Unsupervised Phenotyping via Tensor Factorization: A Case Study in Diabetes

Xingchi Li, MSCS[1], Yanxin Ye, MSA[2], Hung Yi Li, MSCS[3], Xinze Wang, MSA[4]
[1]Georgia Institute of Technology, Atlanta, Georgia, USA

Abstract

Phenotyping is the process of clustering patients with similar clinically meaningful characteristics and define patient groups. Tensor factorization is an effective way to phenotype electronic health records (EHR). TASTE[1] is an unsupervised phenotyping approach based on tensor factorization which tackles the problem of jointly modeling static and dynamic feature of patients. It has been proved successful in many ways including predicting heart failure patients. We conduct a case study in Diabetes phenotyping to demonstrate the practical application of TASTE. We experiment on varying sample size, observation window, time unit, different categorical-dummy variable transformation, identical static features, and number of phenotypes to explore the best practice of phenotyping. Using phenotypes generated by TASTE, a logistic regression classifier is able to predict whether a patient will be diagnosed with diabetes or not in a fast and effective way. TASTE was firstly implemented in Matlab. Since Matlab is not commonly accessible to the public, we implement TASTE in Python3 environment to open-source this method.

1 Introduction

Electronic health records (EHR) exhibit characteristics of high-dimension, messiness, and billing-orientation. Real-life EHRs usually consist of both static and dynamic features for each patient. One of the effective tools for phenotyping is Temporal And Static TEnsor factorization (TASTE)[1]. Proposed by Afshar et al. (2019), TASTE jointly models both temporal and static data from EHRs to extract phenotypes. It helps us understand patients' non time-varying characteristics such as demographics and time-varying characteristics such as diagnoses and corresponding temporal evolution synchronously.

In previous research, TASTE has been successfully used in heart failure phenotyping. We apply the algorithm to a case study of predicting Diabetes. Our approach employs Propensity Score Matching (PSM) to match cases and controls, and experiments on sample sizes, observation window lengths, time units, different categorical-dummy variable transformation, identical static features, and R to explore their impact on predicting powers.

The phenotypes generated based on these 6 experiment scenarios will be run on logistic regression classifiers for diabetes prediction. The model performance will be evaluated with area under the ROC curve (AUC) and Precision-Recall AUC (PRAUC). 5-fold cross-validation processes are planned be conducted following 5 training steps.

The main findings that contradict to our knowledge are three folds. First, the most suitable range of R values for TASTE fitting is subject to sample and feature compositions of the data sets. Predictive power is not necessarily higher as R increases. Second, the most efficient training-testing split also depends. Predictive power could decrease as training sample size increases. Third, changing the way to count clinical visits from by date to by week for dynamic feature construction could facilitate unexpected good prediction performance.

For the following sections, we will first review the previous literature in the section 2, including the description of the TASTE algorithm. In the section 3, we introduce the data source and the data preprocessing steps. Then, we illustrate the experiment details including preprocessed input data description (4.1), evaluation metrics (4.2), experiment designs (4.3), predictive model training steps (4.4), and system settings (4.5). The experiment results will be provided in the section 5. Finally, we will briefly discuss the results of TASTE and predictive models and the constraints in the section 6 and summarize in the section 7.

2 Background and Related Work

2.1 Phenotyping EHRs

Phenotype is the observable physical or biochemical expression of a specific trait in an organism, such as a disease, stature, or blood type, based on genetic information and environmental influences. Some recent studies have focused on unsupervised EHR-derived phenotyping, which typically does not require labor intensive supervision from medical experts. Among them, tensor factorization is an emerging and promising field.

Marble[2] is a sparse non-negative tensor factorization approach which deals with count data. Marble decomposes the tensor into two parts: a bias tensor and an interaction tensor. The bias tensor represents the common characteristics among all patients, while the interaction tensor decomposes the tensor into R different phenotypes. Marble fits a Poisson distribution to the count data and minimizes the KL divergence. Rubik[3] uses the Alternating Direction Method of Multipliers (ADMM)[4] algorithm to deal with binary tensors and minimize the Euclidean distance of the real and predicted tensors. Rubik, like Marble, decomposes the tensor into bias and interaction parts. Phenotypes extracted by Rubik are often more meaningful than Marble because of Orthogonality Constraint, which produces more distinct phenotypes by reducing the overlapping components. Also, Rubik incorporates some predefined knowledge to the objective function to improve the interpretability of phenotypes. Like Marble, Limestone[5] is a nonnegative tensor factorization model to achieve highthroughput phenotyping from EHR data. Limestone captures the interaction of diagnoses and medications among patients. The resulting tensor factors are reported as phenotype candidates that automatically reveal patient clusters on specific diagnoses and medications. Using the proposed method, multiple phenotypes can be identified simultaneously.

To analyze a set of variables across a set of subjects whose observations do not align naturally, PARAFAC2 model was proposed to produce unique solutions for dense data sets[6]. Unique solutions ensure that the pursued solution is not an arbitrarily rotated version of the actual latent factors. Recently, SPARTan provided a scalable approach for PARAFAC2 modeling on large and sparse data[7]. A specialized Matricized-Tensor-Times-Khatri-Rao-Product (MTTKRP) was designed to efficiently decompose the tensor in terms of both speed and memory. Another challenge for PARAFAC2 is to introduce various modeling constrains for interpretable temporal modeling. COPA[8] is a constrained PARAFAC2 model which applies non-negativity, smoothness, and sparsity constraints to the resulting factor matrices.

2.2 TASTE Model

Temporal And Static TEnsor factorization (TASTE)[1] combines the PARAFAC2 model[6] with non-negative matrix factorization to jointly model both static and temporal information to extract phenotypes.

