운전자 정보 및 자동차 정보에 따른 자동차 사고 보험 예측

[데이터 마이닝 팀 프로젝트]

산업정보시스템 공학과	
20182471	김지민
20201362	조인영

목차

1. 서론.....3
1.1 연구소개.....3
1.2 연구동기.....3
1.3 연구필요성.....3
1.4 연구목표.....3

2. 관련연구.....3
2.1 자동차 보험 시장의 마케팅 전략 지향점.....3
2.2 Research on the Features of Car Insurance Data Based on Machine Learning.....4
2.3 머신러닝을 이용한 자동차 보험 청구 예측(Motor Insurance Claim Status Prediction using Machine Learning Techniques)....4

3. 본론.....5
3.1 데이터 소개.....5
3.2 데이터 전처리 및 시각화.....6
3.3 활용 알고리즘.....11
3.3.1 Neural Network.....11
3.3.2 Gradient Boosting.....12
3.3.3 XGBoost.....12
3.3.4 Ada Boosting.....12
3.3.5 Decision Tree.....13
3.3.6 Logistic Regression.....13
3.3.7 KNN.....13
3.3.8 Random Forest.....13

4. 평가 및 분석.....14
4.1 모델평가(Neural Network 외 7개).....14
4.2 결과분석.....15

5. 결론.....18
5.1 분석의 문제점 및 한계.....18
5.2 향후 발전 방향.....19

6. 참고문헌.....19

1.서론

1.1 연구 소개

오늘날 머신러닝 알고리즘을 활용해 고객들의 소비 패턴을 분석하여 고객 맞춤화 콘텐츠를 창출하는 기업이 늘어나고 있다. 고객 맞춤화 콘텐츠를 통해 고객들은 자신의 요구사항을 만족시킬 수 있고 기업은 이를 통해 고객의 충성도를 높일 수 있다. 이와 같이 자동차 보험회사도 머신러닝 기법을 활용해 운전자의 자동차 보험 청구 신청을 예측해 고객에게 합리적인 보험을 청구할 수 있다. 따라서 본 연구에서는 운전자의 보험 청구 신청여부를 정확하게 예측하기 위해 좋은 성능을 이끌어내는 머신러닝 모델을 개발하고자 한다.

1.2 연구 동기

머신러닝 기법이 발전함에 따라 기업들은 전략적인 마케팅 전략을 세울 필요가 있다. 자동차 보험 회사는 고객의 흥미를 끌만한 마케팅 전략을 세우기 위해 운전자와 자동차의 어떤 요소들이 보험 청구 확률과 연관이 있는지 파악해야 한다.

1.3 연구 필요성

기업은 신규 고객에 대한 마케팅 전략을 짜는 것도 중요하지만 기존 고객의 이탈을 방지하는 것이 더 중요하다. 고객과의 관계 유지에 따른 비용은 기간이 증가할수록 감소하지만 고객 한 사람을 유지하는데 따른 수익성은 시간이 흐를수록 급속히 증가하기 때문이다. 따라서 기존 고객의 이탈방지를 위한 기존 고객 맞춤형 상품을 창출해 내야한다. 이를 위해 자동차 보험 회사는 운전자와 자동차의 많은 정보들 중 어떤 것이 보험 청구에 가장 영향을 미치는지 파악하고 보험 청구 여부를 정확하게 예측해야할 필요가 있다.

1.4 연구 목표

운전자의 정보 중 어떤 요소들이 보험 청구 여부와 상관관계가 높은지 파악하고 운전자가 내년에 보험 청구를 시작할 확률을 정확하게 예측해내는 모델을 구축하고자한다.

2. 관련연구

자동차 보험 시장에 대한 기존연구의 논문 위주로 찾아보았다. 또한 비슷한 주제를 가진 연구들을 찾아보며 다양한 성능 좋은 머신러닝 분류 모델을 탐색했다.

