PP19: Utilizing Machine Learning Algorithms to Detect Outliers or Inconsistencies in the Analysis Results in Datasets and Summary Reports

Linshan Yuan (LYuan7@ITS.JNJ.com), Weiwei Xu (wxu52@ITS.JNJ.com)

Abstract

Safety related summary tables (e.g., AE, LB, VS, EG) are a core part of the Clinical Study Report (CSR) and are usually lengthy reports when we display the statistical results by visits/time points. In this poster we will explain how we applied multiple ML methods to 1) detect the outliers (e.g., extreme values in small Ns) and 2) obtain data pattern related to the derivation of Treatment Emergent Adverse Event (TEAE) flag.

Introduction

Machine learning methods has become increasingly significant in many fields. There are increasingly more machine learning methods and applications. Their efficiency and intelligence contribute to improving our daily work and solving complex problems. In this poster, several machine learning methods are explored for the solutions: embedded abnormal outlier identification and important features/patterns extraction from an analysis.

Method 1: unsupervised machine learning methods

ML parameters	Unsupervised ML methods			
	Isolation forest	Local outlier factor	One class SVM	Elliptic envelope sklearn
contamination	0.1, 0.2,0.3	0.1, 0.2	/	0.1, 0.2, 0.3, 0.4
n_neighbors	/	Less than sample size -1	/	/
nu	/	/	0.1, 0.2	/

- Isolation forest: calculate the anomaly score based on mean anomaly score of the trees in the forest.
- Local outlier factor: assign anomaly score based on the local density by estimating distances between data points in the neighbors (k-nearest neighbors).
- One class SVM: calculate the density of points that positive region has means high density of points and negative region indicates small densities.
- Elliptic envelope sklearn: detecting outliers in a Gaussian distributed data.

Multiple unsupervised ML algorithms explored.

Load extracted small Ns to model and outliers are selected based on contamination level.

Case 1: Load extracted small Ns to model and outliers are selected based on contamination level. vsdata=[100,100,97,97,97,94,94,94,93,91,89,87,84,83,81,81,78,78,78,78,78,78,74,74,66,65,66,66,73,2,4,67,67,63,64,62,61,63,60,60,57,56,54]. They are vital signs summary table's small Ns by time point as: Baseline, Week 1 (pre, 30 min, 1 hour), 2, 3, 4…

	Unsupervised ML methods			
	Isolation forest	Local outlier factor	One class SVM	Elliptic envelope sklearn
Abnormal identified	cont.=0.1: [100, 100, 2, 4, 54] cont.=0.2: [100, 100, 2, 4, 57, 56, 54]	n_ne=10, cont.=0.1: [2, 4, 57, 56, 54] n_ne=20, cont.=0.1: [100, 100, 2, 4, 54]	Nu=0.1: [74, 74, 2, 4, 56, 54] Nu=0.2: [100, 100, 97, 97, 97, 2, 4, 57, 56, 54]	cont.=01: [100, 100, 2, 4, 54] cont.=0.2: [100, 100, 97, 97, 97, 2, 4, 56, 54]
Positive alarm	[2, 4]	[2, 4]	[2, 4]	[2, 4]

Case 2: Lab Cholesterol table's small Ns by each time point as: Baseline, Week 4, 8, 12, 24.

choldata=[98,90,84,86,82]

	Unsupervised ML methods			
	Isolation forest	Local outlier factor	One class SVM	Elliptic envelope sklearn
Abnormal identified	cont.=0.1: [98] cont.=0.2: [98] cont.=0.3: [98, 82]	n_ne=4, cont.=0.1: [90] n_ne=4, cont.=0.2: [90]	nu=0.1: [98, 90] nu=0.2: [98, 90]	cont.=01:[98] cont.=0.2: [98] cont.=0.3: [98, 82]
Positive alarm	[82]	[90]	[90]	[82]

Note: cont.=contamination, n_ne.=n_neighbor.

The abnormal elements identified in the middle of list should need to check further if there is positive alarm caused by the analysis issue or data problem.