To fit the TASTE model, both temporal and static features for K patients are required input data. Temporal features $\boldsymbol{X_k}$ indicates the medical features for different clinical visits in matrix $\boldsymbol{X_k} \in \mathbb{R}^{I_k \times J}$ for patient k where I_k is the number of clinical visits and J is the total number of medical features. Static features $\boldsymbol{A}$ represents the features such as age, sex, and race and are recorded in $\boldsymbol{A} \in \mathbb{R}^{K \times P}$ where P is the number of static features.

2.2.1 Objective Function

The TASTE model has the following objective function:

$$f(\{\boldsymbol{Q}_k\}, \boldsymbol{H}, \{\boldsymbol{U}_k\}, \boldsymbol{V}, \{\boldsymbol{S}_k\}, \boldsymbol{F}) = \sum_{k=1}^{K}(\frac{1}{2}\|\boldsymbol{X}_k - \boldsymbol{U}_k\boldsymbol{S}_k\boldsymbol{V}^T\|_F^2) + \frac{\lambda}{2}\|\boldsymbol{A} - \boldsymbol{W}\boldsymbol{F}^T\|_F^2 + \sum_{k=1}^{K}(\frac{\mu_k}{2}\|\boldsymbol{U}_k - \boldsymbol{Q}_k\boldsymbol{H}\|_F^2).$$

To optimize the objective function, we need to update $\{\boldsymbol{Q}_k\}, \boldsymbol{H}, \{\boldsymbol{U}_k\}, \boldsymbol{V}, \{\boldsymbol{S}_k\}$ and $\boldsymbol{F}$ iteratively.

Fed with input data, the TASTE model generates a set of $\boldsymbol{R}$ phenotypes composed of temporal and static features presenting as factor matrices $\boldsymbol{V}$ and $\boldsymbol{F}$ respectively, personalized phenotype scores $\boldsymbol{S}_k$ and temporal phenotype evolution $\boldsymbol{U}_k$ for each patient.

2.3 Solutions for TASTE

The original problem is non-convex. TASTE utilizes Block Coordinate Descent framework[9] to reformulate the objective function into simpler sub-problems. In each iteration, $\{\boldsymbol{Q}_k\}$ is updated based on Orthogonal Procrustes problem[10], $\{\boldsymbol{H}\}$ can be solved by least square solvers. For $\{\boldsymbol{U}_k\}, \boldsymbol{V}, \{\boldsymbol{S}_k\}$, the objective function is reformulated so that factor matrixs are instances of Nonnegativity-constrained Least Squares(NNLS) problem. A novel approach, block principal pivoting method[11] has been proven effective in solving each NNLS sub-problem.

2.3.1 Existing Phenotype Fitting on New Data

The existing phenotypes are applied to the new data in two situations. First, after the discovery of phenotypes closely associated with diabetes with mapping TASTE on training cases, the personalized phenotype scores $\boldsymbol{S}_k$ and temporal phenotype evolution $\boldsymbol{U}_k$ for training controls have to be calculated for further diabetes predictive modeling. Second, to conduct cross-validation on diabetes predictive models, the personalized phenotype scores $\boldsymbol{S}_k$ and temporal phenotype evolution $\boldsymbol{U}_k$ for testing cases and controls have to be calculated as well.

When calculating the personalized phenotype scores $\boldsymbol{S}_k$ and temporal phenotype evolution $\boldsymbol{U}_k$ for new data, we fix the existing phenotype definitions $\boldsymbol{V}$ and $\boldsymbol{F}$. In other words, we project new patients into $\boldsymbol{H}$, $\boldsymbol{V}$, and $\boldsymbol{F}$ by optimizing $\{\boldsymbol{Q}_k\}$, $\{\boldsymbol{S}_k\}$, and $\{\boldsymbol{U}_k\}$ for the following objective function:

$$f(\{\boldsymbol{Q}_k\},\{\boldsymbol{U}_k\},\{\boldsymbol{S}_k\}) = \sum_{k=1}^{K}(\frac{1}{2}\|\boldsymbol{X}_k - \boldsymbol{U}_k\boldsymbol{S}_k\boldsymbol{V}^T\|_F^2) + \frac{\lambda}{2}\|\boldsymbol{A} - \boldsymbol{W}\boldsymbol{F}^T\|_F^2 + \sum_{k=1}^{K}(\frac{\mu_k}{2}\|\boldsymbol{U}_k - \boldsymbol{Q}_k\boldsymbol{H}\|_F^2).$$

3 Dataset Description and Feature Engineering

3.1 Data Source and Features

The data set CMS 2008–2010 Medicare Data Entrepreneurs' Synthetic Public Use File (DE-SynPUF)[1] is synthesized from 5% of real claims data of Medicare beneficiaries in 2008 with variable reduction, suppression, substitution, imputation, date perturbation, and coarsening to lower the chance of re-identification. The data structure are similar to the real claims that we can use it for data application purposes.

DE-SynPUF consists of five types of files – Beneficiary Summary, Inpatient Claims, Outpatient Claims, Carrier Claims, and Prescription Drug Events. We extract patients' demographic information such as date of birth, sex, and race as well as information of chronic diseases from Beneficiary Summary to form the static features. For temporal features, we extract claim date and ICD-9 diagnosis codes and procedure codes from Inpatient Claims, Outpatient Claims, and Carrier Claims.

3.2 Data Preprocessing

To build temporal and static tensors, we apply the following preprocessing steps:

- Remove patients with fewer than 5 visits.
- Remove sparse features (features with less than 5 patients).
- Manage the order of I_k based on the claim date and adjust the time unit such as $by\ date$ or $by\ week$.
- Convert rank-1 temporal tensors into binary data.
- Divide patients into four age groups based on percentile, and create categorical variables accordingly.

