Lumiata Data Science - Take-home assignment

Diabetes is a highly prevalent and expensive chronic condition, costing about $330 billion to Americans annually. Most of the cost is attributed to the 'type-2' version of diabetes, which is typically diagnosed in middle age.

Today is December 31, 2016. A commercial health insurance company has contracted you to predict which of their members are most likely to be newly-diagnosed with type-2 diabetes in 2017. Your goal is to featurize their data, train and optimize a predictive model, and explain your results and approach. (Note: "newly-diagnosed" means members who were **NOT previously coded** with diabetes prior to 2016-12-31, inclusive).

The data provided are real patient claims records from a large insurance company, appropriately de-identified. The data sets have already been split into training and test sets ('train.txt' & 'test.txt'). The proportions of members are about 70% train and 30% test.

Each line in both text files is a patient record, represented as a json string. The health record is parameterized by a set of encounter dates in a YYYY-MM-DD format. The structure of each patient json is as follows:

- 'bday' - patient date of birth in YYYY-MM-DD format
- 'is_male' - True = Male, False = Female
- 'patient_id' - de-identified patient id (each patient is given a unique value)
- 'resources' - dictionary of encounter_date → list of codes (described below)
- 'observations' - dictionary of encounter_date → list of dictionaries (described below)
- 'tag_dm2' - indicates date of first type-2 diabetes diagnosis - will either have a YYYY-MM-DD date or be an empty '' string; this information will be censored from the holdout set. (described above)
- 'split' - indicates a member is in the 'train' or 'test' set; information beyond 2017-01-01 has been **removed** from test.txt.

Each patient record has a key 'tag_dm2', whose value is **either** a 'YYYY-MM-DD' date string indicating the date of first code of a diagnosis of diabetes, **or** an empty string '' (indicating no diabetes in their record).

Your task is to predict each test set member's probability of being newly-diagnosed with diabetes in 2017. Information for each test set member's health record beyond 2017-01-01 has been removed; the true diabetes status of the member is hidden from you and will be used by Lumiata's data science team to evaluate your solution.

You should cohort your data (i.e construct the response variable) in the training set according to the following definitions (check your work with the training set counts given below for each definition):

- A ‘claim’ is someone whose ‘tag_dm2’ date is between 2017-01-01 and 2017-12-31, inclusive (training set count of ‘claim’ = 3410) - the response for these members is a ‘1’
- A ‘never-claim’ is someone whose ‘tag_dm2’ date is **either** after 2017-12-31, exclusive, **or** is an empty string ‘’ (training set count of ‘never-claim’ = 70110) - the response for these members is a ‘0’
- A ‘prior’ is someone whose ‘tag_dm2’ date is **before** 2017-01-01, exclusive - typically ‘priors’ are filtered out of the matrix before training. You may include these people in training, but keep in mind they will be filtered out of ‘test’ when we evaluate your solution.

Each patient record also has two keys describing their health history - ‘resources’ & ‘observations’.

The ‘resources’ key specifies the diagnoses, medications, and procedures that were noted/prescribed/performed at each doctor’s visit - these are represented by different coding systems (icd9/10, rxnorm, cpt, respectively.) Each encounter date in the ‘resources’ key is mapped to the corresponding list of codes issued at that doctor’s visit.

```
In [3]: pprint(ls[0])
 u'patient_id': u'pat_0',
 u'resources': {u'2014-01-13': [],
                u'2014-02-18': [],
                u'2014-03-21': [],
                u'2014-04-30': [],
                u'2014-05-21': [u'cpt_99213',
                                u'icd9_V85.23',
                                u'cpt_87210',
                                u'icd9_789.03',
                                u'cpt_87591',
                                u'cpt_87491',
                                u'icd9_616.10'],
                u'2014-06-05': [],
                u'2014-07-02': [u'cpt_99213', u'icd9_789.03', u'icd9_278.02'],
                u'2014-07-10': [u'cpt_72195', u'icd9_789.09'],
                u'2014-07-30': [],
                u'2014-08-06': [u'cpt_73510', u'icd9_789.03', u'cpt_99213'],
                u'2014-08-20': [u'icd9_564.00',
                                u'cpt_99213',
                                u'cpt_74000'
```

The codes have the format <system>_<code>. For instance, ‘icd9_272.0’, which corresponds to high cholesterol:

http://www.icd9data.com/2015/Volume1/240-279/270-279/272/272.0.htm

Note - encounter dates in ‘resources’ can sometimes have no codes in the code list!

