Data Science in Production Exam

Spring 2022

YOUR NAME HERE

IT University of Copenhagen
YOUR EMAIL HERE@itu.dk
May 9, 2022

Rules

Code of conduct

In answering the questions and working on the exercises, you are allowed to use the lecture material (slides, video recordings), as well as external resources (tutorials, books). Remember that this is an **individual** exam. You are not allowed to discuss the exam with other candidates. You are not allowed to share any resource, pointer, answer, code, or solution (anything, in fact) with any other candidate, directly or indirectly. Not adhering to this code of conduct will result in the invalidation of the exam for both the candidates who shared information and for the ones that received it.

Assignments and hand-in

You will receive a set of 20 questions (below), and a set of 9 exercises. **You must answer 16 questions of you choice and solve 7 exercises of your choice. Do not complete more questions or exercises than the required amount – your score will be determined by taking into account the wrong answers/exercises first.** For the exercises, you will receive a directory with data; you must not modify the structure or content of that directory.

You will deliver your results in the form of a zipped directory, containing:

1. A pdf including questions and answers
2. A directory named "exercises" with one subdirectory per exercise, named "exercise-n" (where n is the exercise number). The code and material of each exercise should be included in the exercise subdirectory, **with the exception of the data**. When running your exercises, I will assume that the data is stored in a "data" directory placed in the base directory of your assignment. You do not need to submit the data together with your assignment.

Figure 1: Example structure of your exam directory. The data directory is the one provided, but should not be included in the final hand-in (because your examiner already has it).

Questions

Instructions. Use the Latex file provided to write your answers. Write your name and email in the header. Write your answer right after each question. Keep the question text on your answer paper, but you can erase the Rules Section in the previous page, as well as this instruction paragraph. Answers should be rather brief, one short paragraph each. As a rule of thumb, the final paper containing questions and answers shouldn't be longer than 6 pages using the Latex template provided. There is no need to include any formula nor illustration to answer the questions correctly. However, it is important to motivate your answers. Some questions might have multiple correct answers with different motivations; you don't need to list all possible answers – one properly motivated correct answer is sufficient. The information needed to answer all the questions is contained in the slides and lecture recordings.

Q1. List three possible ways of sorting the postings lists in an inverted index, and briefly discuss at least one advantage and one disadvantage of the different sorting strategies.

A1. Your answer here

Q2. Given a phrase query, how are search engines able to find relevant documents that contain that phrase query but with tokens that are not contiguous? For example, a document containing the sentence *"a black fat cat ate my homework"* is found relevant to the phrase query *"black cat"*. Please briefly describe the algorithms or datastructures that are relevant to this case.

A1. Your answer here

Q3. The chief of search of an e-commerce service is tasked to evaluate the effectiveness of the website's search functionality. The marketing department has found that a good list of search result is one that includes at least one good relevant result towards the top of the list. One newly-hired data scientist in the team suggests that a good metric to evaluate search performance could be precision, calculated as the fraction of documents retrieved that are deemed relevant. Do you agree with the suggestion and why? Would you suggest to change or improve the evaluation metric?

A3. Your answer here

Q4. A study panel with users of a search engine for legal documents reveals that users find that on average the set of returned results for a query is too small. Interviews revealed that users often search for fairly technical terms that are found in only a handful of documents, but they would like to have an overview of documents that are topically related to their search intent, even if they don't contain the query. How would you change the search engine to better meet the users' needs? Explain briefly the technique you would use.

A4. Your answer here

Q5. A company wants to build a new search engine for patents to compete with Google Scholar. They have invested in a crawling infrastructure to download the title, abstract, and main body of millions of patents across the Web. The chief of staff does not know which of the three fields of text should be the best to be indexed for search queries. What would you suggest? Could you briefly describe how you would structure your suggested search strategy in the production search engine?

A5. Your answer here

Q6. Your manager has just seen a Youtube tutorial about Learning To Rank (LTR) and he is so excited about it that he wants you to implement it for the company's Web search engine (an engine that indexes all types of web pages, very much like Google does). He proposes to implement the following workflow. For each query, LTR should be used first to rank all possible results. Then, to improve the quality of top 10 results, a re-ranking based on the documents PageRank should be performed. Do you agree with the manager and why? What improvements would you possibly propose, if any?

A6. Your answer here

Q7. You work in the engineering department of an online forum. In the forum, user posts

can be voted by other users on a scale from 1 (bad posts) to 5 (good posts). Management wants to implement a new feature to show "hot" posts, namely the most interesting posts published recently. A colleague of yours proposes that a simple but effective baseline to implement the service could be to show the set of posts with highest average score among the posts published in the last 24 hours. Is it a good idea and why? Would you propose any improvement to it? Please briefly describe your alternative solution, if any.

