Wells Fargo Campus Analytics Challenge Report

Xuesong Hou* Chunlin Li* Yu Yang*

August 12, 2020

1 Introduction

In this report, we analyze the data set of the Wells Fargo Campus Analytics Challenge. Based on the characteristics of the data, we develop a data analysis procedure for this challenge, as described in the following workflow (Figure 1).

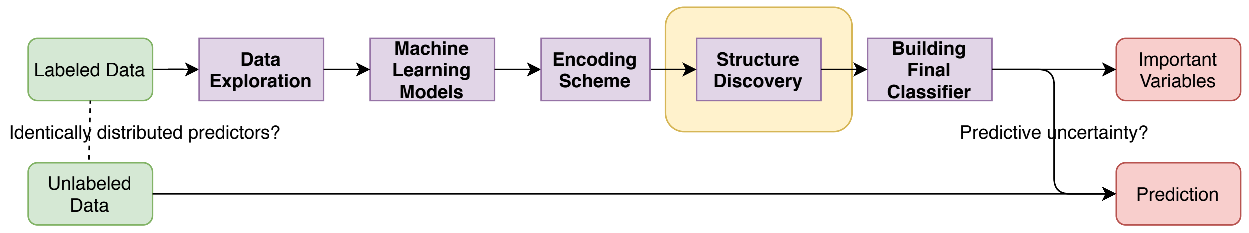

Figure 1: Modeling pipeline.

First, we conduct some routine exploratory data analysis and summarizes the descriptive statistics. Next, we fit popular machine learning models and find linear classifiers outperform the nonparametric models, suggesting a parametric nature of the data. By investigating the coefficients in linear classifiers (linear SVM/logistic regression with L_1 penalty), we figure out an ordinal encoding scheme for the categorical variable.

The foregoing analysis alludes some inherent structures of the data. On this ground, we develop a novel method called Sparse Grouping Pursuit to automatically discover the sparseness and grouping structure among features. This method successfully identifies a single group of important features, leading to a tremendous dimension reduction. Based on this discovery, we fit a linear SVM which turns out to separate the training data perfectly. Finally, we give the prediction and conduct predictive uncertainty analysis.

2 Data Analysis

2.1 Data Exploration

1. Imbalanced label

   The response variable takes a value in classes 0 and 1. Two classes are slightly imbalanced, and about 31% of training data are in class 1.

*School of Statistics, University of Minnesota, Minneapolis, MN, 55455.

2. Distribution of numeric features

   There are 30 numeric features in the data set. All of them have mean zero and standard deviation one, approximately. The Q-Q plots and kernel density estimation plots show that all the numeric features exhibits standard normality, which is further confirmed by the Shapiro-Wilk test. Even for X_{19}, the one with the smallest p-value among 30 tests, the empirical density curve is very close to the standard normal distribution, as shown on the left panel of Figure 2.

3. Correlation of numeric features

   We calculate the sample Pearson correlation, and find that all features are not correlated. A heatmap of correlation is displayed on the right panel of Figure 2.

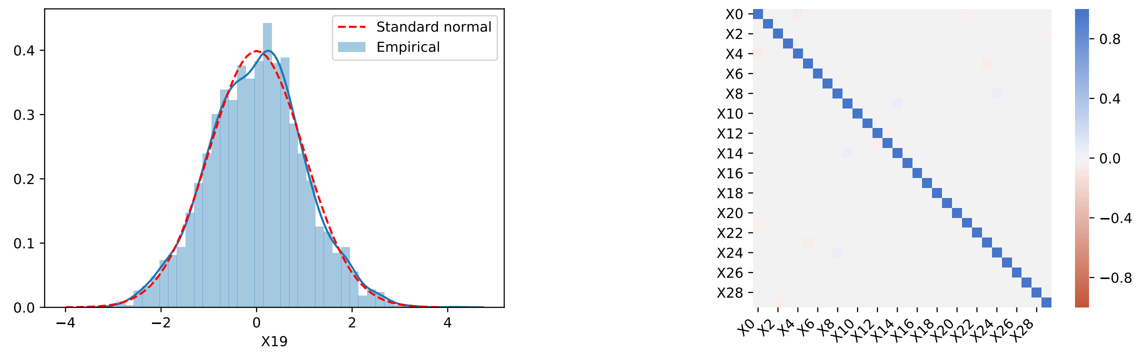

Figure 2: Left: Comparison of empirical density of X_{19} and standard normal. Right: Correlation heatmap.

