Приложение к рабочей программе дисциплины

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ОРГАНИЗАЦИИ САМОСТОЯТЕЛЬНОЙ РАБОТЫ ОБУЧАЮЩИХСЯ

Наименование дисциплины	*Машинное обучение на больших данных*
Направление подготовки / Специальность	*38.03.01 Экономика*
Направленность (профиль) / Специализация	*Экономика и анализ данных*
	ОП *ВО*
Форма обучения	*очная*

Разработчик Капелюк С.Д., профессор научно-учебной лаборатории исследований рынка труда

1. Темы дисциплины для самостоятельного освоения обучающимися
Отсутствуют.

2. План самостоятельной работы:

№ п/п	Учебные встречи	Виды самостоятельной работы	Форма отчетности / контроля	Количество баллов	Рекомендуемый бюджет времени на выполнение (ак.ч.)
1.	Базовые подходы к хранению и извлечению информации	1. Подготовка к тестированию	1. Тестирование	-	2
		2. Выполнение задания для самостоятельной работы	2. Выполнение работы	2	10
2.	Bash для инженера данных	1. Подготовка к тестированию	1. Тестирование	-	2
		2. Выполнение задания для самостоятельной работы	2. Выполнение работы	2	10
3.	Начало работы с большим объемом данных: Hadoop, MapReduce	1. Подготовка к тестированию	1. Тестирование	-	2
		2. Выполнение задания для самостоятельной работы	2. Выполнение работы	2	10
4.	Знакомство со Spark	1. Подготовка к тестированию	1. Тестирование	-	2
		2. Выполнение задания для самостоятельной работы	2. Выполнение работы	2	10
5.	Продвинутое использование Spark	1. Подготовка к тестированию	1. Тестирование	-	2
		2. Выполнение задания для самостоятельной работы	2. Выполнение работы	2	10
6.	Развертывание ML-моделей	1. Подготовка к тестированию	1. Тестирование	-	2
		2. Выполнение задания для самостоятельной работы	2. Выполнение работы	2	10

7.	Анализ системы и продуктовая аналитика	1. Подготовка к тестированию	1. Тестирование	-	2
		2. Выполнение задания для самостоятельной работы	2. Выполнение работы	2	10
8.	Полный цикл разработки ML-сервиса	1. Подготовка к тестированию	1. Тестирование	-	2
		2. Выполнение задания для самостоятельной работы	2. Выполнение работы	2	10
9.	Оптимизация моделей, исполнение на клиенте	1. Подготовка к тестированию	1. Тестирование	-	2
		2. Выполнение задания для самостоятельной работы	2. Выполнение работы	2	10
10.	Поиск приближенного ответа	1. Подготовка к тестированию	1. Тестирование	-	2
		2. Выполнение задания для самостоятельной работы	2. Выполнение работы	2	10
11.	Подготовка к экзамену	Изучение материалов по дисциплине по вопросам к экзамену	-	-	22
	Итого			20	142

3. Требования и рекомендации по выполнению самостоятельных работ обучающихся, критерии оценивания

Вид: Подготовка к тестированию

Краткая характеристика: вид проверки знаний и умений учащихся, который направлен на выявление степени усвоения изученного материала. Оно содержит обобщенный материал по основным изученным темам, требует от учащихся хорошей ориентировки в явлениях и фактах.

Рекомендации для подготовки:

- просмотр видеолекций;
- чтение основной и дополнительной литературы;
- повтор изученного на лекционных и практических занятиях.

Пример теста:

1. Какие бывают архитектурные подходы для построения API?

а) REST

б) GraphQL

в) TCP

г) Beautiful Soup

2. У сети ресторанов есть веб-сайт, где пользователи могут бронировать столики. Как