3.3 Cohort Construction

We identify the case cohort by looking up ICD-9 diagnosis codes defining diabetes – specifically 249 and 250 – appearing at least once in a patient's visits, and assign the date first diagnosed diabetes as the index date. In order to satisfy the requirement of observation window, all case patients have at least 2-year records before index date.

Patients who were neither diagnosed with diabetes nor had history of diabetes were potential candidates for controls. To have a fair comparison between the cases and controls and avoid selection bias among cohorts, we take a reference of the matching features other medical paper adopted[12]. We include date of birth, sex, race and chronic diseases as

[1]`http://www.cms.gov/Research-Statistics-Data-and-Systems/Downloadable-Public-Use-Files/SynPUFs/DE_Syn_PUF`

static features to map control patients to each case patient. The matching technique we adopt is Propensity Score Matching (PSM). There is a ready Python package $\boldsymbol{pymatch}$[13] which supports the computation of PSM. The matching process yields an overall ratio of 7 controls per case. Each control patient is assigned to his most similar case patient, and the index dates of controls are set as the same date as their corresponding case.

3.4 Diagnosis and Procedure Code Transformation

To reduce the amount of features and to have more interpretable medical concepts, we transform ICD-9 diagnosis codes and procedure codes to Clinical Classification Software (CCS level 3). After the conversion, the total number of diagnosis and procedure features reduces from 14,214 to 471.

4 Experiment Design and Setting

4.1 Feature Description

After applying the preprocessing steps, we create a data set with 36,336 patients where 4,538 of them are cases and 31,789 are controls. We extract 277 diagnosis codes and 194 procedure codes for dynamic features and 13 static features from a 2-year observation window. To simplify the problem, prediction window length is set to zero. The average number of visits (length of I_k) is 29.4.

4.2 Model Evaluation and Metrics

To evaluate the performance of classification models, Area Under the ROC Curve (AUC) is one of the most commonly used metrics. Due to the potential imbalance of case-control sampling, we examine both AUC and Precision-Recall AUC(PRAUC).

(Due to time constraint and extremely long time on TASTE fitting/projection, we only run each experiment once without doing cross-validation, so that we could not provide standard deviation at this moment.)

4.3 Experiment Designs

We are interested in answering six questions derived from four aspects of TASTE model fitting and their corresponding predictive power. First, how would the training sample size influence the phenotype extraction and the associated prediction? Second, how would the observation window length affect? Third, what are the influences of feature manipulations? Specifically, for feature manipulations, we will inspect the effects of varying time unit, changing categorical-dummy variable transformation, and disregarding static features. Finally, we would like to see how varying R, the number of phenotypes, affect the predicting results.

In each experiment, we compare with the baseline having the settings of 720-day observation window, by date dynamic features, and N-to-N categorical-dummy variable transformation on static features.

4.3.1 Varying Training Sample Size

In machine learning problems, we always try to find the efficient number of training sample size to balance the model performance and sampling costs. Typically, increasing training samples yield better performance. The improving performance reaches a plateau when there are enough training data that new-added data do not add much useful information.

We conduct the experiment of varying training sample size on the baseline data set. The training-testing sample sizes are split into 0.2-0.8, 0.4-0.6, 0.6-0.4, and 0.8-0.2. For each split, we fit the TASTE model only on training cases to extract the phenotypes and fit the predictive model on both training cases and training controls with their phenotype scores. The resulting predictive model is then used to predict testing cases and controls to generate the AUC and PRAUC scores for comparison.

4.3.2 Varying Observation Window

In the medical context, the length of observation window is essential to the prediction of disease occurrence. We expect to see better prediction performance as a patient has a longer observation window. It is intuitive since patients would typically have more medical records for our reference in a longer observation window.

We conduct the experiment of varying observation window on the data sets in 180-day, 360-day, 540-day, and 720-day observation windows. The 720-day observation window one is exactly the baseline data set. The other three data sets only differ in their observation windows with other conditions fixed. Due to the adjustment of window length, we lose some patients with no visits in the observation window. For each data set, we split the training and testing samples into 0.8-0.2. For each split, we follow the same procedures described in the previous experiment.

4.3.3 Varying Time Unit

Some patients might have intensive health care visits during a short period, while some other patients might have sparse health care visits across a long period. We believe this information also sends out useful information in disease prediction. Thus, other than constructing dynamic features such as diagnoses by date in the baseline data set, we also consider an alternative construction by week. Both data sets have binary values in diagnosis or procedure codes but of a different representation of time unit. We conduct this experiment on two data sets with the same split ratio and procedures described in the previous experiment.

4.3.4 Fitting TASTE with N-to-(N-1) Categorical-Dummy Variable Transformation

We currently adopt an N-to-N categorical-dummy variable transformation on static features. In other words, we have 4 dummies to represent 4 age groups and 4 dummies for 4 races. The purpose is to extract representative phenotypes for every group in the samples. However, traditionally in categorical analysis, (N-1) dummies are used to represent a categorical variable of N categories to avoid high correlation among variables due to multi-collinearity. Also, (N-1) dummies are enough to represent the presence of N categories in that a subject with straight-0 in (N-1) dummies belongs to the category not explicitly shown in the column names. Therefore, we would like to inspect whether adopting N-to-(N-1) categorical-dummy variable transformation would lead to information loss. Two columns – age over 82 and other races – are dropped from the original baseline data set. Following the same split ratio and procedures described in the previous experiment, we compare the experiment results between the baseline and the dropped one.

4.3.5 Fitting TASTE without Static Features

Although the TASTE model is designed for the combination of static and dynamic features, we would like to see how it applies to merely static features, just like in COPA. Based on the baseline data set, we discard all the static features and make up a new column with a value 1 for every patient. Following the same split ratio and procedures described in the previous experiment, we compare the experiment results between the baseline and the identical-static-feature one.

