# Iterative Constringency Optimization: Preclustering Approach to Agent Interactive Data

Binfeng Xu

*Wake Forest University*

**Abstract**

We present ICO, a preprocessing framework to enhance model performance on agent interactive data. Agent interactive data involves two agent data sets and one interaction data set, where the agent data contains private features only related with the entity itself, and the interaction data contains interactive features of two agents. The main idea behind ICO is to fully exploit respective features of two agents before modeling by clustering two agents into groups, hoping to find potential relationships among groups of agents. To determine a proper clustering solution, we optimize Constringency, a quantity describing the overall strength of correlation between groups of two agents. The result of such pre-clustering approach is very promising. We have achieved increasing performances by training models based on multiple preprocessing structures of ICO.

*Keywords:* Clustering, Preprocessing, Constringency

## 1. Introduction

As the database storage capacity and people's awareness of data importance rise, data science are both benefited by stronger support of data records and bothered by more complicated data structures. We recognize that one of the most pervasive data pattern involves two agents and one interaction. For example, in a investment application settings, an investment bank usually have access to three types of data:

- Agent1: Applicants and their personal information(Age, Credit Record, Real Estates, etc.)
- Agent2: Banks and related information(Location, Total Property, Monthly Investing Amount, etc.)
- Interaction: A sample that represents a single application between a specific applicant and bank.

Similar examples are happening everyday and everywhere: Customers in San Francisco seeking restaurants for lunch; Car Owners in US seeking companies for car insurance; people in Winston-Salem seeking banks for applying credit cards. However, it is not uncommon that specific groups of an agent (e.g. Confident investment seekers) have preference or tendency to interact with a specific group of the other agent (e.g. Big-name investment banks). Such assumption implies potential stratification of feature spaces within the original data. As a result, it is not proper to treat all samples as equal when training models for regression/classification tasks. Instead, we hope to find a clustering solution between two agents that captures the potential relationship between clusters in respective agents.

## 2. Optimizing Clustering Solution

### *2.1. Preclustering of Agent Interactive Data*

An Agent Interactive data set consists of identities of two agents as categorical variables and feature spaces pertaining to respective agents. Therefore, each sample can be viewed as an linkage between agents. The idea of preclustering hinges on partitioning the original data into two groups, each containing a single agent together with all features pertaining to it. The selection of such feature groups is subject to their semantic meaning. For instance, if an agent is customer ID, then features like ages, heights, income levels and marital status should be grouped with the agent; If the other agent is bank ID, then the pertaining features are market capacity, Yelp ratings, locations, etc. After partitioning two agent data sets from the full, it is necessary to shrink them so that each unique ID of the agent corresponds to a single sample. One common way is to take all samples with the same agent ID and calculate feature means as new features.

| Customer Label | Store Label | Amount Spent | Type of Store |
|---|---|---|---|
| $S_I$ | | $S_{A_1}$ | $S_{A_2}$ |

Table 1: Illustration: Decomposing $S$ into $S_I, S_{A_1}, S_{A_2}$.

Given an Agent Interactive data $S$ of size $(m \times n)$, we can decompose it into an Interaction $S_I$ of size $(m \times 2)$ and two Agent data $S_{A_1}$ of size $(m \times a_1)$ and $S_{A_2}$ of size $(m \times a_2)$. We can also view $S_I$ as a set of *linkages* :

$$\mathbf{I} = \{(l_1^{(1)}, l_2^{(1)}), (l_1^{(2)}, l_2^{(2)}), ...., (l_1^{(m)}, l_2^{(m)})\}.$$

Where $(l_1^{(i)}, l_2^{(i)})$ is a linkage corresponding to one sample from $S_I$.