A7. Your answer here

Q8. Explain the concept of monotonicity in the task of frequent itemset mining, and state why it is important for algorithms that extract frequent itemsets.

A8. Your answer here

Q9. A new movie recommendation service has implemented a very sophisticated system of catalog creation. As a result, the service database contains virtually all movies ever shot and it is continuously updated with new movies. All movies are rich of metadata (actors, genres, box office performance, etc.). The service is still struggling to attract users, and at the moment only a very small set of hardcore, very active users is using it. Typically, these users get recommendations and rate the movies they have been recommended after they watch them. The management suspects that the site is not doing so well because the recommendations are given at random, and wonders which recommender systems should be implemented to keep existing users engaged, and to provide meaningful recommendations to the few new users who land on the platform for the first time. What solution(s) would you recommend to implement and why?

A9. Your answer here

Q10. Describe a method to find efficiently the nearest neighbors of a given user in a user-user collaborative filtering system. Contextualize your description in a scenario where all users and items can be mapped to a n-dimensional latent space.

A8. Your answer here

Q11. Given an $m \times n$ rating matrix R, with m users (rows) and n items (columns), explain what left/right singular vectors and singular values of the SVD decomposition of R represent.

A11. Your answer here

Q12. Provide a high-level description (i.e., skipping implementation details) of a social link recommender system based on a local topology feature of your choice. Imagine a scenario of undirected, unweighted links (e.g., Facebook friendship). Describe how you would conduct an offline evaluation of that system.

A12. Your answer here

Q13. A music recommender system uses a personalized bandit model for each user to recommend songs to listen to. The system is rudimentary, as it considers only three genres (metal, hip-hop, and electronic) and when a genre is selected, a random song from that genre is recommended. The empirical distributions of rewards (i.e., explicit user ratings) collected by the bandit so far have the following averages and standard deviations: $\mu_{metal} = 2.8$, $\sigma_{metal} = 0.5$; $\mu_{electronic} = 3.3$, $\sigma_{electronic} = 1.0$; $\mu_{hiphop} = 3.6$, $\sigma_{hiphop} = 0.1$. Can we infer something about the user preferences given this information? If so, what? If the bandit algorithm uses Thompson Sampling, which are the genres (or genre) that will be likely recommended next and why?

A13. Your answer here

Q14. Describe a way to evaluate the quality of recommendations from a bandit offline, using historical rating data. Discuss potential disadvantages of the method.

A14. Your answer here

Q15. Your company wants to advertise its product online in the banner of a popular

website. The marketing chief doesn't know much about online ads, so it puts you on the phone with the representative of an advertising marketplace. The person on the phone offers you two different deals. The first deal follows a cost-per-time billing model with a priced-floored, first-price auction. The second deal follows a cost-per-click model with a second-price auction. Which deal do you pick and why?

A15. Your answer here

Q16. Your company is at an historical crossroad. The CEO decided to change the color of the logo for good. The three top color options from the design department are pink, lime, and turquoise, and the data science department is asked to find which one is most effective in attracting website visits. One day, you find two chiefs of data science fighting in the coffee room over this issue. One believes that a bandit approach should be used to determine the best color. The other believes that A/B testing should be used instead. Who do you think is right and why?

A16. Your answer here

Q17. Two word-count jobs are to be submitted to your Hadoop cluster. Each of the two jobs takes in input a different batch of one trillion documents each. Documents in the first batch are written in an ancient language whose words are uniformly distributed: all words have roughly the same probability of occurrence. Documents in the second batch are written in English. Which of the two tasks do you think will finish first and why?

A17. Your answer here

Q18. Your engineer friends are working on a tabular dataset of a supermarket chain. The dataset contains two columns: one for the product name and another for the product price. They are asked to write some Spark code to i) fix wrong punctuation in the title, ii) capitalize the title, iii) adjust the price by 3.3% inflation. Your friends are quite confident that, after loading the dataset, the computation won't be too heavy because it will most likely take only one Spark stage to complete. Are they right and why?

A18. Your answer here

Q19. Your roommate has finished an online Docker course and now they are all hyped-up. They want to build a dockerized version of their own webpage. They plan to write a Dockerfile to define a container. To enhance performance, they will skip the OS layer and have a bare-metal installation of Flask as first layer. As a second layer, they will have the latest Python version, to ensure that Flask can run properly. They will then declare a volume to open the web application to an internal IP and port, so that anyone can access the website from the World Wide Web. Once the container is defined, they plan to instantiate it in an image on a cloud service, so that the website becomes live. Please explain what's wrong with your roommate's plan.

A19. Your answer here

Q20. What is the concept of "Desired state management" in Kubernetes (k8s) and what are the components that help achieving it?

A20. Your answer here

Exercises

Instructions. Follow the directory structure illustrated in the Rules section (first page of this pdf). Use Python notebooks whenever possible. When design or implementation details are not specified, you can make your own choices.