4. Preliminary variable selection

   We compare the boxplots of each numeric feature grouped by class labels. From the visualization, some of features present different quartiles in different classes, indicating a potential influence on the response [5]. For example, the plots of X_4 and X_5 are shown in Figure 3, where X_4 is likely to be more influential on the response variable than X_5. In this fashion, we identify the following variables that might be of importance:

$$X_2, X_3, X_4, X_6, X_7, X_{11}, X_{15}, X_{17}, X_{19}, X_{20}, X_{21}, X_{22}, X_{25}, X_{26}, X_{27}.$$

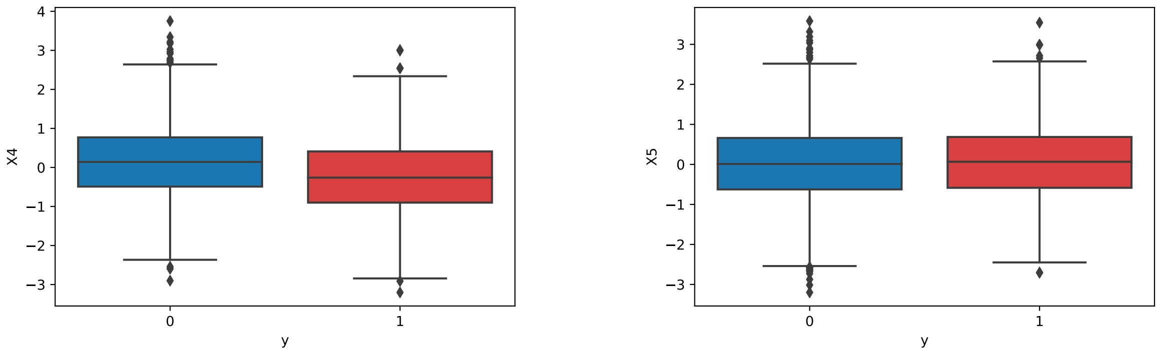

Figure 3: Left: Boxplot of X_4. Right: Boxplot of X_5.

5. Categorical feature

   The data set contains one categorical variable. The counts of each factor level are roughly the same, suggesting that it may be uniformly sampled from $\{A, B, C, D, E\}$. Further, we plot the distribution of the response given each factor level, which shows an ordinal pattern of $\{A, B, C, D, E\}$. More concretely, $X_C = B$ indicates Y is more likely to be 0 comparing with $X_C = A$. This observation suggests an ordinal numerical encoding scheme may be useful.

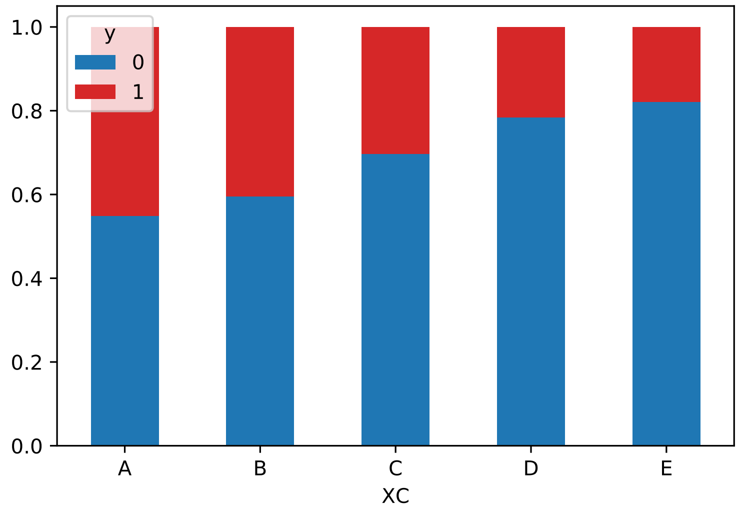

Figure 4: Distribution of Y given X_C.

6. Identical distribution of the training and testing data

   It is important that the training and the testing data are identically distributed in order to make accurate predictions. Therefore, we use the Kolmogorov-Smirnov test to compare the distributions of the training and testing data. It turns out that they are identically distributed.

The code for exploratory data analysis is in `eda.ipynb`.

2.2 Model Fitting and Encoding

2.2.1 Coarse Model Comparison

We commence with multiple candidate models, including Logistic Regression (LR), Linear Discriminant Analysis (LDA), K-Nearest Neighbors (KNN), Classification And Regression Tree (CART), Naive Bayes classifier (NB), Linear Support Vector Machine, LightGBM [4], XGBoost [2], and CatBoost [7]. Logistic Regression and Linear Support Vector Machine use both L_1 and L_2 penalty, and the other models use default parameters. To have a comprehensive evaluation of the models, we consider four performance metrics: AUC, Accuracy, F1, and weighted F1.

As for the encoding of the categorical variable, we start with the most general setting, one-hot encoding, where each factor level is represented by one coefficient. Besides, since the data is imbalanced, although not severe, 10-fold repeated stratified cross validation [6] is used. For a fair comparison, the splitting of the folds in cross validation are the same for all models.

The results are shown in Table 1. The models are sorted in a descending order of performances. The linear classifiers, LR, LDA, and SVM, are the best performing models with respect to all four metrics. In contrast, the gradient boosting methods, which are better at modeling nonlinear relationship, perform worse. It suggests that the data present a parametric characteristic and a linear classifier might be more suitable in this situation.

Figure 5 shows the top 5 models and we can see that L_1-regularized SVM is the best in terms of every metric, and that L_1-regularized Logistic Regression also performs pretty well. Therefore, for the subsequent analysis, we would dig deeper into these two models, in the hope of finding more insights about the data structure. Also note that for both SVM and LR, the L_1-regularized versions outperform their L_2-regularized counterparts, which reveals the sparsity of features and lays foundation for the further analysis.