Since usually there are duplicated entries in the agent ID of $S_{A_1}$ and $S_{A_2}$, we need to group them by the unique agent ID and take average on the rest variables to shrink $S_{A_1}$ and $S_{A_2}$ into data sets $A_1$ of size $(k_1 \times a_1)$ and $A_2$ of size $(k_2 \times a_2)$, where $k_1$ and $k_2$ are the counts of unique ID's of two agents. Then, it is possible to implement unsupervised clustering methods on $A_1$ and $A_2$. While no specific clustering algorithm is preferred, we use KMeans as default for demonstration. The clustering outcome is highly subject to the number of clusters proposed in the first place, bounded by $k_1$ and $k_2$. In order to find the best of such solutions, it is necessary to define a metric called Constringency.

### *2.2. Constringency*

So far we have produced clusters from $A_1$ and $A_2$:

$$G^{(1)} = \{G_1^{(1)}, G_2^{(1)}, ..., G_p^{(1)}\}$$

$$G^{(2)} = \{G_1^{(2)}, G_2^{(2)}, ..., G_q^{(2)}\}$$

Where each $G_i^{(1)}$ is a set of agent1 IDs from $i^{th}$ cluster, same as $G_i^{(2)}$. And $p$, $q$ are the total number of clusters from $A_1$ and $A_2$.

For a cluster $v \in G^{(2)}$, define $F(l_2 \in G_v^{(2)} | l_1 \in G_u^{(1)})$ to be the number of linkages from cluster $u \in G^{(1)}$ to cluster $v \in G^{(2)}$. Let $(l_1, l_2)$ be a random sample from $I$, then *forward Constringency* and *backward Constringency* are defined by:

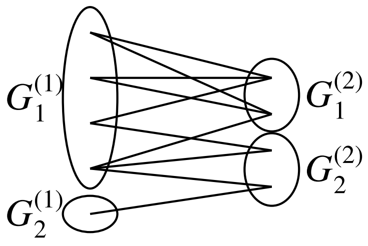


Figure 1: Example of clustering two groups where $p = q = 2$. In this case $C^{p,q}_{forward} = \frac{6}{9} \times \frac{4}{5} + \frac{1}{1} \times \frac{1}{5} = 0.733$, and $C^{p,q}_{backward} = \frac{6}{6} \times \frac{2}{4} + \frac{3}{4} \times \frac{2}{4} = 0.875$. $B^{p,q}_{forward} = \max\{\frac{2}{4}, \frac{2}{4}\} = 0.5$, $B^{p,q}_{backward} = \max\{\frac{1}{5}, \frac{4}{5}\} = 0.8$

$$C^{p,q}_{forward} = \sum_{u=1}^{p} \max_{1\le v\le q}\{F(l_2 \in G_v^{(2)} | l_1 \in G_u^{(1)})\} \frac{|G_u^{(1)}|}{\sum_{i=1}^{p} |G_i^{(1)}|}, \tag{1}$$

$$C^{p,q}_{backward} = \sum_{v=1}^{q} \max_{1\le u\le p}\{F(l_1 \in G_u^{(1)} | l_2 \in G_v^{(2)})\} \frac{|G_v^{(2)}|}{\sum_{i=1}^{q} |G_i^{(2)}|}. \tag{2}$$

Constringency describes how one cluster from Agent 1 maps to a group of clusters from Agent 2. The constringency metric is high when an Agent 1 cluster maps primarily to one cluster in Agent 2. For instance, if all linkages starting from $G_i^{(1)}(1 \le i \le p)$ ends in the same cluster $G_j^{(2)}, 1 \le j \le q$, then the forward constringency reaches its maximum value, $C^{p,q}_{forward} = 1$,vice versa. However, if all such linkages $G_i^{(1)}(1 \le i \le p)$ are randomly distributed in different clusters in $G^{(2)}$, then the constringency has minimum value, defined as *forward and backward baseline*:

$$B^{p,q}_{forward} = \max_{1\le v\le q} \frac{|G_v^{(2)}|}{\sum_{i=1}^{q} |G_i^{(2)}|} \tag{3}$$

$$B^{p,q}_{forward} = \max_{1\le u\le p} \frac{|G_u^{(1)}|}{\sum_{i=1}^{p} |G_i^{(1)}|} \tag{4}$$

