**CRC**

Work directory */nas1/Work/Users/yaron/CRC/*; Subdirectories -

* *Barts*: see, for example, Scores\_07\_2020/commands for the process of checking and scoring a dataset from Barts. Eldan has already repeated the process once
* *Models*: validating and retraining the models.
  + *LGI\_ColonFlag-3.0 : Learning/commands.unmatched* includes the command used to retrain the model *crc\_model\_from\_unmatched.bin* which is the most up-to-date model (used for Barts, for example), though not the model exported to the AlgoMarker: *OrigModel/crc\_model* is ColonFlag 3.0. *Predictions* include symbolic links to all relevant prediction files (Maccabi cross-validation, internal and external validation set (note that we still don’t have the cancer registry for the external validation set), THIN (TRAIN=1), and KP north-west.
  + *LGI-Flag 3.1* model is in *LGI\_FLAG-3.1/Model/LGI-Flag-3.1.model* (same as /*nas1/Products/LGI-ColonFlag-3.0/FrozenVersions/LGI\_3.1.0.0/LGI-Flag-3.1.model*)
* *SingleTest.ReLearn* – relearning a model using only one CBC. JSON file is *single\_test\_model.json*. MHS & THIN predictions and performance analysis file inside
* KPNW repository is in */nas1/Work/CancerData/Repositories/KPNW/kpnw\_crc/*
* Algomarker directory - *MR/Projects/Shared/AlgoMarkers/LGI-ColonFlag-3.0/*
* Neumark + write crc as bin are in  */MR/Tools/Cancer/prepare\_cancer\_matrix/*
* */nas1/UsersData/yaron-internal/MR/Projects/Shared/CRC/Scripts/do\_bootstrap.py –* takes cancer registry (Cancer\_Location) + Directions + predictions and prepare file and then runs bootstrap\_app
* May be interesting - *NoiseSensitivity*/ and *Train\_Size\_Analysis*/

Frozen versions (configs, dll/so, *etc*.) are *in /nas1/Products/LGI-ColonFlag-3.0/FrozenVersions/*

Barak’s runs and results are in */nas1/Work/Users/Barak/hyper/freeze\_crc\_algomarker15/*

**LUCA**

Work directory */nas1/Work/Users/yaron/LUCA/*; Subdirectories –

* *Manuscript* includes analyses done per requests of the reviewers. See *commands*
* Following the discovery of the bias, *lc\_0921/FinalComparison* includes relevant models – see file *Models* for the list of models analyzed here and the command used for their analysis. Baseline is Tammemagi’s model, mes3/5 real\_matched models are learned on samples that are matched by available history (generated by …/lungcancer/scripts/match\_by\_real\_history.py) and GHS is the model derived on Geisinger data.
* Samples are in *lc\_0921/ Sampling.no\_gap/* Readme for generating samples - *lc\_0921/Readme*
* Ron’s original work is in *MR/Projects/Shared/AlgoMarkers/lungcancer/*
* The various models in *FinalComparison* were run by Alon in */nas1/Work/Users/Alon/LungCancer/outputs/models*
* Useful scripts –
  + *RunWindowMatchedBootstrap.py* forces the time-window on controls as well as cases (related to index-date for both)
  + *mPLCO\_train\_test.py* is a friendlier version of training+testing Tammemagi’s model

In GHS, work directory is /opt/medial\_sign/Work/Yaron/Lung;

* subdirectory *etl* includes scripts for generating repositories. See Flow.commands for commands for running Flow. Actual configs, files and dictionaries are in */opt/medial\_sign/Geisinger\_load\_lung\_jul21/*
* Two repositories were generated /opt/medial\_sign/Repositories/Geisinger/lung\_jul21 and lung\_jul21.filtered where inpatient lab tests are filtered out. Note that repository was generated to fit the various KP models, - thus the strange signals names (ICD9\_Diagnosis and ICD9\_Hospitalizations) and the artificial additions to the dictionaries.
* *CompareAllModels* runs multiple models in multiples scenarios (two repositories, sampling methods, etc.) in order to select optimal one. See *commands* . Quarterly sampling is selected on unfiltered repository and more detailed analysis given in *Selected\_mes3\_basic\_cv* (optimal model on GHS)
* Updated KP models (history matched mes3 and mes5) are also analyzed (on full repository) in *Selected\_mes3* and *Selected\_mes5*. Running collect\_sim.py to collect and compare simulation results, yields *collected\_sim*.
* Most up-to-date presentations shown to GHS is in SharePoint under Implementation/Geisnger/3 – Lung Cancer
* Cadence optimization in *CompareAllModels/Optimize\_Cadence*