4.3.6 Varying R

For the above 5 experiment scenarios, we run the experiments with R, the number of extracted phenotypes, ranges from 5 to 40 with 5 per step. As found in previous literature, a higher R usually yields a better prediction performance. The underlying reason is that more phenotypes extracted could represent more information contained in the original features. We would like to see if the previous conclusions apply in our experiments.

4.4 Predictive Model Training Steps

To generate robust statistics for the comparison of the predictive power of classification model under the above mentioned experimental scenarios, we plan to run 5-fold cross-validation processes with 80% training and 20% testing for each fold. (As mentioned in 4.2, we do not run 5-fold cross-validation at this moment but only run each experiment once due to time constraint and extremely long time on TASTE fitting/projection.) We adopt the logistic regression classifier as the algorithm of the base model for the focal 2-class classification problem – diabetes prediction. For each experiment, we will follow the 5 steps below in order:

- Apply TASTE on training cases and extract phenotypes $\boldsymbol{V}$ (temporal) and $\boldsymbol{F}$ (static). Calculate personalized phenotype scores $\boldsymbol{S}_k$ for these cases.
- Assign existing phenotypes $\boldsymbol{V}$ and $\boldsymbol{F}$ to training controls and calculate personalized phenotype scores $\boldsymbol{S}_k$ for these controls.
- Train logistic regression classifier on personalized phenotype scores $\boldsymbol{S}_k$ of training cases and controls.
- Assign existing phenotypes $\boldsymbol{V}$ and $\boldsymbol{F}$ to testing cases and controls and calculate personalized phenotype scores $\boldsymbol{S}_k$ for them.
- Predict AUC/PRAUC for testing cases and controls based on the logistic regression classifier on personalized phenotype scores $\boldsymbol{S}_k$.

4.5 System Setting

Data preprocessing and feature engineering steps were done by Hadoop Hive in Windows Docker Desktop.

In experiment steps, we work on Python 3.6.5 environment, and the RAM of the laptop is 16.0 GB. The code repositories we use for our baseline and experiment are scikit-learn[14] and pymatch[13]. And we set random state as 0 for all models.

5 Experiment Results

Experimental variables have significant impact on the quality of phenotyping. In previous research, the settings of experimental variables rely on physicians' domain knowledge. Lacking domain knowledge, we experiment on training sample size, length of observation window, time unit, categorical transformation method, the presence of static features, and number of phenotypes to study the impact of experimental variables settings.

5.1 TASTE Fitting and Projection

We set the RMSE tolerance to be 10^{-4}. As observed in the figures below, as we vary R, RMSE sometimes converge but sometimes diverge or rebound during the whole fitting process.

From figure 1 to figure 4, we found that there is no pattern in TASTE fitting time as R varies though we expect to see longer fitting time as R increases. Because of divergence and rebound, R=15 and R=25 take longer than R=35. To our surprise, R=35 takes shorter time to converge than R=5.

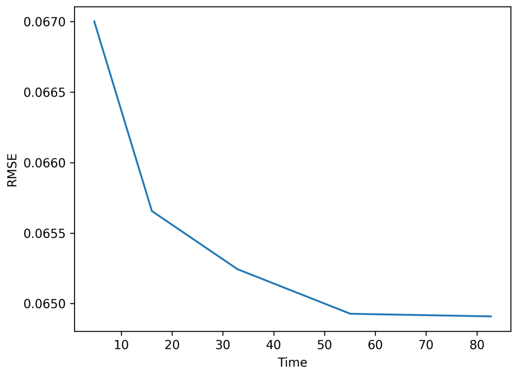

Figure 1: RMSE Time Base R = 5 Train Case

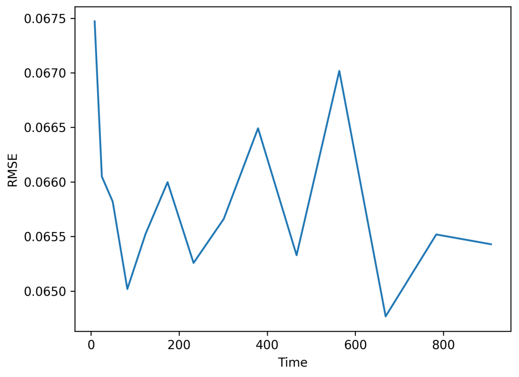

Figure 2: RMSE Time Base R = 15 Train Case

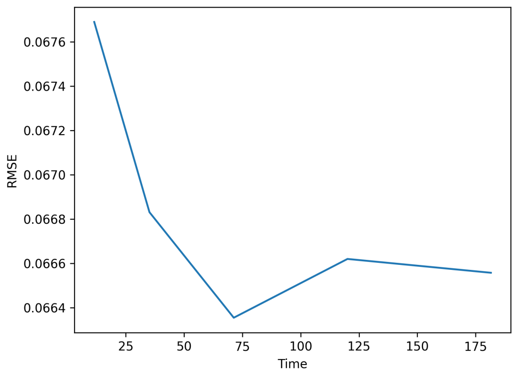

Figure 3: RMSE Time Base R = 25 Train Case

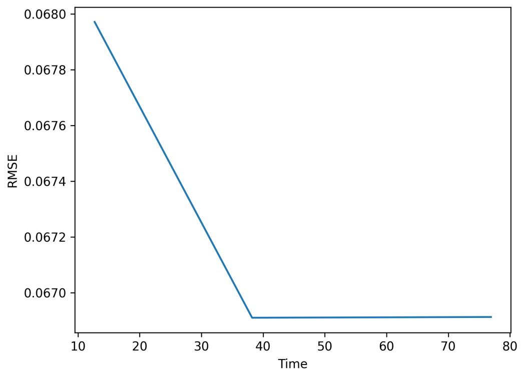

Figure 4: RMSE Time Base R = 35 Train Case

From figure 5 to figure 8, we compare the projection time with fitting time from figure 1 to figure 4 respectively. We can see that projection time is not necessarily shorter than fitting time though some parameters are fixed and probably less parameters need to be optimized in the TASTE projection model. One possible reason is that control patients are about 7 times more than case patients, so that projection is run on more patients than fitting is.