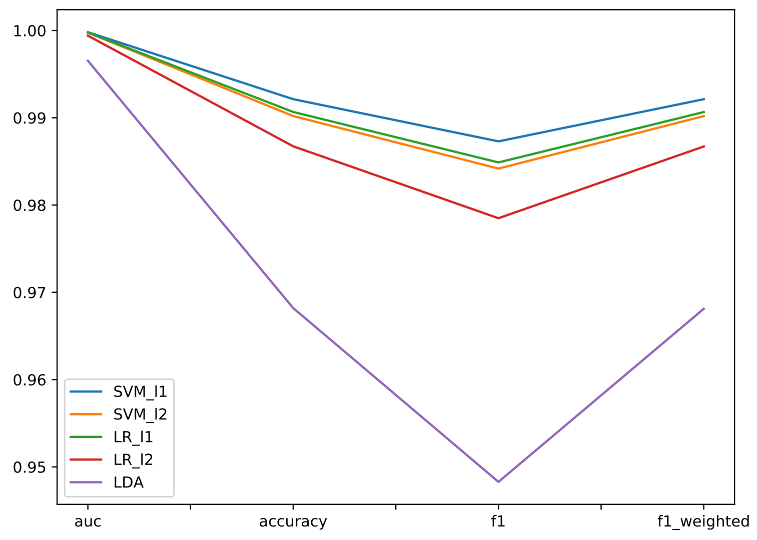

Figure 5: Top 5 models using one-hot encoding.

	AUC	*Accuracy*	*F1*	*Weighted F1*
SVM L_1	0.99981 (0.00020)	0.99213 (0.00515)	0.98730 (0.00832)	0.99213 (0.00516)
SVM L_2	0.99976 (0.00024)	0.99020 (0.00527)	0.98418 (0.00852)	0.99020 (0.00527)
LR L_1	0.99976 (0.00022)	0.99067 (0.00503)	0.98488 (0.00818)	0.99065 (0.00504)
LR L_2	0.99941 (0.00045)	0.98673 (0.00645)	0.97849 (0.01048)	0.98671 (0.00646)
LDA	0.99653 (0.00178)	0.96820 (0.01130)	0.94828 (0.01839)	0.96811 (0.01133)
CatBoost	0.98077 (0.00529)	0.91753 (0.01526)	0.85211 (0.03036)	0.91466 (0.01638)
XGBoost	0.96135 (0.00977)	0.89320 (0.01659)	0.80863 (0.03263)	0.88951 (0.01773)
LGBM	0.95764 (0.00979)	0.88533 (0.01489)	0.79071 (0.02969)	0.88057 (0.01596)
NB	0.93900 (0.01644)	0.88120 (0.01854)	0.79454 (0.03449)	0.87858 (0.01935)
KNN	0.80992 (0.02452)	0.77253 (0.02054)	0.54162 (0.05056)	0.75335 (0.02418)
CART	0.63710 (0.02828)	0.68807 (0.02409)	0.49925 (0.04099)	0.68812 (0.02351)

Table 1: Model comparison with one-hot encoding (in a descending order).

	AUC	*Accuracy*	*F1*	*Weighted F1*
SVM L_1	0.99986 (0.00015)	0.99327 (0.00469)	0.98909 (0.00763)	0.99326 (0.00470)
LR L_1	0.99985 (0.00016)	0.99233 (0.00412)	0.98758 (0.00669)	0.99232 (0.00413)
SVM L_2	0.99980 (0.00019)	0.99213 (0.00484)	0.98726 (0.00786)	0.99212 (0.00485)
LR L_2	0.99963 (0.00030)	0.98967 (0.00496)	0.98320 (0.00812)	0.98964 (0.00498)
LDA	0.99692 (0.00169)	0.97153 (0.01029)	0.95369 (0.01677)	0.97145 (0.01032)
NB	0.98924 (0.00436)	0.90333 (0.01320)	0.81624 (0.02964)	0.89773 (0.01497)
CatBoost	0.98033 (0.00499)	0.91860 (0.01452)	0.85453 (0.02909)	0.91587 (0.01562)
XGBoost	0.96171 (0.00813)	0.89047 (0.01543)	0.80516 (0.02990)	0.88697 (0.01641)
LGBM	0.95763 (0.00859)	0.88560 (0.01454)	0.79320 (0.03079)	0.88125 (0.01591)
KNN	0.83402 (0.02241)	0.79420 (0.01920)	0.59214 (0.04461)	0.77845 (0.02195)
CART	0.65175 (0.02424)	0.70113 (0.02000)	0.51948 (0.03410)	0.70117 (0.01972)

Table 2: Model comparison with ordinal encoding (in a descending order).

2.2.2 Encoding Scheme

Since L_1-regularized SVM model performs the best, and that the estimated coefficients might provide extra information, we decide to dive deeper and check its estimated coefficients. As shown in Table 3 in Appendix, the coefficients for the one-hot encoded categorical variable are approximately $2.5, 0, -2.5, -5, -7.5$, which suggests that the difference between two adjacent levels are almost the same and hence ordinal encoding might be appropriate. The coefficients plot, as shown in Figure 6, further validates our findings.