2.1 자동차 보험 시장의 마케팅 전략 지향점 연구

이 연구 목표는 자동차 보험 시장의 외부환경 및 경쟁수단 분석을 통해 손해보험회사들이 현 시점에서 자동차 보험시장 마케팅 전략을 수행하는데 필요한 전략방향을 도출하는 것이다. 연구에 따르면 우리나라 자동차 보험시장은 성숙기 시장이거나, 성숙기 시장으로 진입하고 있는 시장으로 판단된다. 또한 자동차대수 증가와 밀접한 관계가 있는 대당 인구수를 선진국과 비교할 경우, 성장잠재력이 큰 것으로 판단되지만 인구감소추이, 고령화 비율 증가, 경제 성장률 수준, 온라인 자동차보험 시장의 규모 증가 등을 종합적으로 고려할 경우, 자동차 보험 시장의 성장성은 크지 않은 것으로 판단된다. 자동차보험시장이 성숙기에 접어들었다면, 고객이 다음 계약에서도 자기 회사를 선택하도록 고객의 충성도를 높이는 전략이 향후 마케팅 전략의 중심이 되어야 한다는 것이다. 향후 자동차 보험시장은 상품차별화 전략과 가격전략이 필요한

지를 고려하고 다양한 마케팅 믹스전략을 고려해야한다.

2.2 Research on the Features of Car Insurance Data Based on Machine Learning

이 연구에서는 자동차 보험 데이터의 특징을 분석하고, 자동차 보험 가입 여부에 영향을 미치는 가장 중요한 특징을 추출한다. 또한 Random forest(RF), gradient lifting tree(GBDT), lifting machine algorithm(Light BGM) 모델을 비교한다. 이 연구에 사용된 데이터에는 자동차 브랜드, 보험 재산, 보험 금액, 위험범주, 자동차 연령, 피보험자의 나이 등 운전자에 대한 개인적인 정보, 자동차에 대한 정보, 운전자의 과거 보험에 대한 정보들과 관련된 28개의 변수들이 합쳐져 총 65535데이터들이 있다.

*Model Building

기계 학습 알고리즘은 일반적으로 mini batch 방식으로 훈련되며 훈련 데이터의 크기는 메모리에 의해 제한되지 않는다. GBDT 알고리즘은 각 반복에서 전체 train data가 여러 번 순회해야한다. GBDT가 대용량 데이터를 다룰 때 직면하는 문제를 해결하기 위해 Ligth BGM이 제시된다. Light BGM은 학습 알고리즘 기반 의사결정 트리를 사용하여 더 빠른 학습 속도, 더 낮은 메모리소비, 더 높은 정확도 및 대용량 데이터의 빠른 처리로 효율적인 병렬학습을 지원하는 Grdient Boosting framework이다.

*결과

여러 모델 중 Light GBM의 성능이 가장 좋았다. 또한 여러 변수들 중 car insurance business channel, NCD, car age, new car purchase price이 자동차 보험 가입 여부에 가장 큰 영향을 미쳤다.

2.3 머신러닝을 이용한 자동차 보험 청구 예측(Motor Insurance Claim Status Prediction using Machine Learning Techniques)

이 연구의 목표는 컴퓨터의 발전과 더불어 머신러닝 기법을 이용해 보험 회사의 입장에서 자동차 보험 청구비용 손실을 줄이기 위해 실행된 연구이다. 연구의 목적에 따라 쓰는 머신러닝 모델이 다르다는 점을 확인할 수 있었다. 우리가 진행하고자 하는 프로젝트의 목표인 운전자의 다음연도 자동차 보험 가입 여부에 대한 예측이란 목적에 대해서는 Decision Tree,K−Nearest Neighbors Naïve Bayes, Neural Networks, and Support vector machine algorithms을 주로 사용한다는 점을 알 수 있었다. 해당 연구를 통해 자동차 보험 영역에서 Random forest가 svm 보다 더 좋은 성능을 보인다는 결론을 얻을 수 있었다. 하지만 우리가 진행하는 프로젝트에서도 동일한 결과가 나올지에 대해서 한 번 더 확인해 볼 필요성을 느꼈다.

