작성자 : 정희철

논문 제목 : Anomaly Detection in High Dimensional Data

사용 데이터 : Porto Seguro

**데이터 개요**

본 데이터는 Porto Seguro라는 보험사에서 약 5년 전에 kaggle이라는 데이터분석 플랫폼에 공개한 데이터이다. 각 변수들이 정확히 무엇인지는 공개되어 있지 않고, 그저 데이터 유형을 파악할 수 있는 정도의 최소 정보만이 공개되어 있다. 해당 보험사의 설명에 따르면, ‘~bin’ 으로 끝나는 변수는 binary변수, ‘~cat’ 으로 끝난다면 nominal categorical 변수, 그 외는 연속형 변수이거나 ordinal categorical 변수이다. 또한, Y 변수는 target이라는 변수명을 가지고 있으며, 0 또는 1의 값을 갖는 binary 변수이다. 마지막으로, 모든 결측치는 -1의 값을 갖는다는 정보가 제공되었다.

**EDA**

먼저 binary 변수로 주어진 Y변수의 비율을 확인해보고, 단순한 binary classification 또는 Anomaly Detection 중 어떠한 문제로 접근하는게 좋을지 결정해야 했다. 0과 1의 비율을 시각화한 결과는 아래 그림 1로 제공되었다.

Chart, pie chart

Description automatically generated

그림 1 : 와 의 비율

Binary 변수인 Y변수에서 0과 1의 비율이 약 96대 4의 비율로 할당되어 있는 것을 확인하였고, class imbalance가 매우 심한 것을 확인하였다. 이 경우 모델 성능평가를 단순 accuracy로 할 경우, True Positive와 True Negative에만 의존하는 수식구조상, 단순히 0으로만 예측하여도 매우 높은 성능이 나오기 때문에 좋은 모델이 나오길 기대하기 어렵다. 이러한 이유로 class imbalance인 모델의 성능평가로 적합한 F1-score 또는 Normalized Gini Index를 사용하는 것이 적절할 것으로 생각하였다. 본 분석에서는 이상치라 하여도 소수 클래스인 1이 21694행이나 있다는 것을 고려하여 Normalized Gini Index에 비해 속도적으로 우월한 F1-score를 사용하기로 하였다. (test.csv파일은 Kaggle submission용이라 라벨링된 Y변수가 제공되지 않기에 train.csv파일만 사용하였다)

**변수 제거**

주어진 데이터의 결측치들을 보간해주기 위해, 이에 대한 시각화를 해보았다. 결측치는 -1로 표기되었다는 데이터 설명을 참고하여 -1을 모두 nan으로 변환해주고 시각화를 진행해본 결과, 그림 2와 같은 결과가 나왔다.

Chart

Description automatically generated

그림 2 : msno라이브러리를 사용한 결측치 시각화

확인해본 결과, ‘ps\_car\_03\_cat’변수와 ‘ps\_car\_05\_cat’변수는 결측치 비율이 40%를 상회하여 결측치 보간이 의미가 없을 것이라 생각하여 폐기하기로 결정하였다.

**EDA - 분포 차이 파악**

Porto Seguro측에서 시계열 데이터라는 언급이 없었기에 cross-sectional data라고 간주하고 진행하였다. 일 경우와 일 경우, 즉 정상치일 경우와 이상치일 경우 각 변수의 결측 비율의 차이가 있는지 확인해볼 필요가 있어 시각화해본 결과, 결측 비율이 유사함을 확인하였다 (그림 3, 4).

Chart

Description automatically generated

그림 3 : (정상치)일 경우의 결측치

Chart

Description automatically generated

그림 3 : (이상치)일 경우의 결측치

이제 결측인 행과 결측이 아닌 행의 각 변수의 분포 차이를 파악해볼 필요가 있었다. ‘~bin’이나 ‘~cat’으로 끝나지 않는 변수들은 고유값이 100개 아래인 변수를 ordina로 구분하였다. 아래 그림들은 정상치(주황)일 경우와 이상치(파랑)일 경우를 나눠 각 변수들의 분포 차이를 시각화한 결과들이다.