будет выглядеть REST ресурс, по которому можно узнать информацию о столике с идентификатором 75 в ресторане с идентификатором 3?
а) /restaurants/3/tables/75
б) /tables/75/restaurants/3
в) /restaurants/75/tables/3
г) /tables/3/restaurants/75
3. Что из перечисленного НЕ относится к свойствам больших данных (3V)?
а) Высокая структурированность
б) Высокая скорость генерации
в) Большой объем
г) Высокое разнообразие
4. Чем отличаются документоориентированные базы данных?
а) Каждая запись представляет собой независимый объект — документ
б) Данные хранятся в формате DOC
в) Данные описываются узлами и рёбрами ориентированного графа
г) Любые запросы выполняются гораздо быстрее
5. Для чего используется SQL Join?
а) Для соединения нескольких таблиц в одну по указанному условию
б) Для группировки данных по указанному полю
в) Для агрегации данных
г) Для оптимизации запросов
6. Можно ли делать запросы к реляционной СУБД из Python?
а) Да, например, с помощью библиотеки records
б) Нет, так как SQL не является частью синтаксиса Python
в) Выберите верные утверждения про Spark
г) Хорошо подходит для итерационных алгоритмов
7. Какие бывают основные виды взаимодействия с веб-ресурсом?
а) Статический
б) Динамический
в) Параметрический
г) Энергетический
8. Какие задачи можно решить, используя автоматизированный веб-скрапинг?
а) Собрать данные о всех знакомых в социальной сети
б) Сохранить себе все статьи из русской Википедии
в) Переводить деньги с чужих счетов на свои
г) Полностью сохранить себе веб-сервис, чтобы можно было пользоваться им без доступа в интернет
9. Какой вариант ответа является корректным HTML-кодом?
а) This is header
б) <div class="header" This is header /div>
в)
г) This is header </header>
10. Представьте, что у вас есть коллекция books с полями _id, author, pub, year, title. Как получить коллекцию записей с полем title, исключая все остальные поля?
а) db.books.find({}, {"title": 1, "_id": 0})
б) db.books.findOne({}, {"title": 1})
в) db.books.select({"fields": ["title"]})
г) db.books.findAll({"fields": ["title"]})

Вид: Выполнение задания для самостоятельной работы
Краткая характеристика заданий: Задания направлены на развитие у студентов

практических навыков работы с большими данными: обработка данных, построение моделей и анализ результатов. Тематика заданий охватывает широкий спектр инструментов машинного обучения, которые применяются в практической работе с большими данными.
Рекомендации для подготовки: При выполнении заданий рекомендуется строго следовать формулировке задания и предоставлять результаты в указанном формате (например, JSON). Следует уделить внимание проверке корректности работы программного кода. Для успешного выполнения заданий рекомендуется уделить время изучению официальной документации по инструментам, указанных в задании.

Пример задания для самостоятельной работы к встрече «Базовые подходы к хранению и извлечению информации»:
Дан список пациентов, для которых указаны следующие параметры:
* Имя
* Возраст
* Пол
* Диагноз
* Страна проживания
Для каждого пациента необходимо в базе AACT (Aggregate Analysis of Clinical Trials) найти список подходящих ему испытаний, в которые он может быть записан (в своей стране). Рассматриваются испытания, где используются только препараты (Drug). Каждое испытание имеет свою фазу. Следует учесть, что для пациента предпочтительнее попасть на более позднюю фазу испытания: это повышает его шансы на выздоровление. Порядок фаз следующий:
* Early Phase 1
* Phase 1
* Phase 1/Phase 2
* Phase 2
* Phase 2/Phase 3
* Phase 3
* Phase 4
Ответ должен быть записан в виде файла result.json.

Пример задания для самостоятельной работы к встрече «Bash для инженера данных»:
В текущей директории лежит зашифрованный файл secret.txt и скрипт task.py, который этот файл умеет дешифровать. Однако этот скрипт хочет удостовериться, что вы действительно хотите провести дешифровку, поэтому вначале спрашивает вас 1000 раз, уверены ли вы.
Если все 1000 раз ответить *y*, то тогда этот скрипт дешифрует тот файл, который будет указан в первом аргументе командной строки. Результат дешифровки он напечатает на экран. Ваша задача: дешифровать файл secret.txt и сохранить результат в файл result.txt.

Пример задания для самостоятельной работы к встрече «Начало работы с большим объемом данных: Hadoop, MapReduce»:
Файл events.csv содержит записи вида Пользователь, Исполнитель, Число прослушиваний, Число пропусков.
Задание:
1. Оставьте в данных только тех пользователей, для которых сумма plays строго больше 1000. Сколько таких пользователей?
2. В отфильтрованных на первом шаге данных найдите 5 самых популярных по числу пользователей исполнителей (идентификаторы).
Решение сохраните в файл result.json.
Пример содержимого файла:
json { "q1": 123, "q2": [4, 5, 6, 7, 8] }

Пример задания для самостоятельной работы к встрече «Знакомство со Spark»:
Задание очень похоже на задание, которое вы уже делали на Hadoop. Оцените, насколько проще оно решается на Spark.
Файл events.csv содержит записи вида Пользователь, Исполнитель, Число прослушиваний, Число пропусков.
Задание:
3. Оставьте в данных только тех пользователей, для которых сумма plays строго больше 2000. Сколько таких пользователей?
4. В отфильтрованных на первом шаге данных найдите 5 самых популярных по числу пользователей исполнителей (идентификаторы).
Решение сохраните в файл result.json.