Forward and backward baselines are dependent on the relative size of the largest cluster in an agent. Therefore, when comparing different clustering solutions using constringency metrics, one must transform constringencies into the same scale. Similar to idea of relative error, we define *Relative Constringency* as:

$$RC^{p,q}_{forward} = \frac{\left|C^{p,q}_{forward} - B^{p,q}_{forward}\right|}{B^{p-q}_{forward}}, \tag{5}$$

$$RC^{p,q}_{backward} = \frac{\left|C^{p,q}_{backward} - B^{p,q}_{backward}\right|}{B^{p-q}_{backward}}. \tag{6}$$

## *2.3. Maximizing Constringency*

Clustering algorithms like KMeans takes as a parameter the number of presupposed clusters. For constringency, it is necessary to set upper bounds for the number of clusters for each

**Algorithm 1** Computing Constringency

```
Require: $G1$ and $G2$ are dictionaries that map agent IDs to their clustering labels; $I$ is the interaction data that map Agent1 to Agent2. $N1$ and $N2$ are number of clusters in each group.
 1: I = merge(I, G1) on Agent1
 2: I = merge(I, G2) on Agent2
 3: cf = 0                                              ▷ Initialize forward constringency
 4: for i = 1:N1 do
 5:     w = sizeof(G1.value == i) / sizeof(G1)          ▷ Calculating weights
 6:     tmp = I[I.G1.value == i]
 7:     r = max(tmp.G2.UniqueCount) / sizeof(tmp)       ▷ Calculating max ratio
 8:     cf = cf + w * r                                 ▷ Update forward constringency
 9: end for
10: cb = 0                                              ▷ Initialize backward constringency
11: for i = 1:N2 do
12:     w = sizeof(G2.value == i) / sizeof(G2)          ▷ Calculating weights
13:     tmp = I[I.G2.value == i]
14:     r = max(tmp.G1.UniqueCount) / sizeof(tmp)       ▷ Calculating max ratio
15:     cb = cb + w * r                                 ▷ Update backward constringency
16: end for
```

of the two agents, respectively $P$ and $Q$. Conditional on $P$ and $Q$, we define the numbers of clusters $p^*, q^*$ that maximize constringency as:

$$(p^*, q^*) = \underset{(p,q)}{\operatorname{argmax}} \, (RC^{p,q}_{forward} + RC^{p,q}_{backward}), \tag{7}$$

where $0 < p \leq P$ and $0 < q \leq Q$.

Since a linear search for $p^*$ and $q^*$ takes $O(|pq|)$ time, there is no guarantee to find a global minimum unless the upper bound $P = k_1$ and $Q = k_2$. However, such computation is usually expensive given large $k_1$ and $k_2$. A suggested first value for $P$ and $Q$ is $P = \frac{k_1}{10}$ and $Q = \frac{k_2}{10}$. This is chosen to balance computational time and solution space.

## *2.4. Choice of Clustering Algorithm*

In addition to search boundaries $P$ and $Q$, the other major factor that influences results of optimized Constringency (ICO) is the choice of clustering algorithm. Given the same data, values of Constringency can vary significantly with different clustering algorithms and hyperparameters. Theoretically, there is no guarantee on which algorithm is better or worse. However, it is crucial to choose an algorithm capable of clustering new data, meaning data not used to train the clustering solution. In other words, the algorithm should be able to predict clustering labels for data that are not used to fit the model. Such prerequisite ensures that models implementing ICO is capable of predicting unlabeled test data. Examples of such clustering algorithms are KMeans, Affinity Propagtion[1], Mean Shift[2], Birch[3] and Gaussian Mixture Models. Conversely, algorithms like Spectral Clustering, Agglomerative Clustering and DBSCAN[4] are not recommended.