Exercise 1: Indexing

Use Elasticsearch to index and query the Tesco dataset (file data/tesco/tesco_inventory_clean.csv). Indications on how to perform the indexing and querying are provided next. Write your code in a single Python notebook. You are free to use the Elasticsearch wrapper API, the custom helpers we have used in class, or both. For each of the points below, write some brief comments (or text cells) in your notebook to describe how you addressed that specific point. Use the numbering provided below to refer to the points in your notebook (e.g., *"Point #1: I took care of efficient loading by ..."*)

Indexing. Please index the dataset taking into account the following points:

1. The dataset needs to be loaded efficiently
2. This is intended to be a relatively static database, updated only once a month or so
3. Create an index with three replicas.
4. The documents should allow full text search on the textual description and ingredients, and should allow filtering by category
5. A relevance metric that allows for text length discounting should be used. Decrease by 20% the amount of length discounting compared to the default value

Querying. For each of the points below, write a separate Python parametric function that performs:

7. An exact-match query on the product description field
8. A full-text query on description and ingredients, with description boosted by a factor 2 compared to ingredients
9. A full-text query on both description and ingredients, and that filters the results by category
10. A fuzzy query on the ingredients
11. A full-text query on description, with results sorted by price

Test your functions with one or two example queries each.

Exercise 2: Relevance feedback

Use Elasticsearch to index the arXiv abstracts dataset (directory data/arxiv). The dataset is a reduced version of the data from `https://www.kaggle.com/Cornell-University/arxiv`. It contains the title and abstract of scientific papers. Write a Python function that implements pseudo-relevance feedback in Elasticsearch. Specifically, when a full-text query is submitted, a set of relevant documents is retrieved. Out of this set, important terms should be identified and used to expand the query. The expanded query is then submitted to the index and the final result returned. You can focus on the text of the abstract only, and ignore the title.

Write your code in a single Python notebook. You are free to use the Elasticsearch wrapper API, the custom helpers we have used in class, or both. You should write your own code to implement the relevance feedback functionality; do not use additional plugins or libraries.

Test your function with a few queries and compare results with full-text queries that do not use expansion.

Exercise 3: Learning to rank

The Yahoo Learning To Rank Challenge dataset contains features extracted from (query,url) pairs along with relevance judgments. The queries, urls and features descriptions are not given, only the feature values are. The data is available in the directory (data/yahoo_ltr), and contains a training and test sets, see readme file for format details (tldr; the files are in RankLib format).

Train a set of LTR models (ListNet, RankNet, LambdaMART, and RankBoost) and evaluate their performance on the test set using MAP, NDCG@10, and P@10. Summarize the results in three barplots,

one for each evaluation metric, where each bar represents the performance of one of the LTR models. Write all your code in a Jupyter notebook. In a text cell, comment briefly the results: Did you expect specific differences between models? Any remark about the results obtained across performance metrics?

Exercise 4: Frequent itemsets

To prepare a new promotional campaign, the Tesco marketing department is interested in two pieces of information.

- A list of triplets of products (A, B, C) such that C is purchased frequently after A and B. They are interested in strong associations, such that the likelihood of the association is at least five times higher than chance.
- A list of pairs (A, B) such that B is frequently purchased after A, and that the pair (A, B) is found least in 1% of the carts.

They provide you with a list of carts (file data/tesco/tesco_carts.pickle, in pickle format). Each cart is a list of product ids. The respective descriptions of those ids is given in the file data/tesco/tesco_inventory_clean.csv. Write a Python notebook that satisfies the request of the marketing department. Explain your choices briefly in text cells of your notebook.

Exercise 5: Recommender system

In class (lecture #6), we discussed two datasets useful for experimenting with recommender systems: the food.com dataset (directory data/food.com) and the movielens dataset (directory data/movielens). Pick one and use it for this exercise. If you pick movielens, you can use the small dataset (ml-latest-small) to experiment and the large dataset (ml-latest) to report the final results. If you pick the food.com dataset, be aware that random timestamps have been artificially added to the data, as the temporal component was not available in the original dataset.

Perform a temporal split of the dataset (80% training including the oldest ratings, 20% test including the newest ratings). Implement the following recommendation systems using the training set to train them:

1. Most popular items (non-personalized)
2. Content-based recommender based on nearest neighbors (using any combination of features you like)
3. User-user collaborative filtering
4. SVD

Each recommender system should output a ranked list of up to 100 recommended items for each user, along with the predicted ratings. The recommendation quality is measured on the test set with two metrics: 1) Mean Average Precision (MAP) and 2) Root Mean Square Error (RMSE) calculated between the predicted rating and the actual rating, only for those items that are present both in the list of recommended items for user u and that user u has consumed in the test set.