Пример задания для самостоятельной работы к встрече «Продвинутое использование Spark»:
Рассмотрим задачу классификации на следующие два класса:
- 1 для 'American_movie_actors'
- 0 для 'American_stage_actors'
Проверьте, как поведет себя модель после использования хэширования, и ответьте на следующие вопросы:
1. Какой roc_auc_score на тестовой выборке получается при использовании словаря?
2. Какой roc_auc_score на тестовой выборке получается при переходе со словаря на хэширование?
Порядок выполнения задания:
1. Разбейте выборки на обучающую и тестовую по четности `id` статьи: четные в обучение, нечетные в тест.
2. Для подсчета roc_auc_score вам нужно получить предсказания и истинные ответы для примеров из тестовой выборки. Все пары (предсказание, ответ) помещаются в память.
3. В качестве хэш-функции используйте `murmurhash3_32(x) % 2**20`.
4. Зафиксируйте random seed в начальном приближении весов: `np.random.seed(0); weights = np.random.random(...)`
5. Обучите 500 эпох с шагом 0.3. После каждой эпохи вызывайте `weights_broadcast.destroy()` для удаления broadcast переменной, чтобы не закончилась память.
Решение сохраните в файл `result.json`. Пример содержимого файла:
```json
{
  "q1": 0.123,
  "q2": 0.456
}```

Пример задания для самостоятельной работы к встрече «Развертывание ML-моделей»:
В файле secret.txt зашифрован текст. Существует публичный образ adkosmos/decryptor:latest - он может расшифровать этот файл. Необходимо используя этот образ расшифровать файл secret.txt. Декодированный текст необходимо записать в result.txt.
Образ работает следующим образом: в качестве аргументов командной строки он принимает путь до файла, который нужно декодировать. Декодированную информацию он печатает на экран. Необходимо пробросить файл secret.txt внутрь этого контейнера, декодировать содержимое, скопировать и вставить в result.txt.

Пример задания для самостоятельной работы к встрече «Анализ системы и продуктовая
```

аналитика»:
Необходимо создать офлайн-метрику для оценки популярности сайта новостного интернет-издания «The Guardian» в поисковой выдаче по запросам, связанным с футбольными клубами.
1. Создать метрику, в которой учитывается минимальный ранг документа. Для каждой пары (запрос, регион) вычислить минимальный ранг. Чем этот показатель меньше, тем лучше. Итоговая метрика: сумма таких показателей по всем имеющимся парам. Полученное значение нужно записать в первую строчку выходного файла.
2. Модифицировать метрику, чтобы она по-прежнему работала с рангами, но учитывала множественные вхождения документов в одну выдачу. Полученное значение нужно записать во вторую строчку выходного файла.
3. Оценить, насколько хорош самый лучший сайт по метрике "максимальное значение 11 - rank". Написать в третью строчку домен displayLink, который имеет наибольшее значение этой метрики.

Пример задания для самостоятельной работы к встрече «Полный цикл разработки ML-сервиса»:
Провести эксперимент и ввести в эксплуатацию новую обученную модель.
1. С помощью MLflow Tracking в существующем проекте провести новый эксперимент с параметрами alpha = 0.65, l1_ratio = 0.45.
Зафиксировать результаты в системе, используя контекст mlflow.start_run и сохраняя параметры и метрики, вызвав mlflow.log_param и mlflow.log_metric соответственно.
2. С помощью MLflow Model Registry зарегистрируйте проведенный эсперимент как модель для ввода в эксплуатацию.
Используя клиент mlflow.tracking.MlflowClient (переменная client) найдите проведенный эксперимент (например, по имени с помощью client.get_experiment_by_name(experiment_name))
Получите информацию о последнем запуске с помощью client.list_run_infos
С помощью MLflow Model Registry создайте новую версию из зарегистрированной модели. Для этого с помощью client.create_registered_model зарегистрируйте модель под именем coursera-top-model, затем с помощью client.create_model_version создайте первую версию, указав также идентификатор запуска эксперимента из run_info и путь к модели, который также доступен в run_info.
3. С помощью MLflow Model Registry выложите модель на тестовый сервер и проверьте, что API сервера возвращает результат.
Для этого с помощью client.transition_model_version_stage укажите имя зарегистрированной модели, версию и среду (stage).
Затем запустите сервер с моделью, используя консольный интерфейс mlflow: ! mlflow models serve -m "models:/имя_зарегистрированной_модели/Staging" -p 5005.
Убедитесь, что сервер работает с помощью HTTP POST-запроса, используя библиотеку requests. Для этого возьмите несколько объектов из тестовой выборки и сформируйте запрос, сконвертировав данные в формат JSON: test_x[:10].to_json(orient='split'). Затем отправьте POST-запрос, указав в заголовке формат JSON: requests.post(url=url, headers={'Content-Type': 'application/json'}, data=http_data), где url - адрес сервера с моделью (не адрес сервера MLFlow), например http://127.0.0.1:5005/invocations.
4. Проведите аналогичные пункту 3 действия для эксплуатационного сервера.
5. После выполнения задания, выполните код generate_submission(), чтобы сгенерировать файл submission.json для отправки задания на проверку.