While the only solid way to make the best choice is to try all clustering algorithms, there are some trade-offs among algorithms to consider. KMeans optimizes within-cluster sum-of-squares and can be used for a variety of clustering purposes. The time complexity of Affinity

Propagation and Mean Shift are quadratic with respect to sample size and can be extremely slow for large data. However, both of these algorithms do not take number of clusters as input; Affinity Propagation uses preference for exemplars and damping factors to control messages sent between pairs of samples, while Mean Shift uses bandwidth to restrain regions considered for centroids. Birch reduces the input data to a set of subclusters which are leaves of its Characteristic Feature Tree (CFT), and is fast when sample size is huge. However, Birch does not scale well for high dimensional data. Gaussian Mixture Model is probably less used because it assumes too much normality of data and has a high complexity in time.

In our applications in Section 3, we utilize four different clustering techniques and compare the performance of each with ICO.

### *2.5. Preprocessing Data with ICO*

After optimizing Constringency, we have in hand the optimal set of clusters. There are multiple approaches to exploit these clustering labels. Here we propose three preprocessing methods aiming at improving models for regression and classification tasks.

- **Constringency Encoding**. Map respective angent ID's into the clustering labels for each sample, and One-Hot Encode these labels.
- **Stratified Modeling**. Parse data into smaller data set based on one of the agent clusters. Then fit models and make prediction within each cluster, and finally combine the result.
- **Constringency Feature**. Adding attained clustering labels as new features used in fitting model.

We will show and discuss experimental results of these preprocessing methods on different data set. ICO is not limited to these techniques, and further applications are a subject of future work.

## 3. Experiments

### *3.1. Simulated Data*

We start testing ICO with simulated data of 5000 samples and 23 variables. These variables are consisted of a target variable, 2 agent IDs, and 10 related variables for each agent. Samples from an agent, together with its 10 features are drawn from normal distribution with unique mean and variance. Then, two sets of agents are randomly combined to generate the final data. The target variable is the average of two values drawn from the two normal distributions from respective agents.

Before fitting simulated data to regress on target variable, we preprocess the data separately by the three methods mentioned in section *2.5*. The control group is preprocessed by Label Encoding which transforms identity values of agents into numerical values. Then we compare 5-fold cross validation Mean Squared Error (MSE) from multiple models including Linear Regression, Ridge Regression, K Nearest Neighbors, Random Forest [5] and Gradient Boosting Decision Trees [6]. To ensure ubiquity of experimental results, we set different random seeds to generate 1000 different data set, iteratively recording mean squared error obtained from different models and average these values in the end.

The experiment suggests an overall advantage of Constringency. A majority of models preprocessed by ICO attains lower MSE to some extent (Figure 2). Constringency Encoding results in the biggest reduction in MSE when used in linear models such as Linear Regression and Ridge Regression. However, it is not as effective against tree-based models. Constringency Encoding significantly decreases the cardinality of identity variable( i.e., the Agent IDs), which have ordinal numeric values that the linear models attempt to model, despite the fact the order is in reality meaningless. On the other hand, such transformation is not as effective for tree-based models depending on binning and parsing numerical features. However, we believe that binning on agent identity variables is subject to overfitting because it is semantically meaningless in real life to split trees based on individual agent. Therefore Constringency Encoding more or less prevents potential overfitting of tree models. Constringency Feature works almost oppositely and results in the biggest reduction in MSE when used in Random Forest and Gradient Boosting Decision Tree. Adding cluster labels into tree models gives decision trees a new option to split on. Therefore, the decreased cross validation MSE suggests that the added cluster label contains useful information to make prediction. Finally, Stratified Modeling performs relatively stable reduces MSE evenly in almost all cases. Nonetheless, we do not recommend this method because of its huge complexity in time. Stratified Modeling could be extremely slow when number of clusters, $p^*$ and $q^*$ become large.