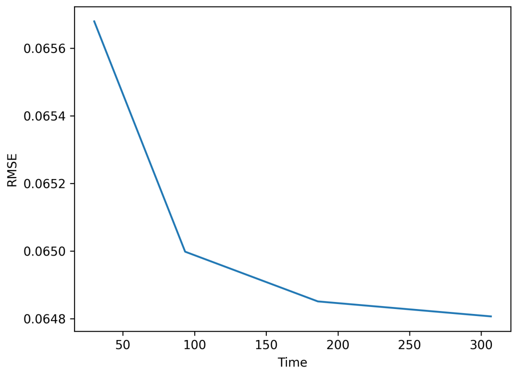

Figure 5: RMSE Time Base R = 5 Train Control

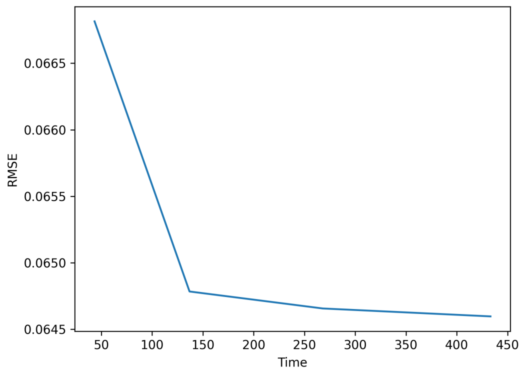

Figure 6: RMSE Time Base R = 15 Train Control

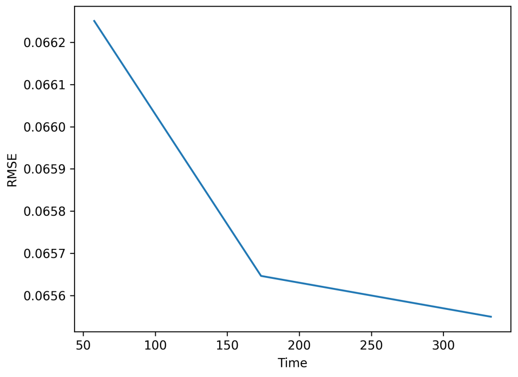

Figure 7: RMSE Time Base R = 25 Train Control

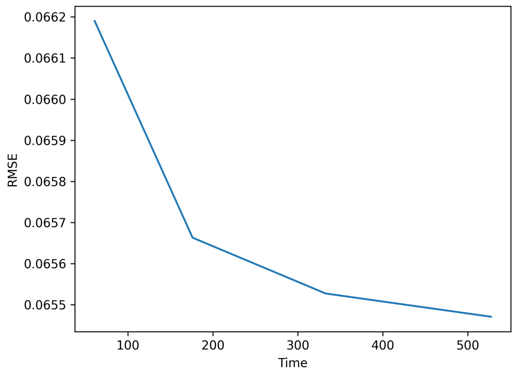

Figure 8: RMSE Time Base R = 35 Train Control

5.2 Predictive Modeling

5.2.1 Varying Training Sample Size

In this experiment, we vary the training sample size of case/control patients to see whether increasing training sample size improves the model performance. We observe that when R is relatively small, larger training samples do not necessarily generate higher AUC and PRAUC. As R increases, the differences narrow. According to the current result that higher R does not yield higher AUC/PRAUC, we would suggest using lower training split ratio to apply to TASTE with lower R.

Table 1: AUC - Metrics Size

R	5	10	15	20	25	30	35	40
0.2	0.647	0.518	0.550	0.469	0.518	0.488	0.000	0.415
0.4	0.619	0.547	0.517	0.531	0.394	0.396	0.429	0.395
0.6	0.591	0.475	0.497	0.437	0.449	0.456	0.438	0.392
0.8	0.596	0.415	0.518	0.493	0.540	0.411	0.436	0.384

Table 2: PRAUC - Metrics Size

R	5	10	15	20	25	30	35	40
0.2	0.216	0.141	0.168	0.134	0.147	0.121	0.000	0.106
0.4	0.199	0.156	0.159	0.153	0.103	0.099	0.105	0.101
0.6	0.186	0.139	0.140	0.120	0.125	0.130	0.119	0.103
0.8	0.190	0.100	0.157	0.147	0.142	0.099	0.119	0.096

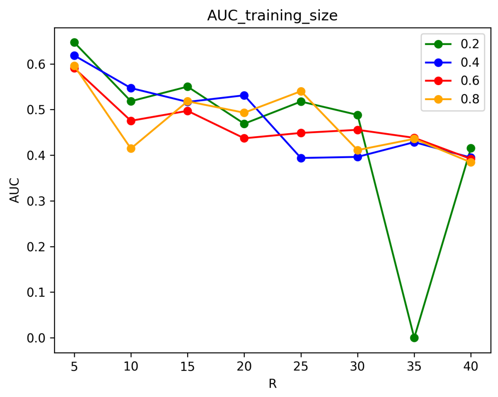

Figure 9: AUC - Varying Training Sample Size

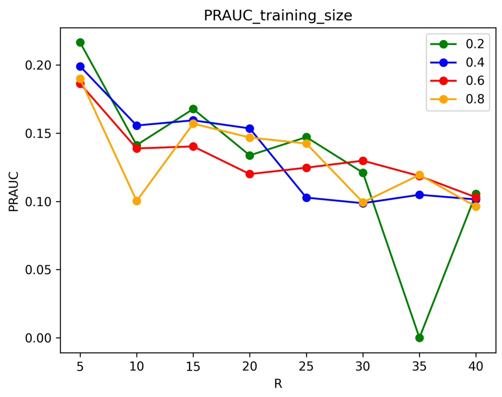

Figure 10: PRAUC - Varying Training Sample Size

5.2.2 Varying Observation Window

In this experiment, we vary the length of observation window in 180, 360, 540, and 720 days to see whether longer observation window improves the model performance. We observe that when R is relatively small, the longest observation window performs the best, while the others perform similarly. When R is relatively large, the differences narrow. One possible explanation is that larger R can offset the disadvantage of lacking medical records in short observation windows.

