные занятия приносили максимальную пользу, необходимо помнить, что упражнение и решение задач проводятся по вычитанному на лекциях материалу и связаны, как правило, с детальным разбором отдельных вопросов лекционного курса. Следует подчеркнуть, что только после усвоения лекционного материала с определенной точки зрения (а именно с той, с которой он излагается на лекциях) он будет закрепляться на практических занятиях как в результате обсуждения и анализа лекционного материала, так и с помощью решения проблемных ситуаций, задач. При этих условиях студент не только хорошо усвоит материал, но и научится применять его на практике, а также получит дополнительный стимул (и это очень важно) для активной проработки лекции.

При самостоятельном решении задач нужно обосновывать каждый этап решения, исходя из теоретических положений курса. Если студент видит несколько путей решения проблемы (задачи), то нужно сравнить их и выбрать самый рациональный. Полезно до начала вычислений составить краткий план решения проблемы (задачи). Решение проблемных задач или примеров следует излагать подробно, вычисления располагать в строгом порядке, отделяя вспомогательные вычисления от основных. Решения при необходимости нужно сопровождать комментариями, схемами, чертежами и рисунками.

Следует помнить, что решение каждой учебной задачи должно доводиться до окончательного логического ответа, которого требует условие, и по возможности с выво-

дом. Полученный ответ следует проверить способами, вытекающими из существа данной задачи. Полезно также (если возможно) решать несколькими способами и сравнить полученные результаты. Решение задач данного типа нужно продолжать до приобретения твердых навыков в их решении.

3.3. Методические рекомендации по самопроверке знаний

После изучения определенной темы по записям в конспекте и учебнику, а также решения достаточного количества соответствующих задач на практических занятиях и самостоятельно студенту рекомендуется, провести самопроверку усвоенных знаний, ответив на контрольные вопросы по изученной теме.

В случае необходимости нужно еще раз внимательно разобраться в материале.

Иногда недостаточность усвоения того или иного вопроса выясняется только при изучении дальнейшего материала. В этом случае надо вернуться назад и повторить плохо усвоенный материал. Важный критерий усвоения теоретического материала - умение решать задачи или пройти тестирование по пройденному материалу. Однако следует помнить, что правильное решение задачи может получиться в результате применения механически заученных формул без понимания сущности теоретических положений.

3.4. Методические рекомендации по написанию научных текстов (докладов, рефератов, эссе, научных статей и т.д.)

Перед тем, как приступить к написанию научного текста, важно разобраться, какова истинная цель вашего научного текста - это поможет вам разумно распределить свои силы и время.

Во-первых, сначала нужно определиться с идеей научного текста, а для этого необходимо научиться либо относиться к разным явлениям и фактам несколько критически (своя идея – как иная точка зрения), либо научиться увлекаться какими-то известными идеями, которые нуждаются в доработке (идея – как оптимистическая позиция и направленность на дальнейшее совершенствование уже известного). Во-вторых, научиться организовывать свое время, ведь, как известно, свободное (от всяких глупостей) время – важнейшее условие настоящего творчества, для него наконец-то появляется время. Иногда именно на организацию такого времени уходит немалая часть сил и талантов.

Писать следует ясно и понятно, стараясь основные положения формулировать четко и недвусмысленно (чтобы и самому понятно было), а также стремясь структурировать свой текст. Каждый раз надо представлять, что ваш текст будет кто-то читать и ему захочется сориентироваться в нем, быстро находить ответы на интересующие вопросы (заодно представьте себя на месте такого человека). Понятно, что работа, написанная «сплошным текстом» (без заголовков, без выделения крупным шрифтом наиболее важным мест и т, п.), у культурного читателя должна вызывать брезгливость и даже жалость к автору (исключения составляют некоторые древние тексты, когда и жанр был иной и к текстам относились иначе, да и самих текстов было гораздо меньше – не то, что в эпоху «информационного взрыва» и соответствующего «информационного мусора»).

Объем текста и различные оформительские требования во многом зависят от принятых в конкретном учебном заведении порядков.

Реферат (доклад) - это самостоятельное исследование студентом определенной проблемы, комплекса взаимосвязанных вопросов.

Реферат не должна составляться из фрагментов статей, монографий, пособий. Кроме простого изложения фактов и цитат, в реферате должно проявляться авторское видение проблемы и ее решения.

Рассмотрим основные этапы подготовки реферата студентом.

Выполнение реферата начинается с выбора темы.

Затем студент приходит на первую консультацию к руководителю, которая предусматривает:

- обсуждение цели и задач работы, основных моментов избранной темы;
- консультирование по вопросам подбора литературы;
- составление предварительного плана.