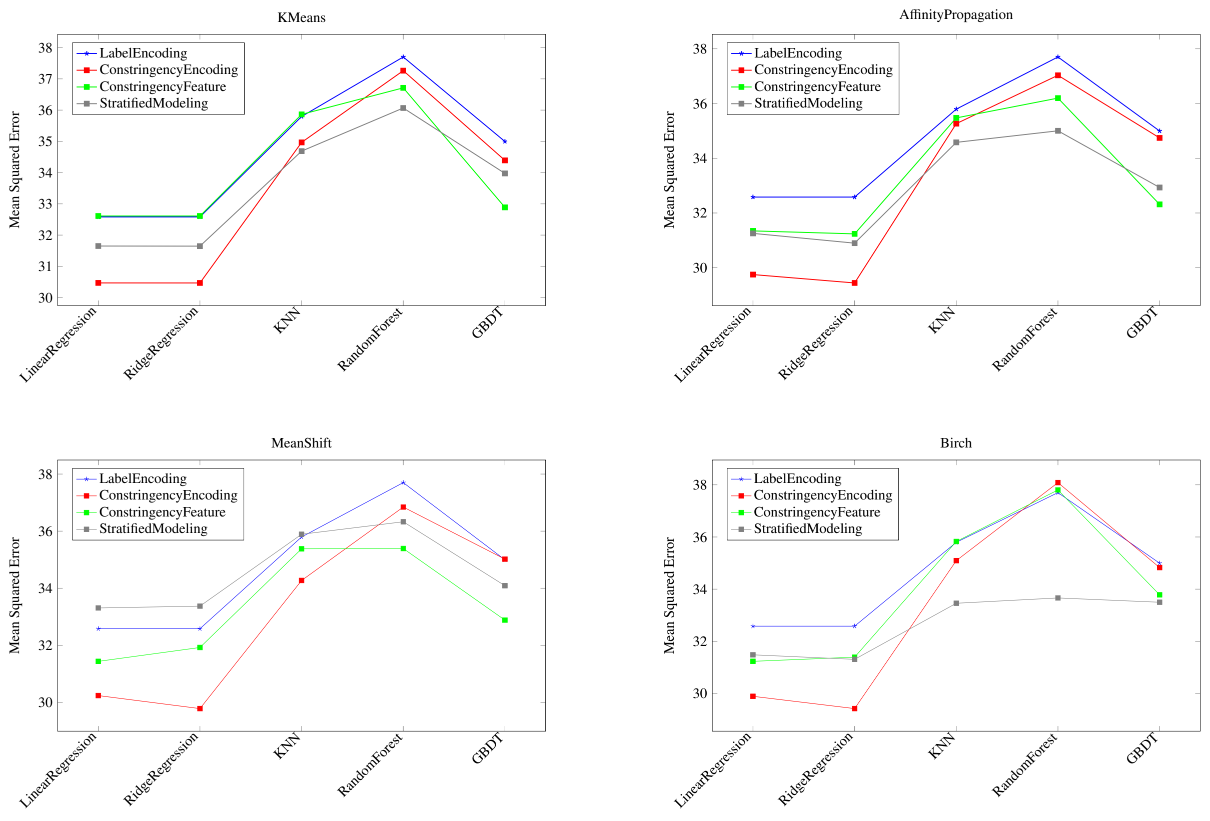
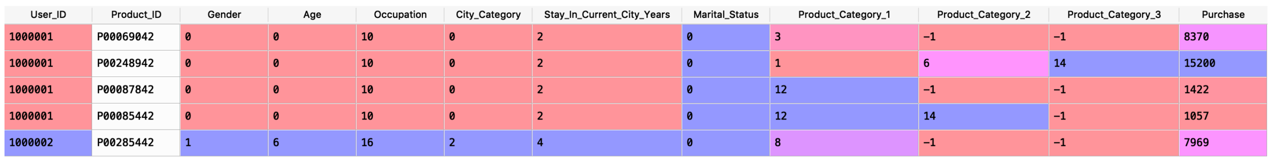


Figure 2: Cross validation Mean Squared Error results from simulated data. Each figure corresponds to the unique clustering algorithm used for optimizing Constringency (KMeans, Affinity Propagation, Mean Shift and Birch), while lines correspond to MSE of different fitted models with specific preprocessing approach. In general, models implementing ICO have smaller MSE than those without using Constringency ( indicated by blue lines).