Table 3: AUC - Metrics Observation Window

R	5	10	15	20	25	30	35	40
180	0.521	0.509	0.460	0.480	0.484	0.445	0.450	0.456
360	0.514	0.528	0.540	0.472	0.424	0.507	0.415	0.472
540	0.501	0.519	0.428	0.434	0.391	0.435	0.407	0.510
720	0.611	0.619	0.455	0.475	0.457	0.544	0.000	0.464

Table 4: PRAUC - Metrics Observation Window

R	5	10	15	20	25	30	35	40
180	0.178	0.169	0.141	0.155	0.150	0.133	0.134	0.143
360	0.150	0.164	0.156	0.133	0.117	0.135	0.116	0.123
540	0.136	0.159	0.114	0.115	0.101	0.113	0.107	0.143
720	0.182	0.210	0.114	0.135	0.122	0.145	0.000	0.118

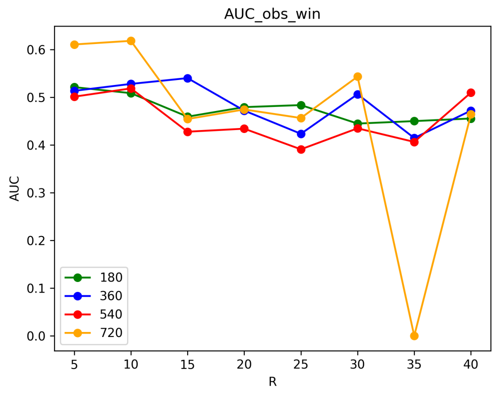

Figure 11: AUC - Varying Observation Window

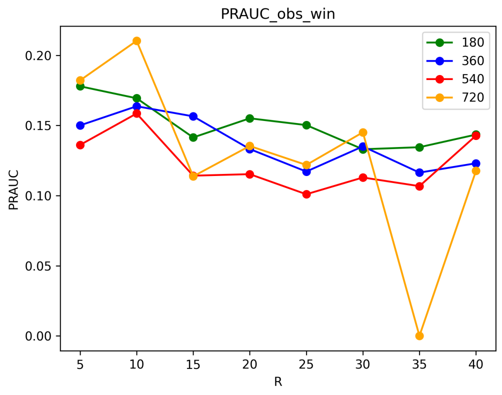

Figure 12: PRAUC - Varying Observation Window

5.2.3 Varying Time Unit

In this experiment, we vary the time unit of I_k $by\ date/visit$ and $by\ week$ to see the difference in model performance. We observe that except for relatively small or large R, constructing dynamic features by week outperforms by date construction. One reasonable explanation is that intensive health care visits in a very short period like in a week should only be counted as once, because the causes, symptoms, diagnoses, procedures, medications, and other medical related features might aim at treating the same one medical accident occurrence. Or maybe some patients tend to be more nervous than others so that they pay several clinical visits when the symptoms do not disappear immediately after one visits. If we count them as multiple times, the severity might be amplified.

Table 5: AUC - Metrics Varying Time Unit

R	5	10	15	20	25	30	35	40
base	0.601	0.465	0.395	0.440	0.478	0.451	0.463	0.537
week	0.568	0.588	0.471	0.461	0.469	0.475	0.409	0.489

Table 6: PRAUC - Metrics Varying Time Unit

R	5	10	15	20	25	30	35	40
base	0.204	0.138	0.102	0.119	0.127	0.117	0.122	0.142
week	0.177	0.193	0.131	0.126	0.128	0.135	0.109	0.132

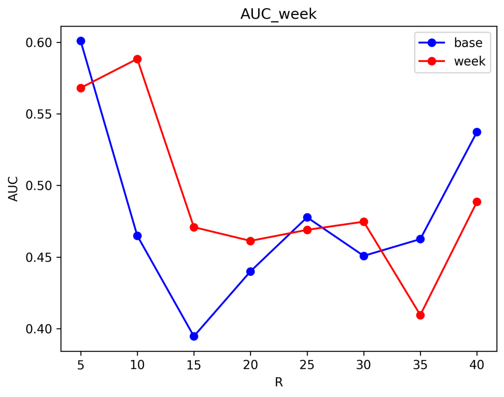

Figure 13: AUC - Varying Time Unit

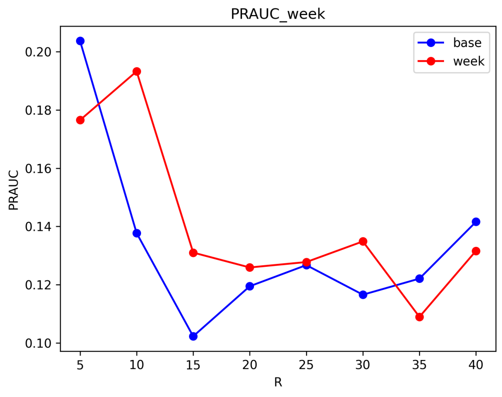

Figure 14: PRAUC - Varying Time Unit

5.2.4 Fitting TASTE with N to (N-1) Categorical-Dummy Variable Transformation

In this experiment, we fit the TASTE model with typical N-to-(N-1) categorical-dummy variable transformation instead of N-to-N categorical-dummy variable transformation to see the difference in predictive model performance. We found it is hard to say which one performs better. We can only say the performance is more stable using N-to-(N-1) categorical-dummy variable transformation.