Write all your code in a single Python notebook, and present the results in two bar plots, one for each of the two metrics, where each bar reports the score of one of the methods. To implement the recommender systems, you can use custom code or rely on existing Python libraries, or both.

In a text cell in the notebook, write a brief reflection about the results obtained, highlighting expected findings and surprising findings, if any.

Exercise 6: Bandits!

You are provided with a list of rewards that users expressed after consuming news items. The data consists of just two columns, one for the news article type (a number from 1 to 5) and one for the reward (in the range $[0, 10]$). The user ids are not reported – you can pretend that all the rewards come from the same user. The data is in the directory data/bandits.

Find a valid way to evaluate two bandits algorithms of your choice in an offline fashion on the provided dataset. As evaluation metric, you should use cumulative reward after 5000 arm pulls. Compare the performance of the two algorithms with five naive strategies that consist in pulling exclusively each one

of the five arms (e.g., naive strategy 1 pulls always arm number 1). For each algorithm (including the five baselines) plot how the cumulative reward grows over time (using one or more lineplots).

Write all your code in a Python notebook, and in a text cell write a brief comments about the result, mentioning especially expected and unexpected findings.

Exercise 7: A/B testing

The online customer service of a phone company prompts the customers with a small survey to evaluate the service at the end of the session. The customer can rate the service on a continuous slider that goes from a minimum rating of 0 to a maximum rating of 5. After 5000 ratings, the manager of the service decides to roll out a new automated assistant feature powered by an AI module. To test the effectiveness of the new system, the manager decides to run an A/B test, where 10% of the traffic is directed to the new AI system and 90% to the old system. The service usually receives 100 customers every day. After 10 days, the manager stops the A/B test and hands the data over to the Data Science department (you), to establish whether the new AI system had an effect. Management would like to see an *absolute* increase of 0.1 in the average user satisfaction score to roll out the feature to the whole user base.

In the directory (data/abtest) you find a table with two columns: the first including the rating from the users, the second containing a marker that indicates whether that rating was collected prior to the A/B test (*historical*), or during the A/B test in the *control* group or in the *treatment* group (i.e., those customers exposed to the AI service).

Write some Python code in a notebook to analyze the data and to determine the significance and magnitude of the effect. Write also some code to check whether the A/B test ran for long enough; if not, provide a recommendation to the manager on how much longer the test should run for. Comment your reflections and coding choices in the code, using text cells.

Exercise 8: Distributed computing

Write a contact recommender system that, given a user u in a directed unweighted social network, provides up to 10 suggestions of new contacts (fewer if there are not enough contacts that can be recommended). The 10 suggestions should be sorted by the Triangle Overlap (TO) measure, defined as:

$$TO(u,v) = \frac{CN(u,v)}{|\Gamma(u)|}, \tag{1}$$

where CN is the number of common neighbors between u and v , $\Gamma(u)$ is the set of neighbors of node u, and $|\Gamma(u)|$ is the cardinality of that set (i.e., the number of neighbors of u).

Write two implementations of the recommender system: one in Hadoop streaming (in Python) and one in PySpark. Test your code on the aNobii dataset (directory data/anobii). The dataset consists of two temporal snapshots of the network (time $t = 1$ and time $t = 2$). Calculate recommendations for all users on the network at time $t = 1$, and measure the quality of the recommendation by comparing the recommended contacts with the contacts created at time $t = 2$. If a node did not create any contact at time $t = 2$, you can discard the node from the evaluation. Use average precision at 10 (P@10) as evaluation metric. It is recommendable to write two separate jobs, one for the computation of recommendations and one for the evaluation.

You can test your Hadoop streaming code using piping (i.e., no need of installing Hadoop locally) and your Spark code using Google Colab (i.e., on the cloud, no need of installing Spark locally). Hand in your code in the form of Python files or notebooks. If you use Google Colab, please include all the instructions in the notebook to run the code correctly (e.g., including spark installation, data upload, etc.).

Exercise 9: Docker

Write a Docker application with two web interfaces, one open on port 9000 and one on port 9001. Using Web Interface A, the user can post a piece of text to the backend. The backend is composed by three separate microservices. Microservice X gets the text from the Web Interface A. It then provides the text to Microservice Y, which returns the number of words contained in the text. After that, it provides the text to Microservice Z, which returns the number of characters of the longest word in the text. Last,

Microservice X uses the two numbers (let's call them number 1 and 2) to update a persistent storage. The storage is supposed to store the average of numbers 1 and 2 over all users interactions. Last, Web Interface B can be accessed to see the two values stored in the persistent storage.

Deliver a folder with a docker configuration file and the resources (e.g., Python files, Docker files) that are needed to build the project. The backend microservices must be implemented in Python. You can use any tool you like to implement the frontend and the storage. Use Linux-based images.