3.본론

3.1 데이터 소개

본 데이터는 운전자의 운전 정보와 자동차의 상태를 나타내는 정보로 총 44개의 열과 58592개의 행으로 구성되어있는 데이터이다. 수치형 데이터 16개, 범주형 데이터 28개로 구성되어 있는 데이터로 데이터에 대한 간단한 설명을 담은 표는 밑의 표와 같다.

데이터에 대한 간단한 설명

NO.	Attribute	Explanation
1.	policy ID	보험 증서 ID
2.	Policy tenure	보험증서 남은 기간
3.	age of car	차의 나이 (정규화된 값)
4.	age of policy holder	보험계약자의 나이 (정규화된 값)
5.	area cluster	보험계약자 거주지
6.	population density	보험계약자 거주지에 대한 인구 밀집도
7.	make	차량 제조회사 (encoding 된 값)
8.	segment	차의 종류 ex) Small sedans, Family cars,
9.	model	차량 모델 (encoding 된 값)
10.	fuel type	연료 타입
11.	max torque	최대 회전력 (nm rpm)
12.	max power	최대 마력 (bhp rpm)
13.	engine type	엔진 타입
14.	airbags	에어백 개수
15.	in esc	전자 제어 주행 안정 장치
16.	is adjustable steering	운전대 조정가능 여부
17.	is tpms	자동차 타이어 압력 센서 여부
18.	is parking sensor	주차센서의 여부
19.	is parking camera	주차카메라의 여부
20.	rear brakes type	차 뒷바퀴에 사용된 브레이크의 종류
21.	displacement	엔진 배기량 (cc)
22.	cylinder	차 엔진 앞쪽의 실린더 개수
23.	transmission type	변속기 종류
24.	gear box	기어의 개수
25.	steering type	파워 스티어링 설치 타입
26.	turning radius	차량 회전 직경
27.	length	차량 길이
28.	width	차량 폭
29.	height	차량 높이
30.	gross weight	총 중량
31.	is front fog lights	전면 안개등 사용 가능 여부
32.	is rear window wiper	후방 창문 와이퍼 사용가능 여부
33.	is window washer	창문 워셔의 사용가능 여부
34.	is rear window defogger	서리제거장치 존재 여부

35.	is brake assist	브레이크 어시스트 시스템의 사용가능 여부
36.	is power door lick	파워 도어 잠금장치 사용가능 여부
37.	is central locking	중앙 잠금 장치 사용가능 여부
38.	is power steering	파워 스티어링 사용 가능 여부
39.	is driver seat height adjustable	운전자 좌석 높이 조절 가능 여부
40.	is day night rear view mirror	주야간 거울 존재 여부
41.	is ecw	엔진 체크 경고등의 사용가능 여부
42.	is speed alert	스피드 경고 시스템의 사용가능 여부
43.	ncap rating	유럽의 자동차 안전 평가 프로그램 등급
44.	is claim	6개월 이내 보험청구 여부 (Target value)

- **용어에 대한 설명**

① power steering : 자동차의 핸들 조작을 쉽게 하기 위한 자동차의 장치 일종.

②turning radius: 차량의 회전 직경. 차량이 회전하는데 필요한 최소 직경을 말한다.

③ day night rear view mirror : '주야간 거울'이라고 하는 뒷 차의 헤드라이트의 눈부심 방지 기능이 포함된 거울을 말한다.

[그림] power steering

④esc : 전자 제어 주행 안전 장치. 차량의 안정성을 향상시키고 미끄러짐을 예방하기 위한 능동안전기술을 말한다.