Table 7: AUC - Metrics Categorical Variable Transformation

R	5	10	15	20	25	30	35	40
base	0.601	0.465	0.395	0.440	0.478	0.451	0.463	0.537
(N-1) dummies	0.569	0.512	0.477	0.480	0.419	0.467	0.464	0.456

Table 8: PRAUC - Metrics Categorical Variable Transformation

R	5	10	15	20	25	30	35	40
base	0.204	0.138	0.102	0.119	0.127	0.117	0.122	0.142
(N-1) dummies	0.569	0.512	0.477	0.480	0.419	0.467	0.464	0.456

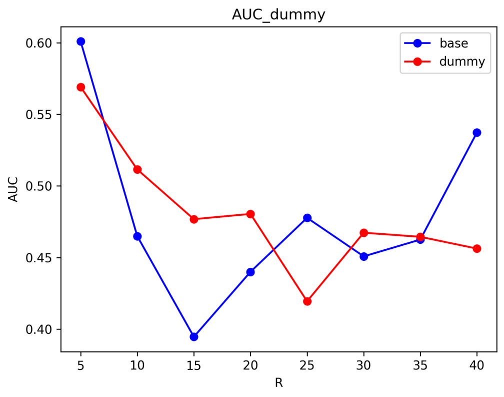

Figure 15: AUC - Dummy Variable Transformation

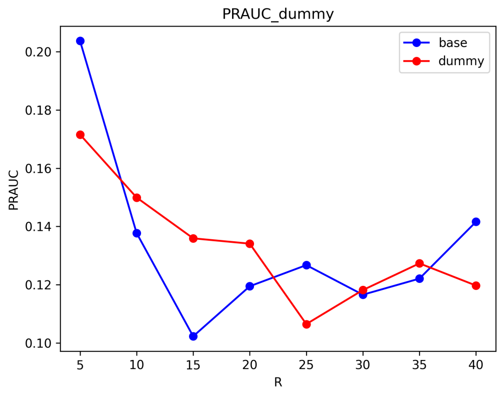

Figure 16: PRAUC - Dummy Variable Transformation

5.2.5 Fitting TASTE without Static Features

In this experiment, we fit the TASTE model without static features. More precisely, we fit the TASTE model with identical static features across all patients. To our surprise, except for the situation where the singular matrix error occurs, fitting the TASTE model without static features performs very well, which is even the best among all the experiments. Similar to the experiment of categorical-dummy variable transformation, one possible explanation is that the manipulation of static features need more attentions to provide meaningful information.

Table 9: AUC - Metrics without Static Features

R	5	10	15	20	25	30	35	40
base	0.601	0.465	0.395	0.440	0.478	0.451	0.463	0.537
without static	0.963	0.927	0.637	0.923	0.000	0.000	0.000	0.000

Table 10: PRAUC - without Static Features

R	5	10	15	20	25	30	35	40
base	0.204	0.138	0.102	0.119	0.127	0.117	0.122	0.142
without static	0.819	0.872	0.293	0.837	0.000	0.000	0.000	0.000

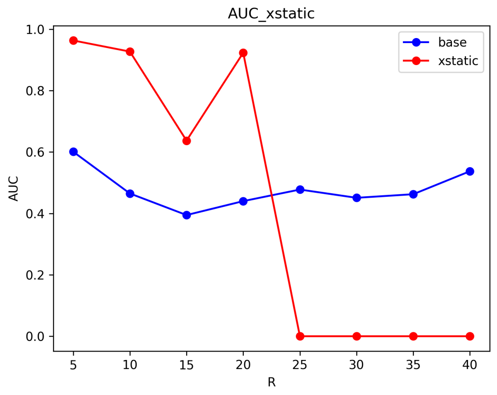

Figure 17: AUC - Without Static Features

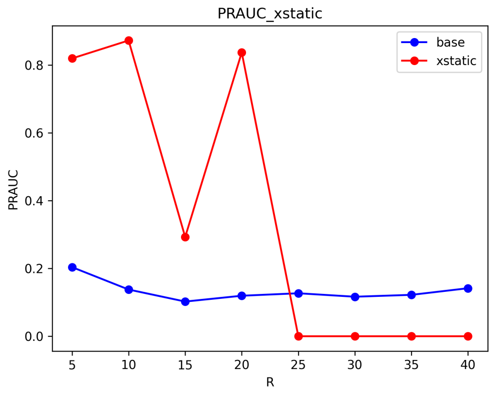

Figure 18: PRAUC - Without Static Features

5.2.6 Varying R

In this experiments, we vary the number of phenotypes $\boldsymbol{R}$ specified in the TASTE model for the above 5 experiments to see whether increasing number of phenotypes improves the model performance. From the figures and tables in previous experiments, we found that increasing R up to more than 10 usually generate worse results. The number we found here can only be applied to our data sets. With other data sets of different patient compositions and features, the optimized number of extracted phenotypes would change.

6 Discussion

From the 6 experiment results, we found that R values less than 10 are more suitable for our data sets. To fit the TASTE model with R values less than 10, we only need 0.2 to 0.4 training sample split to have a good enough performance. Under this condition, an observation window of 720 days is suggested, which meet our hypothesis that longer observation windows increase predictive power. The most interesting discoveries happen in feature manipulation. We suggest that constructing dynamic features by week would be a better choice than by date construction. We also found that the current static feature manipulation could be problematic that discarding static features, which seems losing some useful information, even outperforms all the other experimental settings. Further research needs to pay more attention to static feature construction to provide more meaningful information.