Therefore, we redo model comparison with the categorical variable encoded by ordinal levels, i.e., $[A, B, C, D, E] \Rightarrow [0, 1, 2, 3, 4]$. Comparing the results in Table 2, with those using the one-hot encoding scenario (Table 1), we find that almost every model achieves a higher score in terms of every metric, especially for Naive Bayes, which justifies the ordinal encoding. The detailed comparisons among the models using the ordinal encoding scheme are shown in Figures 9-14 in Appendix. In the subsequent analysis, we use the ordinal encoding for X_C.

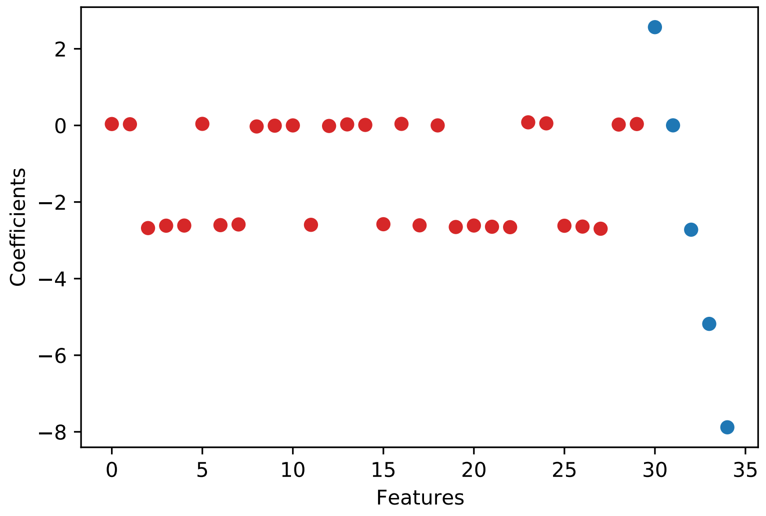

Figure 6: Coefficient plot for SVM using one-hot encoding. The coefficients for the categorical feature is in blue.

2.3 Structure Discovery: Sparse Grouping Pursuit

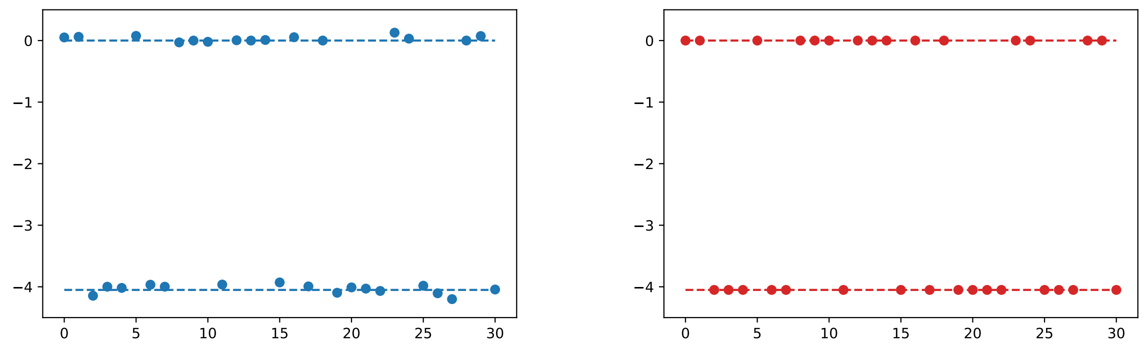

Figure 7: The feature coefficients β estimated by the L_1 logistic regression (Left) and sparse grouping pursuit logistic regression (Right).

To explore the structures of the data, we plot the coefficients estimated by L_1 regularized logistic regression. As shown in Figure 7 (Left), the estimated coefficients are divided into two clusters: one cluster is centered around 0, while the other is centered around -4. It suggests that some features might share the same coefficients.

On this ground, we generalize the Grouping Pursuit [8], referred to as Sparse Grouping Pursuit. Specifically, the estimator is given by

$$(\widehat{\beta}_0, \widehat{\beta}) = \arg\min_{\beta_0,\beta} \; \text{Loss}(\beta_0, \beta) + \lambda(\text{Pen}_0(\beta) + \text{Pen}_G(\beta)), \tag{1}$$

where $\beta_0 \in \mathbf{R}$ is an intercept, $\beta = (\beta_1, \ldots, \beta_d) \in \mathbf{R}^d$ are coefficients of features, and

$$\text{Pen}_0(\beta) = \sum_{j=1}^{d} \min(|\beta_j|/\tau, 1), \quad \text{Pen}_G(\beta) = \sum_{j<j'} \min(|\beta_j - \beta_{j'}|/\tau, 1). \tag{2}$$

In (2), the loss function can be any large margin loss (e.g. logistic loss), and $\lambda > 0$ and $\tau > 0$ are tuning parameters, which can be selected by cross-validation.

Importantly, Pen_0 penalizes on the magnitude of β_j when $|\beta_j|$ is smaller than a threshold τ, but leaves the penalty of β_j constant when $|\beta_j|$ is larger than τ. This will induce sparsity but avoid over-penalization on the estimate. Similarly, Pen_G penalizes on the differences $|\beta_j - \beta_{j'}|$ when it is smaller than τ, but does not impose any more penalty when it is larger than τ. This helps in detecting the potential groups, but avoid false grouping caused by over-penalization where all coefficients are pushed together.