3.2 데이터 전처리 및 시각화

3.2.1 결측치 처리

이번 데이터는 결측치가 없는 것으로 확인되어 데이터 그대로 사용할 수 있었다.

```
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   policy_id                     58592 non-null  object
 1   policy_tenure                 58592 non-null  float64
 2   age_of_car                    58592 non-null  float64
 3   age_of_policyholder           58592 non-null  float64
 4   area_cluster                  58592 non-null  object
 5   population_density            58592 non-null  int64
 6   make                          58592 non-null  int64
 7   segment                       58592 non-null  object
 8   model                         58592 non-null  object
 9   fuel_type                     58592 non-null  object
 10  max_torque                    58592 non-null  object
 11  max_power                     58592 non-null  object
 12  engine_type                   58592 non-null  object
 13  airbags                       58592 non-null  int64
 14  is_esc                        58592 non-null  object
 15  is_adjustable_steering        58592 non-null  object
 16  is_tpms                       58592 non-null  object
 17  is_parking_sensors            58592 non-null  object
 18  is_parking_camera             58592 non-null  object
 19  rear_brakes_type              58592 non-null  object
 20  displacement                  58592 non-null  int64
 21  cylinder                      58592 non-null  int64
 22  transmission_type             58592 non-null  object
 23  gear_box                      58592 non-null  int64
 24  steering_type                 58592 non-null  object
 25  turning_radius                58592 non-null  float64
 26  length                        58592 non-null  int64
 27  width                         58592 non-null  int64
 28  height                        58592 non-null  int64
 29  gross_weight                  58592 non-null  int64
 30  is_front_fog_lights           58592 non-null  object
 31  is_rear_window_wiper          58592 non-null  object
 32  is_rear_window_washer         58592 non-null  object
 33  is_rear_window_defogger       58592 non-null  object
 34  is_brake_assist               58592 non-null  object
 35  is_power_door_locks           58592 non-null  object
 36  is_central_locking            58592 non-null  object
 37  is_power_steering             58592 non-null  object
 38  is_driver_seat_height_adjustable  58592 non-null  object
 39  is_day_night_rear_view_mirror  58592 non-null  object
 40  is_ecw                        58592 non-null  object
 41  is_speed_alert                58592 non-null  object
 42  ncap_rating                   58592 non-null  int64
 43  is_claim                      58592 non-null  int64
```

3.2.2 Label Encoding

데이터를 사용하기 위해 범주형 변수를 수치형 변수로 바꾸어주었다.

is_central_locking	is_power_steering	is_driver_seat_height_adjustable	is_day_night_rear_view_mirror	is_ecw	is_speed_alert	ncap_rating
0	1	0	0	0	1	0
0	1	0	0	0	1	0
0	1	0	0	0	1	0
1	1	1	1	1	1	2
1	1	0	1	1	1	2

3.2.3 변수선택

데이터에 target 값을 제외한 변수들의 개수는 43개였다 attribute 개수가 많아지게 되면 그들끼리의 상관관계가 높아질 수 있어 target 값을 예측하는데 방해가 될 수 있으며 이는 모델의 성능 저하로 이어지게 된다. 따라서 attribute 개수를 줄이기로 결정하였고 주성분분석과 k-best selection 방법을 사용했다.

- **주성분분석(PCA)**

주성분분석은 가장 널리 사용되는 차원 축소 기법 중 하나로, 원 데이터의 분포를 최대한 보존하면서 고차원 공간의 데이터들을 저차원 공간으로 변환하는 기법이다. PCA는 기존의 변수를 조합하여 서로 연관성이 없는 새로운 변수, 즉 주성분들을 만들어 내는 것이다. 주성분분석의 개수를 10개로 지정하고 각각의 주성분의 기여율을 표로 정리하면 다음과 같다.