Следующим этапом является работа с литературой. Необходимая литература подбирается студентом самостоятельно.

После подбора литературы целесообразно сделать рабочий вариант плана работы. В нем нужно выделить основные вопросы темы и параграфы, раскрывающие их содержание.

Составленный список литературы и предварительный вариант плана уточняются, согласуются на очередной консультации с руководителем.

Затем начинается следующий этап работы - изучение литературы. Только внимательно читая и конспектируя литературу, можно разобраться в основных вопросах темы и подготовиться к самостоятельному (авторскому) изложению содержания реферата. Конспектируя первоисточники, необходимо отразить основную идею автора и его позицию по исследуемому вопросу, выявить проблемы и наметить задачи для дальнейшего изучения данных проблем.

Систематизация и анализ изученной литературы по проблеме исследования позволяют студенту написать работу.

Рабочий вариант текста реферата предоставляется руководителю на проверку. На основе рабочего варианта текста руководитель вместе со студентом обсуждает возможности доработки текста, его оформление. После доработки реферат сдается на кафедру для его оценивания руководителем.

Требования к написанию реферата

Написание 1 реферата является обязательным условием выполнения плана СРС по любой дисциплине профессионального цикла.

Тема реферата может быть выбрана студентом из предложенных в рабочей программе или фонде оценочных средств дисциплины, либо определена самостоятельно, исходя из интересов студента (в рамках изучаемой дисциплины). Выбранную тему необходимо согласоваться с преподавателем.

Реферат должен быть написан научным языком.

Объем реферата должен составлять 20-25 стр.

Структура реферата:

- Введение (не более 3-4 страниц). Во введении необходимо обосновать выбор темы, ее актуальность, очертить область исследования, объект исследования, основные цели и задачи исследования.
- Основная часть состоит из 2-3 разделов. В них раскрывается суть исследуемой проблемы, проводится обзор мировой литературы и источников Интернет по предмету исследования, в котором дается характеристика степени разработанности проблемы и авторская аналитическая оценка основных теоретических подходов к ее решению. Изложение материала не должно ограничиваться лишь описательным подходом к раскрытию выбранной темы. Оно также должно содержать собственное видение рассматриваемой проблемы и изложение собственной точки зрения на возможные пути ее решения.
- Заключение (1-2 страницы). В заключении кратко излагаются достигнутые при изучении проблемы цели, перспективы развития исследуемого вопроса
- Список использованной литературы (не меньше 10 источников), в алфавитном порядке, оформленный в соответствии с принятыми правилами. В список использованной литературы рекомендуется включать работы отечественных и зарубежных авторов, в том числе статьи, опубликованные в научных журналах в течение последних 3-х лет и ссылки на ресурсы сети Интернет.

- Приложение (при необходимости).

Требования к оформлению:

- текст с одной стороны листа;
- шрифт Times New Roman;
- кегль шрифта 14;
- межстрочное расстояние 1,5;
- поля: сверху 2,5 см, снизу – 2,5 см, слева - 3 см, справа 1,5 см;
- реферат должен быть представлен в сброшюрованном виде.

Порядок защиты реферата:

Защита реферата проводится на практических занятиях, после окончания работы студента над ним и исправления всех недочетов, выявленных преподавателем в ходе консультаций. На защиту реферата отводится 5-7 минут времени, в ходе которого студент должен показать свободное владение материалом по заявленной теме. При защите реферата приветствуется использование мультимедиа-презентации.

Оценка реферата

Реферат оценивается по следующим критериям:

• соблюдение требований к его оформлению;

• необходимость и достаточность для раскрытия темы приведенной в тексте реферата информации;

• умение студента свободно излагать основные идеи, отраженные в реферате;

• способность студента понять суть задаваемых преподавателем и сокурсниками вопросов и сформулировать точные ответы на них.

Критерии оценки:

Оценка «отлично» выставляется студенту, если в докладе студент исчерпывающе, последовательно, четко и логически стройно излагает материал; свободно справляется с задачами, вопросами и другими видами применения знаний; использует для написания доклада современные научные материалы; анализирует полученную информацию; проявляет самостоятельность при написании доклада.

Оценка «хорошо» выставляется студенту, если качество выполнения доклада достаточно высокое. Студент твердо знает материал, грамотно и по существу излагает его, не допуская существенных неточностей в ответе на вопросы по теме доклада.