The Sparse Grouping Pursuit is a general method for sparse and group structured data. Figure 7 (Right) shows that the features are divided into two index groups by Sparse Grouping Pursuit,

$$G_0 = \{0, 1, 5, 8, 9, 10, 12, 13, 14, 16, 18, 23, 24, 28, 29\},$$
$$G_1 = \{2, 3, 4, 6, 7, 11, 15, 17, 19, 20, 21, 22, 25, 26, 27, 30\}.$$

where the features in a group have the same coefficients. In particular, G_0 includes all features with zero coefficient. Therefore, Sparse Grouping Pursuit automatically detects the sparsity and the group structure in the data.

Remark. The statistical properties of Sparse Grouping Pursuit are inherited from the original Grouping Pursuit [8]. For computation, (1) can be effectively solved by combining the Majorization Minimization algorithm [3] using the Local Linear Approximation [9] (MM-LLA) and the Alternating Direction Method of Multipliers (ADMM) [1]. The program code for solving (1) is included in `group.py`.

2.4 Final Classifier: Separating Hyperplane

Suppose $G_0, \dots, G_L$ are groups discovered by the Sparse Grouping Pursuit, where $G_k \cap G_{k'} = \emptyset$ and $\bigcup_{k=0}^{L} G_k = \{1, \dots, d\}$. Let G_0 be the group of features with zero coefficients. Then the discovered group structure helps to reduce the dimension of the model further. In particular, we define the grouped features $Z_k = \sum_{j \in G_k} X_j$; $k = 1, \dots, L$. Then we fit a prediction model (e.g. SVM) based on $Z = (Z_1, \dots, Z_L)$.

Since the Sparse Grouping Pursuit discovered groups G_0 and G_1, we define a new feature Z_1 and plot the response Y against Z_1 in Figure 8. Remarkably, the data are linearly separable. Therefore, we consider constructing a separating hyperplane that maximizes the margin of two classes (i.e., linearly separable SVM):

$$\widehat{C}(Z_1) = 0.5 \times (1 - \mathrm{Sign}(Z_1)). \tag{3}$$

Then model (3) is used for final prediction and the prediction result is in `pred.csv`.

2.5 Predictive Uncertainty

It is exciting that our model fits the data perfectly on the training set. Meanwhile, we would like to quantify the confidence of our model on the test set. Since the final classifier (3) is a one-dimensional SVM, we inspect its margin to assess the prediction uncertainty. Generally, a larger margin implies a more robust classifier. For points outside the margin, we would be more confident in the prediction, while for points inside the margin, they are more likely to be misclassified.

For the training data, we obtain a margin $[-2.75 \times 10^{-3}, 1.33 \times 10^{-3}]$. Applying model (3) to the test set, we find that only one point (out of 7000) falls inside this margin, with $Z_1 = -3.58 \times 10^{-4}$. Based on this, we estimate the predicted F1 score would be in $[0.9997, 1.0]$.

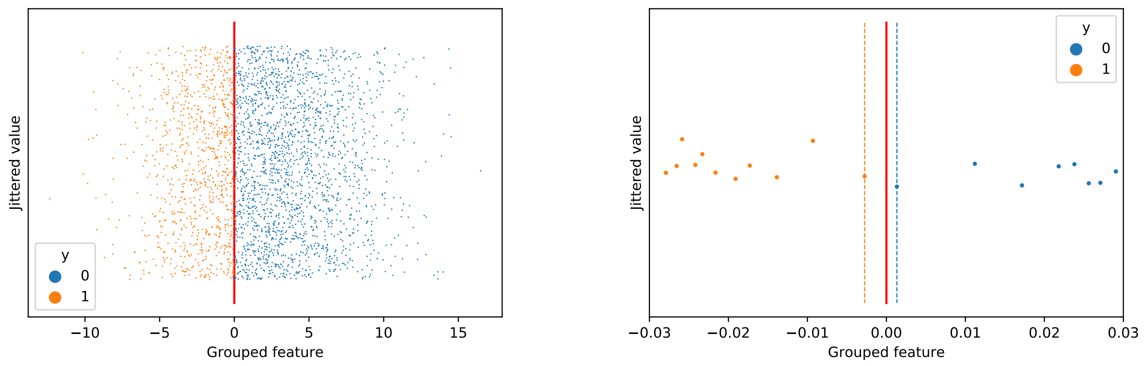

Figure 8: Left: Training set is linearly separable (The vertical axis has no actual meaning.) Right: Zoomed in of the left panel

3 Conclusion and Discussion

3.1 Conclusion

In this challenge, we start with data exploration to examine data distribution and identify the data patterns. Then by model comparison, we find that L_1-regularized SVM and Logistic Regression outperform the other models, and that ordinal encoding scheme yields better performance than one-hot encoding. With the insights obtained in the preceding analysis, we develop a novel method, Sparse Grouping Pursuit, to investigate the inherent parametric structures. The features

$$X_2, X_3, X_4, X_6, X_7, X_{11}, X_{15}, X_{17}, X_{19}, X_{20}, X_{21}, X_{22}, X_{25}, X_{26}, X_{27}, X_C$$

form a least set of informative feature variables as well as a single group, reducing the dimension of the feature space to one and leading to a linearly separable case. Applying the final classifier to the training set, we get a perfect F1 score being 1. The predictive uncertainty analysis shows a high confidence of the prediction on the test set.