## 3.2. *Black Friday Data Set*

The next experiment exploits a data set about purchase happened on Black Friday [7]. The data contain Customer ID’s (Agent 1) with personal features such as age and occupation,



| User_ID | Product_ID | Gender | Age | Occupation | City_Category | Stay_In_Current_City_Years | Marital_Status | Product_Category_1 | Product_Category_2 | Product_Category_3 | Purchase |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1000001 | P00069042 | 0 | 0 | 10 | 0 | 2 | 0 | 3 | −1 | −1 | 8370 |
| 1000001 | P00248942 | 0 | 0 | 10 | 0 | 2 | 0 | 1 | 6 | 14 | 15200 |
| 1000001 | P00087842 | 0 | 0 | 10 | 0 | 2 | 0 | 12 | −1 | −1 | 1422 |
| 1000001 | P00085442 | 0 | 0 | 10 | 0 | 2 | 0 | 12 | 14 | −1 | 1057 |
| 1000002 | P00285442 | 1 | 6 | 16 | 2 | 4 | 0 | 8 | −1 | −1 | 7969 |

Figure 3: Head of Black Friday data set.

Product ID's (Agent 2) with related product category features, and finally the target variable, Purchase Amount. The data size, after removing entries with missing values, is 537577 × 12.

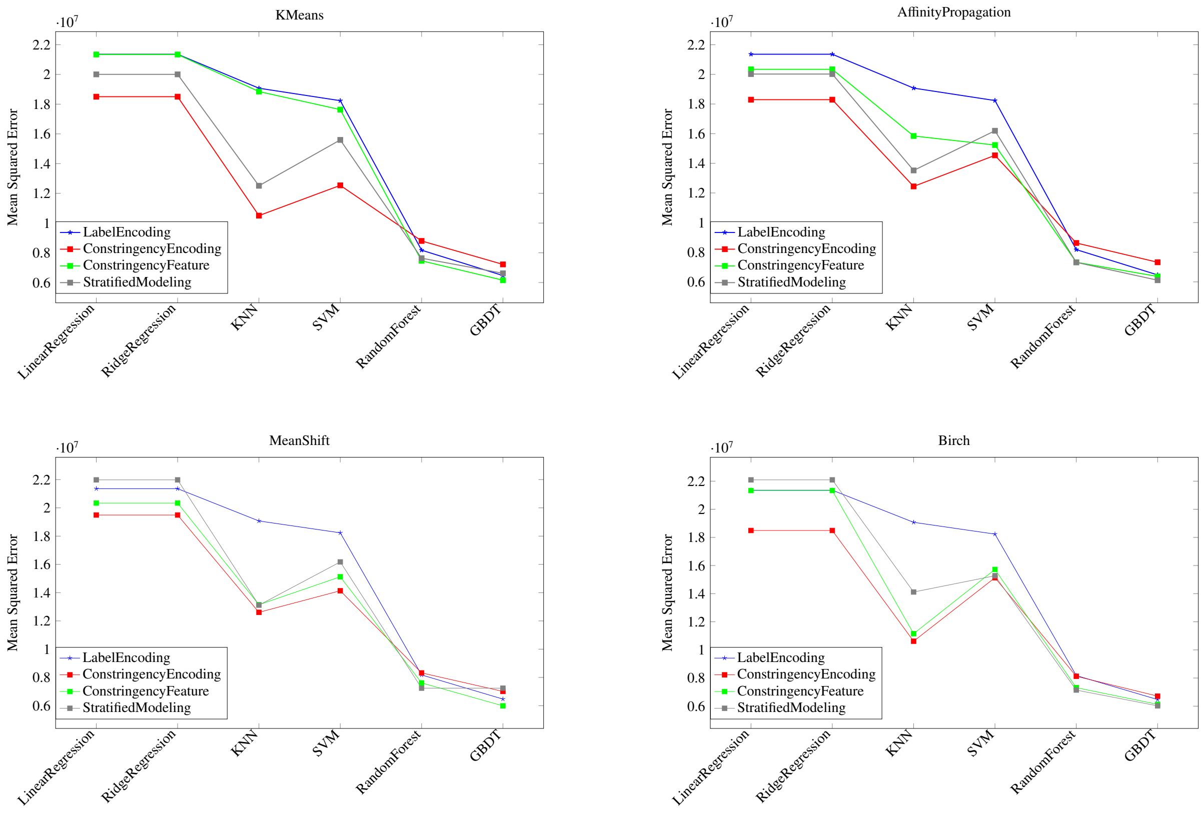


Figure 4: Cross validation Mean Squared Error results from Black Friday data set.