**NLP**

Work directory is */nas1/Work/Users/yaron/NLP* ; code directory (git) is *Projects/Shared/NLP* ; most relevant work was done in subdirectory *Mimic* for entailment – for example, use notes to entail weather a given ICD code is the admission diagnosis

* *prepare\_for\_entailment* creates file for deriving/testing entailment models. It reads notes from MIMC (given in parameters “tabulated” + “types”), and signal to compare to (“signal”, usually DIAGNOSIS, given in ICD9 code).

The program checks texts and ‘signal’ (from structured data) that correspond to the same admission and generates samples of entailment (i.e., text + query (=ICD9 description) that appear together) and non-entailment (text + query that do not appear together, randomly selected according to the entailed queries distribution)

The program splits both ids and ICD9 codes into test and train sets and generates training and validation files of the format :

* + id, admin-Id ,time ,train-Id, query (ICD code desc.), label(0/1), text

“print\_level” tells the program the number of levels to climb up in the ICD9 hierarchy when generating an ‘entailment’ sample, and “check\_level” tells it how much to climb when assigning a ‘non-entailment’ sample (and checking that a randomly selected ICD9 code is indeed not related to the text/admission)

* in scripts – *apply\_entailment.py* and *train\_entailment.py* train and test various BERT models on the files generated by *prepare\_for\_entailment*. *train\_lstm\_on\_bert.py* and *apply\_lstm\_on\_bert.py* train and test LSTM model on the output of the entailment models handling several notes from the same admission (i.e. entailing the same ICD9 code)
* In *AllDiagnoses.Preprocessed –* 
  + *Commands –* examples of running *prepare\_for\_entailment* on various texts
* In *AllDignoses* –
  + new\_commands.txt – examples of running various train/apply scripts

**TrinetX**

Work directory for checking ColonFlag on TrinetX is /*nas1/Work/Users/yaron/TrinetX* . ETL scripts and various configs are in *MR/Tools/RepoLoadUtils/trinetx\_etl*

* data files for repository loading –
  + *FinalSignals.all\_encounters* – include all lab tests
  + *FinalSignals.amb\_encounters* – only ambulatory lab tests
  + *FinalSignals.am\_encounters.180d* – also excluding 180 days after admissions
  + *rep\_configs* – repository loading configuration and dictionary files
* Analysis files are in *Summary* (in particular the docx file). Some matrices (and the corresponding JSONs) used for the summary are in – *Analysis/AgeFix*
* Most of the relevant exploratory work was done on Jupyter notebooks:
  + “trinext – incidence by age”: check age dependent incidence of CRC for various repositories
  + “thin vs trinetx” and “thin vs trinetx II” – showing the behavior of THIN cases and controls compared to TrinetX cases and controls
  + “trinetx – controls hgb” – analysis of Hemoglobin distributions
  + “trinetx – dist” – some more distribution analyses
* See summary document and presentation in SharePoint under algoteam/TrinetX
* Samples generation : /*MR/Projects/Shared/CRC/TrinetX/generate\_samples.py and rerun\_generate\_samples.py*
* Training model: */nas1/Work/Users/yaron/TrinetX/LGI/Rerun/commands*

**Causal Inference**

Interesting solution is *MR/Projects/Shared/ CausalEffects/CausalEffectsUtils*. Projects of interest include –

* *generate\_realistic\_data* for generating toy-models data for evaluating various models
* *check\_toy\_models* for comparing the models on the toy data (including true generative model)
* *GenerativeModels* – a class for generative models (used both for generating data and evaluating true ITE)

The models tested by *check\_toy\_models* include cfr (counterfactual regression, by Uri Shalit), and “general” models with train/test scripts (e.g., Quasi-Oracle models with NN). Relevant scripts are -

* *CausalEffectScripts/CFR.py*
* *CausalEffectScripts/QuasiOracle.py*

**AlgoMedical Framework presentations**

See in SharePoint, under algoteam/Infrastructure Presentations