3.2 Strengths and Limitations

With a perfect F1 score on the training set, one may raise concerns of overfitting, but this will not be the case. Our method focuses on digging out the inherent structure of the data and the resulting final model is actually a linear classifier with dimension one, having low variance statistically (in terms of bias-variance tradeoff). We would credit our success to the Sparse Grouping Pursuit.

In this challenge, the Sparse Grouping Pursuit is developed for a specific classification task, but it is versatile. It can be easily generalized to other cases where the response is continuous, binomial, multinomial, poisson and so on. As long as a parametric base model is valid, it can be used to find out the inner structure of the data. Meanwhile, the limitation is apparent in that it is not applicable for non-parametric models. We would suggest consider model averaging to combine the insights from both parametric and non-parametric models if non-parametric models turn out to have dominant performances.

3.3 Potential Applications

There are two main usages of the Sparse Grouping Pursuit, reducing dimension and handling multicollinearity. In terms of the former, with the help of Sparse Grouping Pursuit, one can effectively reduce the dimension to the number of groups, and detect low dimensional sparse signals. For example, in gene network analysis where the data is usually high-dimensional, our method can identify homogeneous subnetworks, which would substantially deduct dimension of features and hence help with understanding a disease.

For multicollinearity, if not handled appropriately, it will increase the uncertainty in both estimation and prediction, rendering unreliable and nonrobust results. Specifically, many models in finance, e.g. credit card fraud detection models, are built on top of a wide source of information and it is very likely that these features are highly correlated. With fine-tuned hyper-parameters, Sparse Grouping Pursuit is able to group and aggregate the highly linearly correlated variables. In this way, the multicollinearity issue will be greatly alleviated. Thus, our method can help to stabilize the estimation and prediction of the models and yield more robust results.

3.4 Miscellaneous

Gradient Boosting methods are not necessarily the top one choice when building a model. Sometimes, simple linear models may have better performance if the data nature is linear, such as this project. The key to build good models should always lie in discovering and exploiting the inherent characteristics of the data.

References

[1] Stephen Boyd, Neal Parikh, and Eric Chu, *Distributed optimization and statistical learning via the alternating direction method of multipliers*, Now Publishers Inc, 2011.

[2] Tianqi Chen and Carlos Guestrin, *Xgboost: A scalable tree boosting system*, Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining, 2016, pp. 785–794.

[3] David R Hunter and Kenneth Lange, *Quantile regression via an mm algorithm*, Journal of Computational and Graphical Statistics **9** (2000), no. 1, 60–77.

[4] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu, *Lightgbm: A highly efficient gradient boosting decision tree*, Advances in neural information processing systems, 2017, pp. 3146–3154.

[5] Ker-Chau Li, *Sliced inverse regression for dimension reduction*, Journal of the American Statistical Association **86** (1991), no. 414, 316–327.

[6] 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** (2011), 2825–2830.

[7] Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin, *Catboost: unbiased boosting with categorical features*, Advances in neural information processing systems, 2018, pp. 6638–6648.

[8] Xiaotong Shen and Hsin-Cheng Huang, *Grouping pursuit through a regularization solution surface*, Journal of the American Statistical Association **105** (2010), no. 490, 727–739.

[9] Hui Zou and Runze Li, *One-step sparse estimates in nonconcave penalized likelihood models*, The Annals of Statistics **36** (2008), no. 4, 1509–1533.

A Tables

X_0	X_1	X_2	X_3	X_4	X_5	X_6
0.036534	0.030191	-2.680076	-2.617866	-2.614526	0.040446	-2.603064
X_7	X_8	X_9	X_{10}	X_{11}	X_{12}	X_{13}
-2.585802	-0.027194	-0.00686	0.0	-2.595467	-0.014203	0.026846
X_{14}	X_{15}	X_{16}	X_{17}	X_{18}	X_{19}	X_{20}
0.012906	-2.579457	0.04015	-2.608736	0.000415	-2.652319	-2.613898
X_{21}	X_{22}	X_{23}	X_{24}	X_{25}	X_{26}	X_{27}
-2.645435	-2.65635	0.080488	0.053938	-2.619766	-2.641279	-2.697076
X_{28}	X_{29}	A	B	C	D	E
0.022468	0.036541	2.565274	0.001774	-2.722318	-5.181473	-7.881363

Table 3: Estimated coefficients by L_1-regularized linear SVM.