Overall, from above cases, the methods of Local outlier factor and One class SVM could identify abnormal elements embedded in the list some of which help us trace back to the analysis issues in the analysis datasets.

Combined other checks with rule-based algorithms.

Specific rule-based algorithm can be applied as additional check to identify the outlier along with above ML methods:

Rule-based algorithm	Case 1: vital sign data	Case 2: lab cholesterol data
Higher than neighbors	[73] (negative alarm)	[86] (positive alarm)
Lower than neighbors	none	[84] (positive alarm)

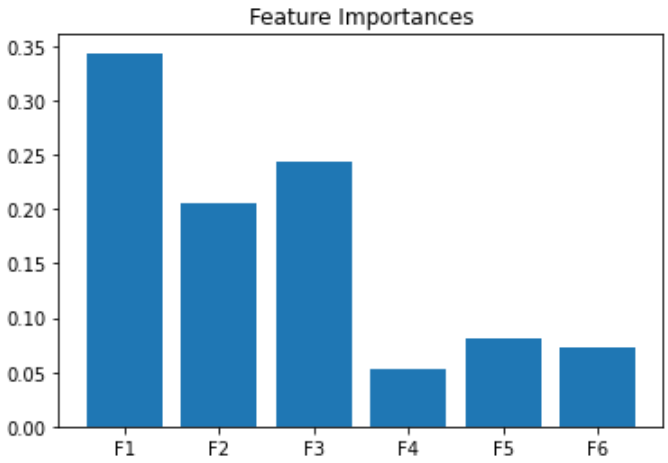
Different ML methods have different algorithms with diverse mathematical rules. Each of them has application limitation relying on the data nature and problem. Therefore, utilizing multiple ML methods along with rule-based algorithm is the solution to resolve the problem.

Method 2: decision tree

To get the pattern of AE start date, AE end date, treatment start date related to the derivation of TEAE flag. Turn AE dates relative to treatment start date into simple features.

- Performance evaluation: training data and testing data are split with 80% to 20%.
  Accuracy: 0.96
  Precision: 0.97

- Features obtained from the decision tree are listed in order of importance:



F1: 0.2849 (startyearcomp: 1 for AE start year > treatment start year, 0 for otherwise)
F2: 0.2026 (startdaycomp: 1 for AE start date > treatment start date, 0 for otherwise)
F3: 0.2019 (startmonthcomp: 1 for AE start month > treatment start month, 0 for otherwise)
F4: 0.1723 (enddaycomp: 1 for AE end date > treatment start date, 0 for otherwise)
F5: 0.0951 (endmonthcomp: 1 for AE end month > treatment start month, 0 for otherwise)
F6: 0.0432 (endyearcomp: 1 for AE end year > treatment start year, 0 for otherwise)

Above Important features are listed based on the score indicating its relative importance in the decision-making process of the tree. Higher scores suggest that the feature has a more substantial impact on the model's predictions.

