Introduction to Machine Learning (67577)
Hackathon 2022

Teachers: Prof. Matan Gavish, Dr. Gabriel Stanovsky

TAs: Avihu Dekel, Dan Derazne, Gilad Green, Gili Lior, Michael Tomer

Tzars: Nitay Alon, Yuval Spiizer, Eitan Wagner

Second Semester, 2022

Contents

1 General Instructions

Welcome to the IML Hackathon 2022! Please read carefully the entire document before selecting a challenge to work on.

- On **Wednesday 01.06.2022 7pm** we all meet in the auditorium for the reveal of this years challenges. The hackathon ends on **Friday 03.06.2022 at 11am**.
- During the hackathon you will be working in teams of **three or four students only**.
- The instructions and data for both challenges will be available on the course Moodle.
- After reading through the entire document select **one** of the challenges and solve it as best as you can.
- **Please note:** both challenges are equally easy (or equally hard). Don't choose the challenge that looks easier - choose the challenge that looks more interesting and will be more fun to work on. You are far more likely to have good results on a challenge you find fun and interesting; Also, the performance you can hope to achieve is different between the two challenges.
- **We will provide you with a set of test examples for which you will submit your predictions**.
- You will also submit **a description of your project**, up to 2 pages long, as a PDF file. Explain your solution, describe your work process, and do your best to justify any design decisions you have made. Feel free to include supporting figures if you want.
- Your will also submit **your code**. To ensure we are able to run your code without errors we require all submissions to define and use a virtual environment. Submit the environment's `requirements.txt` file. Instructions on the virtualenv and the freeze command are available on Moodle. The virtual environment will allow you to install packages via `pip install <package\_name>` and use packages that are not installed on the CS computers.
- Solutions must be written in Python3. To use Python 3 on CS computers, use python3 instead of python.
- Submission of your solution is done through the course Moodle. Submit a single zip or tar file named `id_number.zip` (or .tar), where `id_number` is a 9-digits ID of one team member. The file should include:
  - A folder named `task1` or `task2`, depending on the challenge you choose. This folder should include:
    - `predictions.csv` – your prediction file.
    - Your code files.
    - Any supplementary files you used (trained models etc.).
    - `requirements.txt` — the required packages for your virtual environment, as described above.
  - `README.txt` — contains a file list and a brief description of each file.
  - `USERS.txt` — contains the logins and IDs of all the team members. Use one line per student, in the format login, ID.
  - `project.pdf` -– the description of your work.
- Make sure to write good, clean and documented code. Our ability to understand what you are performing and why is important for receiving a high grade.
- We advise to begin with finding a basic solution that works. Then try to improve it as much as

you can in the given time.

1.1 Evaluation

For both challenges your grade will be determined by two factors (in descending order of importance):

1. The quality of your prediction on the test set. This will be based on a ranking between the performance of all participants, where any team that will produce a model that has some reasonable performance ('reasonable' depends on which task you chose) will receive a grade of 80 and the top ranking teams will receive a grade of 100, and there will be some distribution between these two extremes.
2. The quality and depth of your written description of your work that you wrote in the PDF. We will grade your written description based on the following:
    - Did you describe the dataset, and any challenging characteristics it has?
    - Did you describe (briefly) the data cleaning and preprocessing?
    - Did you describe the considerations that guided your design of learning systems?
    - Did you describe (briefly) various methods you tried and the results you obtained?
    - Did you describe the learning system you ended up using?
    - Did you provide a prediction (and explanation) of the generalization error you expect your system to have?

Good Luck!

1.2 Verify Before Submission

IMPORTANT: Our tests have **zero tolerance** hence you have to follow the requested formats. Submission that fail to comply with the requested format will receive zero points.

- You are obligated to submit files in the defined file folder structure as below.
- Output data has to be in the matching format.
- Your code should only call files present in the submitted folder. When doing so use a relative path. For example : `"weights.txt"` is good while `"C:\Users\name\IML\weights.txt"` is bad.
- You are required to submit a `requirements.txt` file.
- Your code cannot throw any exception or error when running.
- Your code has to finish prediction (including any necessary loading or initialization) in a time frame of 5 minutes when running on the CS computers.
- After zipping your project, it should have a maximum size of 20 MB.
- Submit only **one** of the challenges (see directory structure).
- Submit the zip (or tar) file through the Moodle of only **one** of the team members.
- You must use the following directory structure in your submission zip file:

File or directory	**Description**
`<student-id>.zip`	Submission zip or tar file
├── `USERS`	
├── `README`	
├── `project.pdf`	
├── `task1`	***Only if submitting task 1***
│ ├── `predictions.csv`	
│ ├── `requirements.txt`	
│ ├── `regerssion.py`	
│ └── *`<other source files>`*	
└── `task2`	***Only if submitting task 2***
├── `predictions.csv`	
├── `requirements.txt`	
├── `classifier.py`	
└── *`<other source files>`*	

2 Hackathon Challenge 1: Waze!

Unless you live on the moon you probably used Waze for navigation when driving. The app's ability to plan the best route and react to new events made it popular. But merely reacting to events isn't enough - the ultimate goal is to predict events and solve congestion problems before they occur.
Your task, if you choose to accept it, is to answer the following questions - (1) What is the most likely next event given a sequence of events? (2) What is the distribution of events in a given time point?
For more inspiration about the importance of this task read this story: Waze and N12 traffic analysis.

2.1 Dataset

The data set holds about 41K traffic events with 19 features, collected between 2021-02-07 to 2022-05-24. Each row describes a single event.
Notice - not all features are guaranteed to appear in each row. The only features that are guaranteed to be present are ID, linqmap_type, x, y

2.1.1 Feature Description

- **id** - Unique identifier for the event - int
- **linqmap_type** - describing the event family - string
- **linqmap_subtype** - describing the event in details - string
- **pubDate** - the report date - string
- **linqmap_reportDescription** - the name of the city (Hebrew) - string
- **linqmap_street** - the street name (Hebrew) - string
- **linqmap_nearby** - interest points near the event (Hebrew) - string
- **linqmap_roadType** - road type code - string
- **linqmap_reportMood** - user mood (as assessed by Waze) - string
- **linqmap_reportRating** - report rating - int
- **linqmap_expectedBeginDate** - event expected beginning - string
- **linqmap_expectedEndDate** - event expected ending - string
- **linqmap_magvar** - orientation w.r.t to the north pole - int
- **nComments** - comments - string
- **linqmap_reliabi flity** - event reliability - int
- **update_date** - when the event was reported - string
- **x** - x coordinate of the event - int
- **y** - y coordinate of the event - int

2.2 Tasks

Please note that the tasks are independent. Partial submission will grant you partial grade.

2.2.1 Next Event Prediction

Given a sequence of 4 consecutive events in **Tel-Aviv** (ordered by time) predict the next event. That is, given a sequence of 4 events $x_1, \ldots, x_4$ predict the following features of the 5th event: (**linqmap_type**, **linqmap_subtype**, x coordinate, y coordinate).
In this section the evaluation method is a weighted combination of F1-macro loss for **linqmap_type**, **linqmap_subtype** and l2 loss for the location - $(\hat{x}-x)^2+(\hat{y}-y)^2$,
The input for this problem is a dataframe with groups of 4 events in Tel Aviv with same structure as the training data and a number indicating which group they belong to (the last column). **The output**

is a dataframe with a single row per group and 4 columns corresponding to the values above. There are **5** different input dataframes in the test set.

2.2.2 Event Distribution Prediction

Given a time range (start-end) predict the distribution of events **across the nation**. That is, for the following 3 time slots *8:00-10:00, 12:00-14:00, 18:00-20:00* predict the number of events of each type. The input is one of the dates: 05.06.2022, 07.06.2022 and 09.06.2022. The output is a **3 by 4 table** where each row corresponds to a time slot. 8:00-10:00 should be the first (top) row, 12:00-14:00 is second row, and 18:00-20:00 is the third row. The columns match the **linqmap_type**. That is, from left to right,

[ACCIDENT, JAM, ROAD_CLOSED, WEATHERHAZARD]

It is important that the your results table will match this specific order!!
In this section the grading is computed by the following weighted MSE:

$$\sum_{row}\sum_{t}\frac{(\hat{y}_{event,t}-y_{event,t})^2}{y_{event,t}+1} \tag{1}$$

where $\hat{y}_{event,t}$ is the number of predicted of events of some type at time t, $y_{event,t}$ is the actual number of events of that type at time t.
For example if there are 3 types of events - A,B,C and in the first time slot their frequency is $A = 52, B = 32, C = 1$ and the predicted values are $\hat{A} = 48, \hat{B} = 35, \hat{C} = 0$ then the loss for that time slot is 1.575
Please note, not all events types occur in each time slot.