B Figures

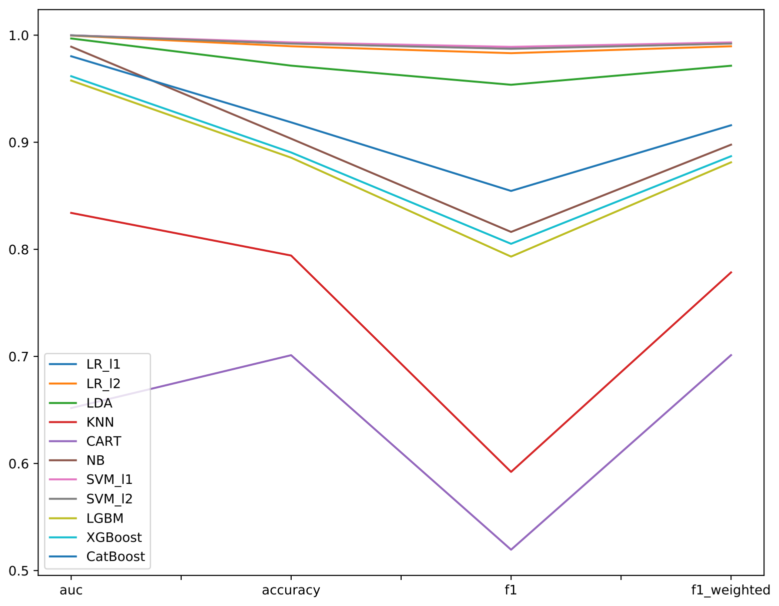

Figure 9: Model comparison using ordinal encoding.

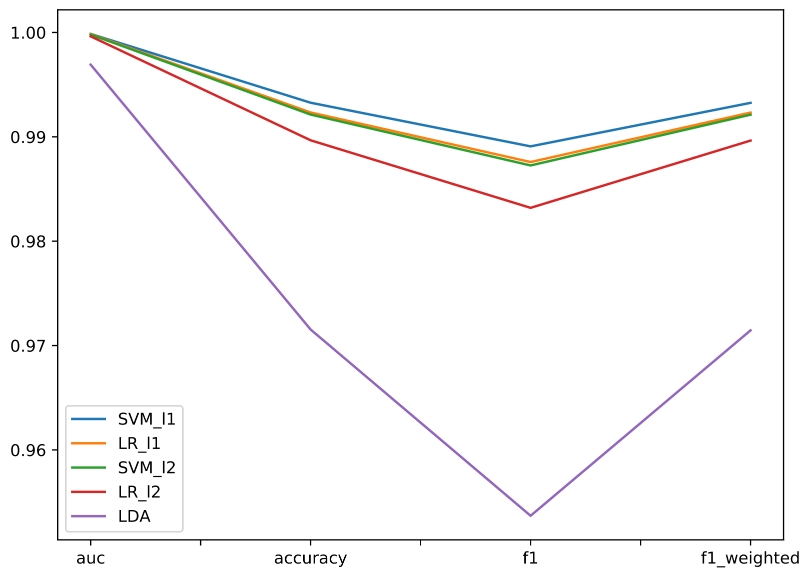

Figure 10: Top 5 models using ordinal encoding.

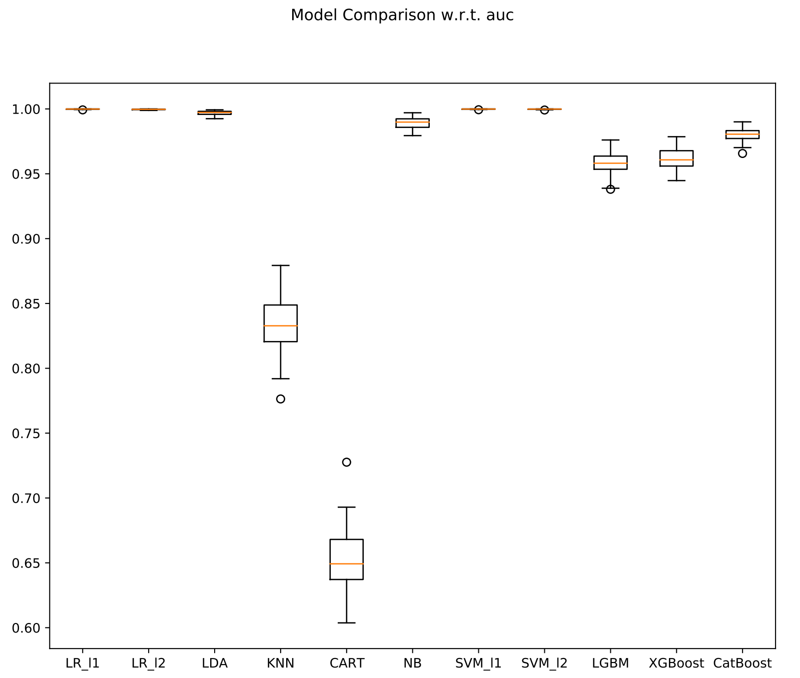

Figure 11: Model comparison in AUC using ordinal encoding.