Similar to what we did for simulated data, we preprocess Black Friday data set with three approaches based on ICO, and fit regression models to predict purchase amounts for each customer-product pair. Model performance metric is measured by 5-fold cross validation MSE. As seen in Figure 4, in general, ICO based preprocessing approaches improve models again. Similar to simulated data, Constringency Encoding reduces most of the MSE for Linear Regression and Ridge Regression. K-Nearest Neighbors and Supporting Vector Machines [8] benefit significantly from this method as well. However we do not observe an obvious drop of MSE for Random Forest and Gradient Boosting Decision Trees. When carefully examining nodes in fitted decision trees, we find that ProductID's and CustomerID's stay near the top of the trees and have a huge influence on decision making. Such phenomenon is not preferred because we don't want the model to be overfitting for individual person and product. Therefore, using Constringency Encoding to replace agent ID's while keeping MSE in the same level is already an improvement. Constringency Feature and Stratified Modeling, on the other hand, steadily reduce MSE for KNN, SVM and even overfitted tree models, but such effect is less obvious for Linear/Ridge Regression. Generally speaking, the results from Black Friday data set match well with our simulated data.

## *3.3. Elo Merchant Category Recommendation*

We use part of the data from Elo Merchant Category Recommendation [9] as our last experiment. The data contains cardID and merchantID and a target variable, purchaseAmount. The goal is the make commercial recommendation based on the purchase quantity of credit card holders when provided with service from different merchants. As shown in Figure 5, we observe an significant decrease in MSE using Constringency Encoding. Even though not obvious on Birch, the improvement of linear models by Constringency Ecoding is generally consistent as before. Besides, the MSE of Supporting Vector Machines is lowered for whatever preprocessing structures and clustering algorithms used.