Пример задания для самостоятельной работы к встрече «Оптимизация моделей, исполнение на клиенте»:
Ниже представлена простая сверточная архитектура LeNet. Задача: проредить сеть без

сильной потери в качестве.

```
class LeNet(nn.Module):
    def __init__(self):
        super(LeNet, self).__init__()
        self.conv1 = prune.identity(nn.Conv2d(1, 6, 3), "weight")
        self.conv2 = prune.identity(nn.Conv2d(6, 16, 3), "weight")
        self.fc1 = prune.identity(nn.Linear(16 * 5 * 5, 120), "weight")
        self.fc2 = prune.identity(nn.Linear(120, 84), "weight")
        self.fc3 = prune.identity(nn.Linear(84, 10), "weight")

    def forward(self, x):
        x = F.max_pool2d(F.relu(self.conv1(x)), (2, 2))
        x = F.max_pool2d(F.relu(self.conv2(x)), 2)
        x = x.view(-1, self.num_flat_features(x))
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x

    def num_flat_features(self, x):
        size = x.size()[1:]
        num_features = 1
        for s in size:
            num_features *= s
        return num_features

    def prune(self, *args, **kwargs):
        pass
```

Чтобы сдать задание, необходимо проредить модель и сохранить ее на диск в файл result.bin. На стороне проверки модель будет восстановлена из этого файла и у нее будет проверено качество и количество параметров. Качество будет замеряться на этом же датасете. Используйте инструменты из torch.nn.utils.prune.

Пример задания для самостоятельной работы к встрече «Поиск приближенного ответа»:
Дописать функцию подсчёта хэшей для LSH с метрикой cosine. В ответе сохранить ответы и полученные значениия метрик при запросах к LSH с указанными параметрами размерности хэша в сравнении с реализацией NearestNeighbors.

- Описать функцию _hash для CosineLSH (код ниже).
- Получить список 10 ближайших для запросов из HOMETASK_QUERIES.
- Посчитать Precision и Recall считая результаты реализацией поиска ближайших соседей (sklearn.neighbors.NearestNeighbors) истиной, пользуясь функцией класса calc_metrics (пример ниже).

В качестве результата работы ожидается сдача файла cosine_lsh.py, файл должен содержать класс CosineLSH, реализующий требования задания.

4. Рекомендации по самоподготовке к промежуточной аттестации по дисциплине

Вопросы для самопроверки к экзамену

1. Введение в большие данные: определение и основные понятия.
2. Сущность и особенности реляционных баз данных.
3. Основные элементы реляционных баз данных.

4. Основы SQL.
5. Запросы и фильтрация данных в SQL.
6. Документоориентированные базы данных.
7. Основные команды MongoDB.
8. Базовые подходы к извлечению информации из сети Интернет.
9. Основные техники веб-скрапинга.
10. Работа с API.
11. Основы Bash для инженера данных.
12. Управление потоками данных в Bash.
13. Конвейеризация команд в Bash.
14. Автоматизация задач с Bash.
15. Основные команды и утилиты Bash.
16. Архитектура и компоненты экосистемы Hadoop.
17. Функциональные возможности HDFS (Hadoop Distributed File System).
18. Основы парадигмы MapReduce.
19. Задача Word Count в MapReduce.
20. Использование Apache Spark для обработки больших данных.
21. Работа со Spark RDD.
22. Spark SQL.
23. Практическое использование Spark.
24. Классификация текстов с использованием Spark.
25. Реализация модели "мешка слов" и логистической регрессии в Spark.
26. Проблема перекошенного ключа в Spark.
27. Векторизация текстов и создание словаря в Spark.
28. Основы развертывания моделей на вычислительных кластерах.
29. Архитектурные подходы для построения систем обработки данных.
30. Контейнеризация и оркестрация.
31. Инструменты Celery и Flask.
32. Синхронная и асинхронная обработка запросов.
33. Основы продуктовой аналитики.
34. A/B-тестирование.
35. Онлайн- и офлайн-метрики в продуктовой аналитике.
36. MLFlow.
37. MLOps.
38. Оптимизация моделей машинного обучения.
39. Оптимизированные архитектуры для запуска на мобильных устройствах.
40. Поиск приближенного ответа.