Оценка «удовлетворительно» выставляется студенту, если материал доклада излагается частично, но пробелы не носят существенного характера, студент допускает неточности и ошибки при защите доклада, дает недостаточно правильные формулировки, наблюдаются нарушения логической последовательности в изложении материала.

Оценка «неудовлетворительно» выставляется студенту, если он не подготовил доклад или допустил существенные ошибки. Студент неуверенно излагает материал доклада, не отвечает на вопросы преподавателя.

Темы рефератов по дисциплине «Большие данные»

- Архитектура системы обработки больших данных**.**
- Параллельные алгоритмы для работы с большими данными**.**
- Программные платформы и системы для больших данных**.**
- Центры обработки больших данных**.**
- Методы хранения больших данных**,** базы данных.
- Платформы больших данных**.**
- Методы и методики аналитической обработки больших данных**.**
- Алгоритмы кластеризации больших данных**.**
- Нечёткие методы представления данных больших данных**.**
- Роль аналитика по данным **(Data Scientist).**
- Нейронные сети как реализация алгоритмов машинного обучения**.**
- Применение технологий больших данных для задач управления в реальном времени**.**
- Роль корреляции и регрессии в аналитике больших данных**.**
- Задачи классификации и кластеризации больших данных**.**
- Подготовка данных и больших объёмов данных**,** анализ, визуализация, презентация**.**
- Проблемы анализа и обработки большого объёма данных**.**
- Базовые принципы обработки больших данных**.**

Список литературы для СРС

Перечень основной литературы:

1. МакКинни У. Python для анализа данных / У. МакКинни. – 2-е изд. – СПб.: Питер, 2020. – 544 с. – ISBN 978-5-4461-1345-3.
2. Теейт Р. М. П. SQL для Data Scientists: Руководство для начинающих по созданию наборов данных для анализа / Р. М. П. Теейт. – М.: ДМК Пресс, 2021. – 280 с. – ISBN 978-5-97060-912-7.
3. Брэдшоу Ш. Mongo DB: Полное руководство / Ш. Брэдшоу, Э. Бразил, К. Чодоров. – М.: Вильямс, 2020. – 720 с. – ISBN 978-5-9909445-8-7.
4. Уайт Т. Hadoop: Подробное руководство / Т. Уайт. – 4-е изд. – СПб.: Питер, 2019. – 768 с. – ISBN 978-5-4461-1157-2.
5. Дамджи Дж. Изучаем Spark / Дж. Дамджи, Б. Уэниг, Т. Дас, Д. Ли. – М.: ДМК Пресс, 2021. – 480 с. – ISBN 978-5-97060-899-1.

Перечень дополнительной литературы:

1. Парк Э. Data Science для начинающих / Э. Парк. – М.: ДМК Пресс, 2020. – 320 с. – ISBN 978-5-97060-835-9.
2. Марц Н. Большие данные: Принципы и практика построения масштабируемых систем обработки данных в реальном времени / Н. Марц, Дж. Уоррен. – М.: Вильямс, 2015. – 480 с. – ISBN 978-5-8459-1992-2.
3. Клеппман М. Высоконагруженные приложения: Программирование, масштабирование, поддержка / М. Клеппман. – М.: Питер, 2019. – 592 с. – ISBN 978-5-4461-1108-4.
4. Чарльз Д. Искусство обработки данных: Как извлечь знания из больших данных / Д. Чарльз. – М.: ДМК Пресс, 2018. – 360 с. – ISBN 978-5-97060-567-9.
5. Рамачандра К. Основы Data Science и Big Data: Python и наука о данных / К. Рамачандра. – М.: Вильямс, 2020. – 400 с. – ISBN 978-5-9909445-7-0.
6. Майер-Шёнбергер В. Большие данные: Революция, которая изменит то, как мы живем, работаем и мыслим / В. Майер-Шёнбергер, К. Кукьер. – М.: Манн, Иванов

и Фербер, 2014. – 240 с. – ISBN 978-5-91657-947-9.

Перечень ресурсов информационно-телекоммуникационной сети «Интернет», необходимых для освоения дисциплины

1. https://hadoop.apache.org/docs/current/ - документация Hadoop.
2. https://spark.apache.org/docs/latest/ - документация Apache Spark.
3. https://docs.mongodb.com/ - документация MongoDB.
4. https://pandas.pydata.org/docs/ - документация Pandas.
5. https://scikit-learn.org/stable/ - документация Scikit-learn.
6. https://www.kaggle.com/ - Платформа для работы с реальными наборами данных и участия в соревнованиях по Data Science.
7. https://datasetsearch.research.google.com/ - Поиск открытых наборов данных для анализа.