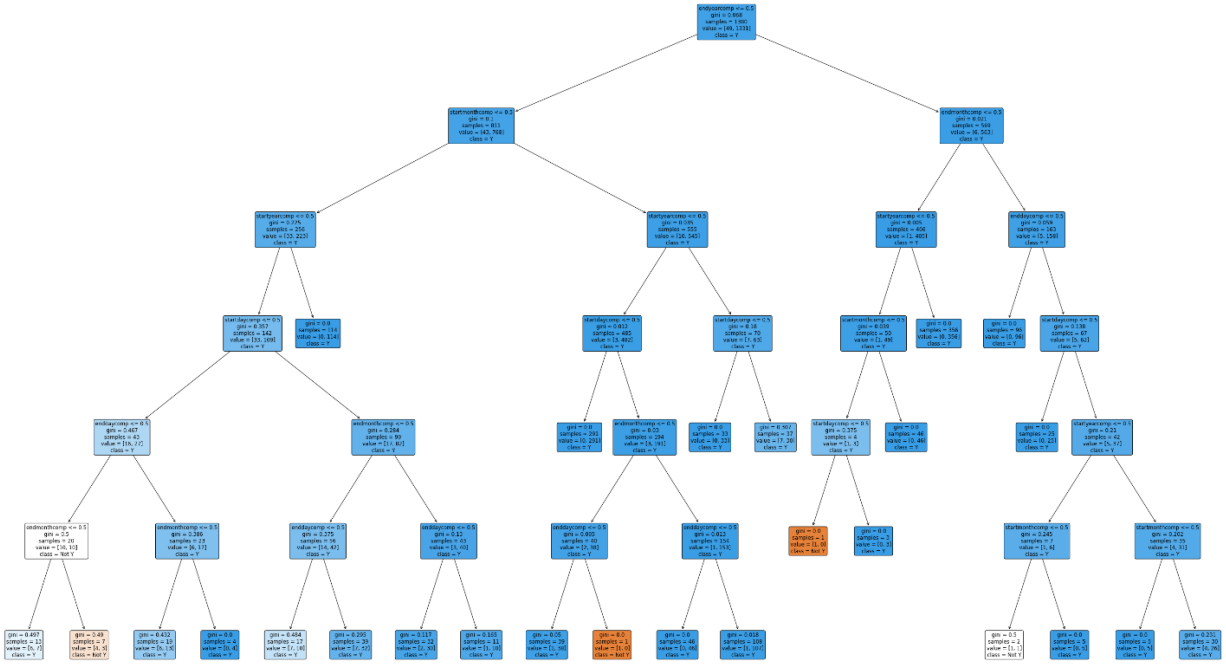
The important features F1, F2 and F3 extracted from decision tree show the consistency with our rule-based algorithm in the analysis. The application the decision tree method could help extract the important features from complex derivation cases as well.

- Extract the decision rule for one of important features: F1 as startyearcomp

```
Decision Rules for feature 'startdaycomp':
|--- endyearcomp <= 0.50
|   |--- startmonthcomp <= 0.50
|   |   |--- startyearcomp <= 0.50
|   |   |   |--- startdaycomp <= 0.50
|   |   |   |   |--- enddaycomp <= 0.50
|   |   |   |   |   |--- endmonthcomp <= 0.50
|   |   |   |   |   |   |--- weights: [6.00, 7.00] class: 1
|   |   |   |   |   |--- endmonthcomp >  0.50
|   |   |   |   |   |   |--- weights: [4.00, 3.00] class: 0
|   |   |   |   |--- enddaycomp >  0.50
|   |   |   |   |   |--- endmonthcomp <= 0.50
|   |   |   |   |   |   |--- weights: [6.00, 13.00] class: 1
|   |   |   |   |   |--- endmonthcomp >  0.50
|   |   |   |   |   |   |--- weights: [0.00, 4.00] class: 1
|   |   |   |--- startdaycomp >  0.50
|   |   |   |   |--- endmonthcomp <= 0.50
|   |   |   |   |   |--- enddaycomp <= 0.50
|   |   |   |   |   |   |--- weights: [7.00, 10.00] class: 1
|   |   |   |   |   |--- enddaycomp >  0.50
|   |   |   |   |   |   |--- weights: [7.00, 32.00] class: 1
|   |   |   |   |--- endmonthcomp >  0.50
|   |   |   |   |   |--- enddaycomp <= 0.50
|   |   |   |   |   |   |--- weights: [2.00, 30.00] class: 1
|   |   |   |   |   |--- enddaycomp >  0.50
|   |   |   |   |   |   |--- weights: [1.00, 10.00] class: 1
|   |   |--- startyearcomp >  0.50
|   |   |   |--- weights: [0.00, 114.00] class: 1
|   |--- startmonthcomp >  0.50
```

The above decision rule gives complex decision path. To improve it, more features or interaction features between date-related features could help. Additional tree pruning such as pre-pruning or post-pruning in the context of decision trees could be another exploratory to remove the nodes to create a simpler or more generalized tree that give better decision rule on new predicted data.

Attached decision tree plot:



Reference:

ML methods applied in the paper are using scikit-learn Python package.
Related details are available through https://scikit-learn.org/stable/modules.