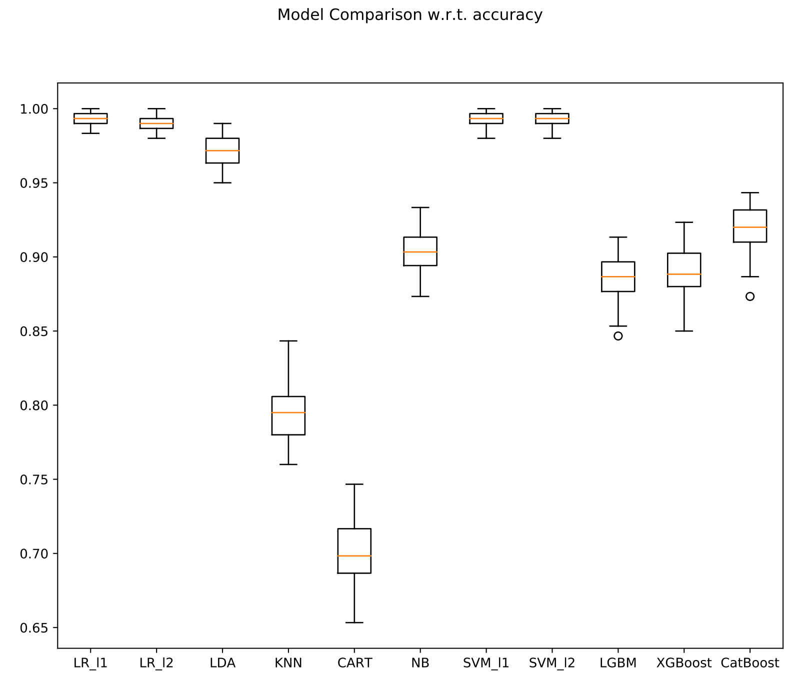

Figure 12: Model comparison in Accuracy using ordinal encoding.

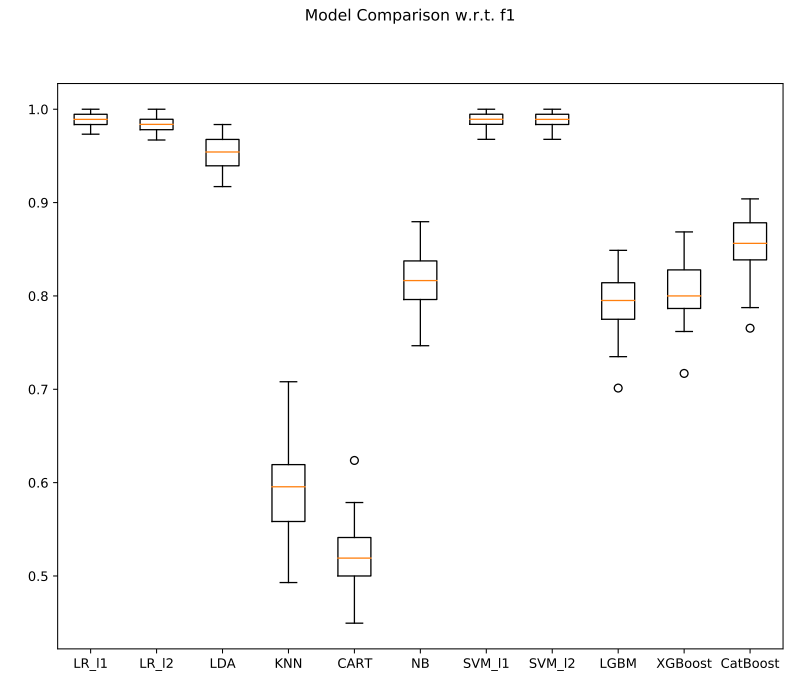

Figure 13: Model comparison in F1 using ordinal encoding.

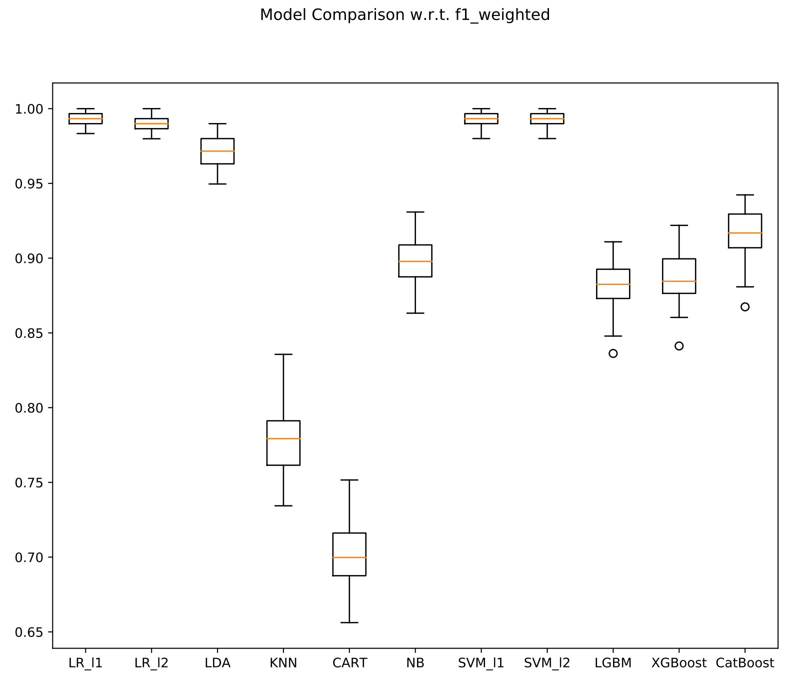

Figure 14: Model comparison in F1-weighted using ordinal encoding.

C Code

1. The final prediction file is `pred.csv`.
2. Exploratory data analysis is implemented in `eda.ipynb`.
3. The code for fitting Sparse Grouping Pursuit is in `group.ipynb`.
4. The code for modeling, encoding, and predicting is in `models.ipynb`.
5. The functions utilized are in `group.py` and `models.py`.
6. The conda environment configuration file is `environment.yml`.
7. The data sets converted from the original excel files in the challenge are `train.csv` and `test.csv`.