There are some limitations in our experiments. First, we lack k-fold cross-validation processes to yield robust statistics. Also, we are not able to provide a confidence range of metrics because standard deviation is not available without cross-validation. The statistics thus might be subject to data sampling in the train-test split process. Second, we encounter an error of LinAlgError: Singular matrix when fitting the TASTE model in some situations, especially when R is set as some values in some experiments. We handle the error by skipping it and putting 0 as the metrics values but do not properly address the issue itself. Our wild guess to the issue without further proofs is that some combinations of R and static/dynamic feature matrices of particular patient samples are mathematically problematic in matrix operations. Third, the logistic model we use to evaluate the predictive power of different feature extractions is not fine tuned. We only pass in one parameter to reflect the imbalance of cases/controls. Fourth, for the interpretation of experiment results, it is hard for us to provide explanations to our observations because our experiments are not fully controlled to exclude confounding effects. For example, we cannot say TASTE projection takes longer than fitting is due to the computation complexity or due to sample size changes. Another example is that performance changes coming along with training sample size changes could be the result of TASTE phenotyping or the result of logistic regression model training. We could not separate the effects of TASTE modeling and logistic regression modeling to tell their respective attributions. Overall, we are bound by time to satisfy the rigorosity. We believe all the issues worth more attentions and explorations in the future research.

7 Conclusion

Temporal And Static TEnsor Factorization (TASTE) proposed by Afshar et al. (2019)[1] is an effective algorithm to identify relevant features for phenotyping, especially in the condition where the data consists of both time-varying and non-time-varying variables and the duration of time-varying variables varies for subjects. We apply this cutting-edge algorithm to medical records, which exhibit characteristics of high-dimension, messiness, and billing-orientation. TASTE is indeed one of the most suitable techniques in this particular context.

MATLAB has been a language that is widely used in Computational Mathematics. However, in our project, we implemented not only TASTE in Python, but also Nonnegative matrix factorization (NMF) algorithms based on alternating non-negativity constrained least squares, which was originally developed in MATLAB by Dr. Haesun Park. According to our experiences, MATLAB should be faster than Python, since MATLAB was designed to do the matrix manipulation. However, in our experiment, we found that Python, especially NumPy had improved a lot. Although we did not do a thorough comparison between NumPy and MATLAB, as that is not our main purpose of the project, we found that NumPy beats MATLAB in all the occurrences. Thus we believe that our project turns out to be far better that we expected as we not only implemented TASTE and NMF in python, but also found that NumPy is not inferior to MATLAB.

We have brought TASTE and NMF to the world of Python[2]. As currently most popular big data tools, machine learning libraries and state-of-art deep learning algorithms are mostly implemented in Python or can be easily connected to Python, we have confidence to say that we made a great contribution to the field of big data in health. Future researchers can directly apply our code to utilize TASTE or NMF in Python without switching to MATLAB.

References

[1] Ardavan Afshar, Ioakeim Perros, Haesun Park, Christopher deFilippi, Xiaowei Yan, Walter Stewart, Joyce Ho, and Jimeng Sun. Taste: Temporal and static tensor factorization for phenotyping electronic health records. In *Proceedings of the ACM Conference on Health, Inference, and Learning*, CHIL '20, page 193–203, New York, NY, USA, 2020. Association for Computing Machinery.

[2] Joyce C Ho, Joydeep Ghosh, and Jimeng Sun. Marble: high-throughput phenotyping from electronic health records via sparse nonnegative tensor factorization. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 115–124, 2014.

[2] `https://github.com/Yanxin-Ye/TASTE`

[3] Yichen Wang, Robert Chen, Joydeep Ghosh, Joshua C Denny, Abel Kho, You Chen, Bradley A Malin, and Jimeng Sun. Rubik: Knowledge guided tensor factorization and completion for health data analytics. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1265–1274, 2015.

[4] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122, 2011.

[5] Joyce C Ho, Joydeep Ghosh, Steve R Steinhubl, Walter F Stewart, Joshua C Denny, Bradley A Malin, and Jimeng Sun. Limestone: High-throughput candidate phenotype generation via tensor factorization. *Journal of biomedical informatics*, 52:199–211, 2014.

[6] Richard A Harshman. Parafac2: Mathematical and technical notes. *UCLA working papers in phonetics*, 22(3044):122215, 1972.

[7] Ioakeim Perros, Evangelos E Papalexakis, Fei Wang, Richard Vuduc, Elizabeth Searles, Michael Thompson, and Jimeng Sun. Spartan: Scalable parafac2 for large & sparse data. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 375–384, 2017.

[8] Ardavan Afshar, Ioakeim Perros, Evangelos E Papalexakis, Elizabeth Searles, Joyce Ho, and Jimeng Sun. Copa: Constrained parafac2 for sparse & large datasets. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 793–802, 2018.

[9] Jingu Kim, Yunlong He, and Haesun Park. Algorithms for nonnegative matrix and tensor factorizations: A unified view based on block coordinate descent framework. *Journal of Global Optimization*, 58(2):285–319, 2014.

[10] Peter H Schönemann. A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 31(1):1–10, 1966.

[11] Jingu Kim and Haesun Park. Fast nonnegative matrix factorization: An active-set-like method and comparisons. *SIAM Journal on Scientific Computing*, 33(6):3261–3281, 2011.

[12] Edward Choi, Andy Schuetz, Walter F Stewart, and Jimeng Sun. Using recurrent neural network models for early detection of heart failure onset. *Journal of the American Medical Informatics Association*, 24:361–370, 2017.

[13] Ben Miroglio. Matching techniques for observational studies. `https://github.com/benmiroglio/pymatch`, 2017.

[14] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.