2.3 Code Requirements

You need to submit a separate file for each task, named `task_i.py`, where *i* is the number of the task.
In addition you have to submit a `main.py` file that has an executable code (`if __name__ == "__main__" ...`).
In this file you implement a method called `main` that takes as input a relative path to the training set, relative path to task 1 test set and task 2 list of dates (in `pd.date` format - YYYY/MM/DD).
Each task code should run **independently** when we the main file is executed (that is, if task one fails task two should still run).
NOTICE Since we are not going to train your model, you are required to load your trained model in this method. Please verify this method works as expected (with all the additional files it is required - s.a. the weights file).

2.4 Performance Measure

As described in each section. You'll get partial grading for submitting only one of the tasks.

2.5 Tips

- Note the missing data - you are encouraged to use imputation methods to fill in the blanks.
- The dates are given in POSIX time - you'll have to convert them to dates you can work with.

3 Hackathon Challenge 2: Detecting Attributes of Breast Cancer

Breast cancer is one of the most pervasive malignant diseases among women in the world and in Israel in particular. If treated early and appropriately, the chance of remission is high.[1] In this task you're given (anonymized) data of breast cancer patients treated in Israel over the last years. Your task is to predict certain medical characteristics of the disease for each patient based on their available data. This can help save doctor's time, validate and "double-check" their decisions, and alleviate the cost of expensive tests.

If you choose to do this task, Part 1 and 2 are mandatory, Part 3 is highly encouraged.

Note:

You're given actual medical data provided with consent by the medical center after thorough anonymization for research purposes. Please treat it with respect and don't distribute it beyond the participants of the course. The findings made by hackathon projects have the potential to contribute to ongoing research, and we'll be happy to follow up with groups that have made interesting findings with regards to the specific tasks and in the data in general.

Challenges:

- The data requires a **normalization pass**. For example, some of the categorical values were inputted in free-text.
- **Multilabel problem** - the categorical value in Part 1 can have between 0-3 correct values.

Dataset:

The training dataset contains 65,798 entries (across train and test), each with 34 features, as detailed below in Section 3.4. It is supplied in the following files:

- **train.feats.csv**: 49,351 records in a csv file, where each row contains a single instance (single patient visit) with 34 features. The first row describes feature names.
- **test.feats.csv**: 16,447 records in a csv file, each containing a single instance (single patient visit) with 34 features. The first row describes feature names.

3.1 Part 1: Predicting Metastases

Given each visit characteristics, predict metastases sites (multi-label, multi-class categories).[2]

You are given the corresponding labels for the train set:

- train.labels.0.csv: 49,351 records, each line corresponds to the patient visit described in the same line index in train.feats.csv.

You are required to submit code to train a model and make predictions, along with its prediction for test features. It should be in the same format as the train labels, but with 16,447 entries, corresponding to predictions for the test features.

3.1.1 Submit

- model code
- **part1/predictions.csv**: 16,447 entries each corresponding to *test.feats.csv*, following the same format as *train.labels.0.csv*.

[1] https://www.cancer.org.il/template/publications.aspx?maincat=12

[2] https://en.wikipedia.org/wiki/Metastasis

3.1.2 Evaluation

Will be done using:

1. Micro average F1 score
2. Macro average F1 score

You are given the evaluation script with which you can test your output format validity: *evaluate_part_0.py* (run the file without parameters to get help).

Note: We suggest that before submitting you run your model on the train set and compare with the given gold predictions to verify your output format and calculate the empirical loss.

3.2 Part 2: Predicting Tumor Size

Given each visit characteristics, predict the diagnosed tumor size (in mm).

You are given the corresponding labels for the train set:

- train.labels.1.csv: 49,351 records, each corresponding with the features described in Section 3.4.

You are required to submit code to train a model and make predictions, along with its prediction for test features. It should be in the same format as the train labels, but with 16447 entries, corresponding to predictions for the test features.

3.2.1 Submit

- Your code and model.
- **part2/predictions.csv**: 16,447 entries each corresponding to *test.feats.csv*, following the same format as *train.labels.1.csv*.

3.2.2 Evaluation

Will be done using mean squared error.

You are given the evaluation script with which you can test your output format validity: *evaluate_part_1.py* (run the file without parameters to get help).

Note: We suggest that before submitting you run your model on the train set and compare with the given gold predictions to verify your output format and calculate the empirical loss.

3.3 Bonus Part 3: Unsupervised Data Analysis

Teach us something interesting about the data!

We've discussed in class various techniques for unsupervised analysis (e.g., clustering, dimensionality reduction, principal component analysis). In this task you are required to perform any of these methods to identify interesting trends in the data. For example (but not limited to):

1. What does the principal component patient look like?
2. Can you identify recurring patterns?
3. Can you provide interesting clustering of the data (k-means or spectral)?