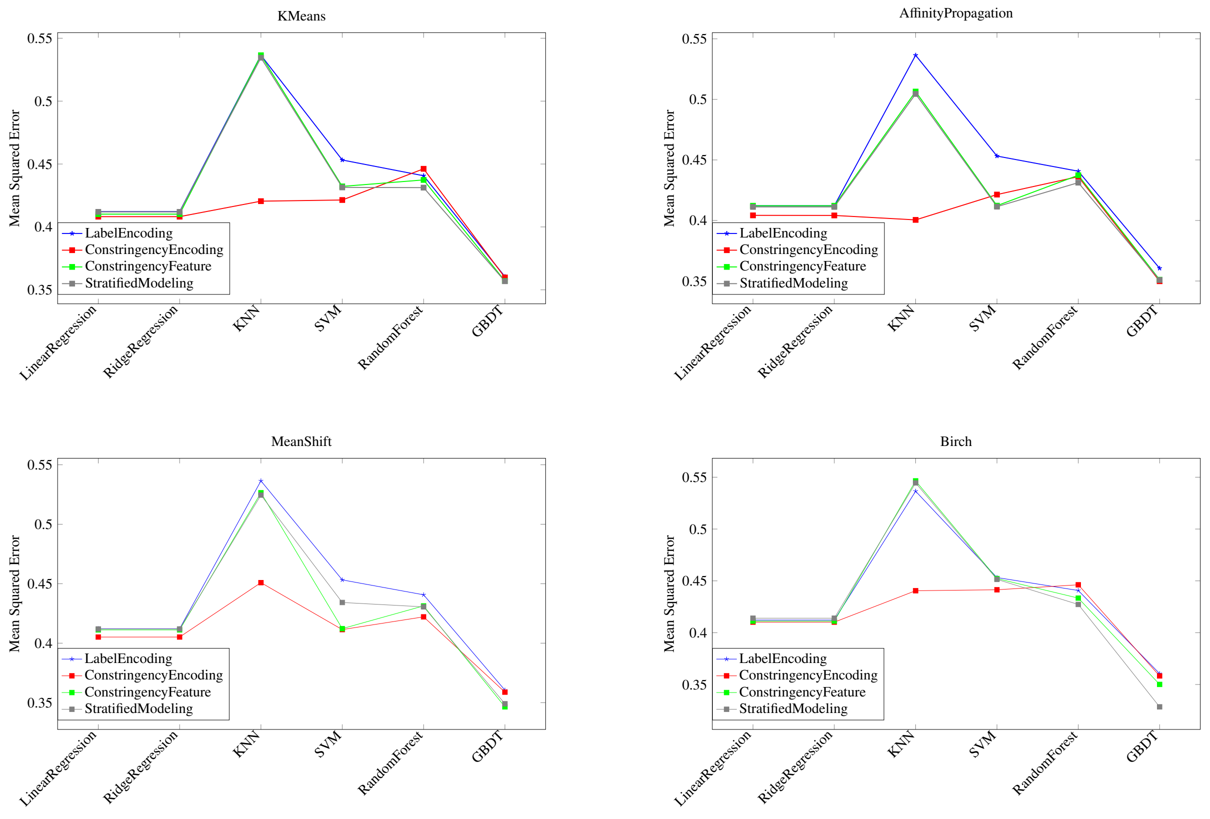


Figure 5: Cross validation Mean Squared Error results from Elo Merchant data set.

MSE's for Gradient Boosting Decision Trees in the last two experiments suggest a sheer advantage over other models. While proposed preprocessing structures are not as effective in terms of reducing MSE to Gradient Boosting Decision Trees as for linear models, one must notice that improving GBDT is relatively difficult. Noticing that preprocessing structures of ICO have minimal improvement of results in almost all cases, it is definitely worthy to try these techniques when optimizing GBDT for advanced accuracy.

## 4. Conclusion and Future Work

So far we have discussed an innovative metric, Constringency, to compare between possible clustering solutions. We've also proposed three preprocessing structures based on optimizing Constringencies. Then we implemented our algorithms on simulated data, Black Friday data and part of data from Elo Merchant Category Recommendation. The experimental results indicate a general improvement of different types of model for whatever clustering algorithms we choose. The overall decrease in cross validation MSE's suggest that our preprocessing techniques effectively enhance predictive power of models on regression tasks. We will continue exploring new approaches to process data based on Constringency optimization and keep experimenting on more real data that involves interactive agents. Since we have only used our method for regression tasks, our next step is to find a suitable data aiming at classification. We are anticipating similar success on such tasks as well.

## References

[1] B. J. Frey, D. Dueck, ”clustering by passing messages between data points”, Science Feb (2007).

[2] D. Comaniciu, P. Meer, mean shift: A robust approach toward feature space analysis, IEEE Transactions on Pattern Analysis and Machine Intelligence (2002) 603–619.

[3] M. L. B. Tian Zhang, Raghu Ramakrishnan, an efficient data clustering method for large databases. (????).

[4] H. P. K. J. S. Ester, M., X. Xu, a density-based algorithm for discovering clusters in large spatial databases with noise, In: Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining, Portland, OR, AAAI Press (1996) 226–231.

[5] Breiman, random forest, Machine Learning (2001).

[6] J. H. Friedman, greedy function approximation: a gradient boosting machine, Annals of statistics (2001) 11891232.

[7] https://www.kaggle.com/sdolezel/black-fridaytrain.csv (????) 2017.

[8] J. C. Platt, probabilistic outputs for support vector machines and comparisons to regularized likelihood methods (1999).

[9] https://www.kaggle.com/c/elo-merchant-category-recommendation/data (2019).