The ‘observations’ key specifies the lab tests that were completed - each encounter date is mapped to a list of dictionaries, each of which has the following keys:

- ‘code’ - the ‘loinc’ code corresponding to the lab test
- ‘interpretation’ - whether the lab was ‘H’ for high, ‘L’ for low, ‘N’ for normal, or ‘A’ for abnormal

- ‘value’ - the value extracted from the lab

```
In [3]: pprint(ls[0])
{u'bday': u'1959-10-10',
 u'is_male': False,
 u'observations': {u'2014-05-21': [{u'code': u'loinc_24111-7',
                                    u'interpretation': None,
                                    u'value': None},
                                   {u'code': u'loinc_6356-0',
                                    u'interpretation': None,
                                    u'value': None}],
                   u'2014-10-30': [{u'code': u'loinc_20405-7',
                                    u'interpretation': u'L',
                                    u'value': u'0.0'},
                                   {u'code': u'loinc_21000-5',
                                    u'interpretation': u'N',
                                    u'value': u'13.0'},
                                   {u'code': u'loinc_2514-8',
                                    u'interpretation': None,
                                    u'value': None},
                                   {u'code': u'loinc_25428-4',
                                    u'interpretation': None,
                                    u'value': None},
```

For instance, the lab could have been a blood glucose test ‘loinc_2345-7’, whose value may have been 130, and hence whose interpretation would be ‘H’ (a cut-off for high blood glucose is 106:
https://s.details.loinc.org/LOINC/2345-7.html?sections=Comprehensive)

Note - the values in the ‘interpretation’ and ‘value’ keys can sometimes be ‘None’!

The keys in the ‘resources’ and ‘observation’ dictionary correspond to the encounter date with the doctor. All dates are formatted as string in YYYY-MM-DD format, e.g. “2016-04-30”.

The format of the file you submit to us should be a csv file, formatted as ‘<your_name_here>_dm2_solution.csv’. We should be able to read in your solution using pandas as follows:

```
In [50]: df = pd.read_csv('rohun_dm2_solution.csv')
         df.head(10)
```

Out[50]:

	patient_id	dm2_prob
0	pat_37603	0.989843
1	pat_61407	0.978793
2	pat_86010	0.976513
3	pat_11058	0.956268
4	pat_71195	0.944390
5	pat_88537	0.940543
6	pat_1130	0.911845
7	pat_76344	0.887259
8	pat_22566	0.873087
9	pat_77241	0.837598

for each test set patient_id.

You will also need to submit the python code you ran to execute your analysis.

We will judge the model performance using a precision-recall curve, and look at a couple choice metrics, like precision@20 recall, and PRAUC (read about it here: http://scikit-learn.org/stable/auto_examples/model_selection/plot_precision_recall.html) and compare to our own internal model results on this data set.

Additionally, your code will be judged along the following axes:

1. Experimental setup/design
2. Feature design/feature selection
3. Model selection/tuning
4. Model performance evaluation (i.e how well it does along the metrics above)
5. A short write-up (2-3 paragraph) of your approach - be sure to include the following:
    - Top 5 most important features for predicting diabetes.
    - Briefly describe the model you used for training.
    - Explain how you optimized your model, and what performance metrics you optimized for in training.
    - How did you prevent overfitting?

Note - model selection/tuning isn't the most important criteria here! We will primarily evaluate your approach using your output performance. However, consideration will be given for well thought-out #1-5, as well as performant, documented code.

We should be able to run your code using only the train.txt & test.text files as inputs, so please **don't** keep any local paths in your code!

You are free to use any resources you like in solving this problem. Please return your solution within a few days of starting it.

Thanks, and best of luck!

Lumiata Data Science