As mentioned above, this data is part of ongoing research, so any finding here would be valuable, and we'd be happy to follow up with the teams that make the most insightful finding. Be creative!

Note:

Make sure to do this part using just *train.feats.csv*. This task is unrelated to the other two.

3.3.1 Submit

A short report outlining your findings, along with any supporting code. This can be achieved via python notebook, streamlit, or any other form you find suitable.

3.3.2 Evaluation

As this is an open ended task, we'll manually examine the results and test their soundness, validity and observations.

3.4 Feature description

1. 'Id-hushed_internalpatientid': Id of patients, we have 11623 different patients - 11623 values, Hash - long HEX
2. 'Form Name': Type of medical visit - 9 values, Hebrew words
3. 'Hospital': Hospital code - 4 values, float
4. 'User Name': User name of the doctor reporter - 154 values, <NUM>_Onco
5. 'Age': Age of patient - 11623 values, float
6. 'Basic stage': Carcinoma Basic stage - 4 Different stages (c, p , r, null)
7. 'Diagnosis date': Date of diagnosis - Datetime
8. 'Her2': Tumor marker test that determines the number of copies of the HER2 gene or the amount of HER2 protein in a cancer cell - various formats
9. 'Histological diagnosis': Histological diagnosis - 41 values, CONSTS (english caps)
10. 'Histopatological degree': Histopatological degree - 6 values - G1 to G4 + GX + Null
11. 'Ivi -Lymphovascular invasion': Whether the tumor invaded blood vessels or to lymph nodes - 17 values, various formats.
12. 'KI67 protein': The rate of cell multiplication in the tumor - a number - various formats.
13. 'Lymphatic penetration': How much the Lympha was penetrate - 5 values, L<x> + null
14. 'M -metastases mark (TNM)': Amount of existence of metastases - 6 values, M<x>
15. 'N -lymph nodes mark (TNM)': Amount of lymph invasion - 21 values, N<x>
16. 'T -Tumor mark (TNM)': Size of tumor in the first exam - 22 values, T<Something>
17. 'Margin Type': Tumor margin type - 3 values, hebrew consts
18. 'Nodes exam': How many Lymph nodes were examined - 42 values, float
19. 'Positive nodes': How many of Lymph nodes contained carcinoma metastases - 28 values, integers
20. 'Side': Breast side of tumor - 3, hebrew const
21. 'Stage': Stage of cancer - 17 values, english const
22. 'Surgery date1': Date of first surgery - Datetime
23. 'Surgery date2': Date of second surgery - Datetime
24. 'Surgery date3': Date of third surgery - Datetime
25. 'Surgery name1': Name of first surgery - 23 values, CONSTS (english caps)
26. 'Surgery name2': Name of second surgery - 18 values, CONSTS (english caps)
27. 'Surgery name3': Name of third surgery - 6 values, CONSTS (english caps)
28. 'Surgery sum': Number of surgeries: 3 values (1, 2, 3)
29. 'Tumor depth': Depth of tumor - 6 values, float
30. 'Tumor width': Width of tumor - 31 values, float
31. 'er': Tumor marker test that determines the sensitivity to estrogen of the cancer cell - various formats
32. 'pr': Tumor marker test that determines the sensitivity to progesterone of the cancer cell - various formats
33. 'surgery before or after-Actual activity': Name of surgery before diagnosis - 10 values, hebrew short string

34. ‘surgery before or after-Activity date’: Date of surgery before diagnosis - Datetime

4 Hackathon FAQ

Please review these questions carefully as we will not address them separately.

Q: We've started working on task A and after a day we decided to work on task B, can we do that?
A: Yes, you will be graded for one task only (either Waze or Breast Cancer prediction) but you may switch between them. If you submit 2 tasks (or parts of them) you'll get no grade.

Q: Can we use external python libraries?
A: Yes, as long as you follow the instructions above and make sure you provide a working `requirments.txt` and verify that these libraries do not require additional sources (`c++ libraries`, drivers, etc.).

Q: Can we use external data to solve the tasks?
A: NO. You are allowed to use the web for inspiration and code solutions but you are not allowed to augment the data provided with external data. Groups that will do so will get no grade.

Q: If we're able to run the code but it failed to run on evaluation time, do we get full grading?
A: NO. One of the demands of this hackathon is to be replicable. If we can't run your code, we can't verify your predictions and your submission is disqualified

Q: We need a place to sit quietly, where can we find it?
A: There's a list of available offices during the hackathon - please use them considerably - others would like to use them too.