Diagram, PowerPoint

Description automatically generated with medium confidenceDiagram

Description automatically generated

그림 4 : Binary 그림 5 : Nominal Categorical

Calendar

Description automatically generated with low confidenceChart, histogram

Description automatically generated

그림 6 : Ordinal Categorical 그림 7 : Continuous

전체적으로 분포가 Y값에 독립적으로 유사한 것을 확인하였다. 통합적으로 고려해본 결과, 결측치들이 MCAR일 것으로 추정이 되므로, 본격적인 결측치 대체 전에 이에 활용할 수 있는 변수를 판별하기 위해 상관분석을 실시하였다. 연속형 변수는 Pearson’s Correlation을 사용하였고, 이 외의 데이터들은 Cramer’s V를 사용하여 상관관계를 알아보았다 (그림 8, 9, 10, 11). 분석 결과, 연속형 변수를 제외하면 같은 유형의 변수들간 상관관계가 낮은 모습을 확인하였다. 이러한 이유로 연속형 자료끼리는 선형모형을 사용하여 보간해주고 (Bayesian Ridge), 범주형 자료는 비선형모형을 사용하기로 판단하였다(Random Forest Classifier). A picture containing chart

Description automatically generatedChart

Description automatically generated

그림 8 : Continuous 그림 9 : Ordinal Categorical

Chart, waterfall chart

Description automatically generatedChart, waterfall chart

Description automatically generated

그림 10 : Binary 그림 11 : Nominal Categorical

**Modeling**

위의 과정들을 통해 전처리가 끝난 데이터들을 사용하여 모델링을 하였지만, 아무리 긴 시간동안 기다려봐도 모델링이 끝나지 않아 데이터를 축소하기로 결정하였다. 595212행 중 10%만을 사용하기로 결정하였다. 이 때, 10%를 sampling하면 이상치가 안 뽑힐 확률이 있으므로 stratified sampling을 사용하여 정상치와 이상치 각각 10%씩 sampling 될 수 있게 설정하였다. 그리고 10%중 20%를 다시 test set으로 설정하여 아래와 같은 결과들을 얻을 수 있었다.

**Modeling – Isolation Forest**

성능비교용 모델로 Isolation Forest를 선택하였다. 하이퍼 파라미터는 자동으로 최적의 하이퍼 파라미터를 찾아주는 Optuna 라이브러리를 사용하여 튜닝하였다. 최적의 파라미터와 위에 전처리된 데이터로 모델을 돌려본 결과, F1-score가 약 0.93인 모델을 얻을 수 있었다.

**Modeling – HDoutliers Algorithm**

모두 정상치로 분류되었다.

**Modeling – Stray Algorithm**

모두 정상치로 분류되었다.

**Conclusion**

HDoutliers Algorithm과 Stray Algorithm은 unsupervised learning algorithm으로 Isolation Forest와 동일한 방법으로 성능을 평가하기엔 무리가 있다. 하지만 2개의 알고리즘 모두 그 어떤 이상치를 분류해내지 못했다는 사실에 의아했다. 특히, Anomalous Threshold를 보수적으로 설정하여 어느 정도 수준의 False Negative(정상치를 이상치로 분류)를 감수하는 Stray Algorithm의 경우에도 단 하나도 이상치로 분류하지 않아 매우 의아했다.

데이터 자체가 고차원이라 모든 변수에 대한 시각화를 이용하여 이에 대한 원인을 파악할 수 없었다. 그리고 논문의 저자 역시 논문에 사용된 데이터의 다양성이 부족하고, 아직 어떠한 문제가 있는지 명시하지 못했음을 인지하였다. 그러나 원인에 대해 짐작해보자면, HDoutliers Algorithm과 Stray Algorithm은 모두 거리에 기반하여 클러스터링을 한다. 이는 결국 이상치는 정상치 클러스터에서 거리가 떨어져 있어야 하고, 이상치 자체끼리 역시 어느 정도는 클러스터를 형성할 수 있어야 할 것이다.

이상치 클러스터와 정상치 클러스터가 같은 면을 고려하고 있다고 가정하면(이러한 경우, radius를 사용하는 HDoutliers Algorithm 역시 효율적이지 못할 것이다), KNN을 사용하는 Stray Algorithm의 특성상 이를 탐지하기는 어려울 것으로 생각된다. 또한, 만약 이상치 클러스터가 정상치 클러스터 내에 있다면, 거리기반 알고리즘들은 당연히 탐지하지 못한다. 마지막으로, 가장 기본적으로, 만약 고차원일지라도 이상치와 정상치 간의 거리 차를 잡아낼 수 있는 변수 자체가 없다면, 당연히 모델 성능이 좋을 수 없다.