	설명가능한 분산 비율(고윳값)	기여율	누적기여율
pca1	19.256794	0.458487	0.458487
pca2	8.241030	0.196212	0.654699
pca3	2.539814	0.060471	0.715170
pca4	2.270465	0.054058	0.769227
pca5	1.541541	0.036703	0.805930
pca6	1.323057	0.031501	0.837431
pca7	1.161504	0.027654	0.865085
pca8	1.125472	0.026796	0.891882
pca9	1.058706	0.025207	0.917089
pca10	0.828582	0.019728	0.936817

하지만 제1 주성분의 기여율이 약 0.46정도 밖에 되지 않으며 그 이후에는 기여율의 값이 매우 작았다. 또한 주성분 분석을 사용하게 되면 순수한 변수의 특성을 잃는 것이기 때문에 주성분 분석을 이용한 차원 축소는 고려해볼 필요가 있었다.

- Select K-Best

Select K-best는 scikit-learn에서 제공하는 모듈로 target변수와 그 외 attribute간의

상관관계를 계산하여 k개의 가장 높은 상관관계를 가지는 변수를 선택해주는 모듈이다. 상관관계를 분석하는 방법에는 f-Regression, 카이제곱방식 (chi2)이 있다. f-Regression의 경우 일반적으로 회귀문제에서 사용되고 카이제곱방식의 경우 일반적으로 분류 문제에서 사용된다. 이번 프로젝트의 경우 classification에 해당하는 문제이기 때문에 카이제곱방식을 사용하기로 결정하였다. 너무 많은 데이터의 손실을 막기 위해 20개의 attribute를 선택하기로 하였고 결과는 다음과 같다.

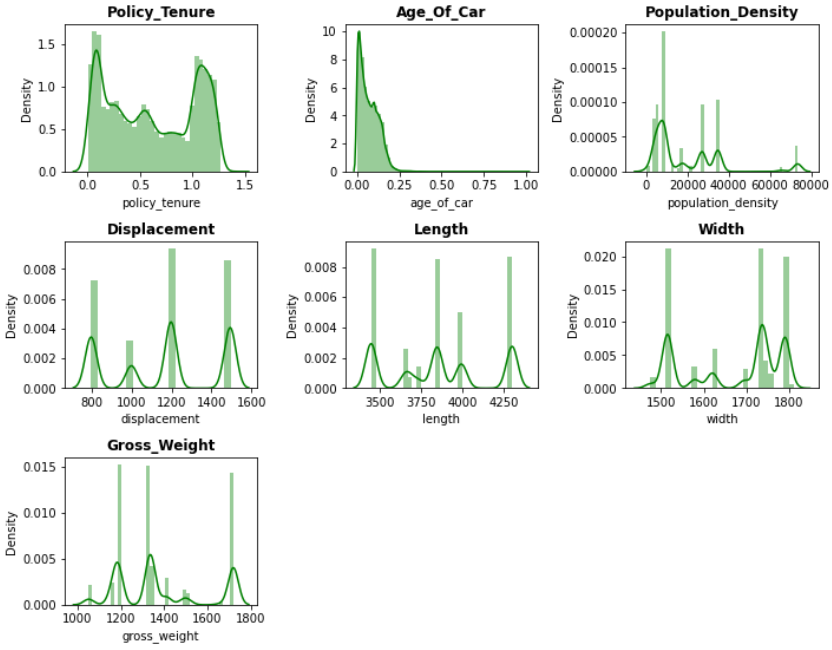
```
Selected naems:  Index(['policy_id', 'policy_tenure', 'age_of_car', 'area_cluster',
       'population_density', 'segment', 'model', 'fuel_type', 'max_torque',
       'max_power', 'is_adjustable_steering', 'displacement', 'steering_type',
       'length', 'width', 'gross_weight', 'is_front_fog_lights',
       'is_brake_assist', 'is_driver_seat_height_adjustable',
       'is_day_night_rear_view_mirror'],
      dtype='object')
```

모델의 성능을 향상시킨 변수 선택법을 택하고자 하였고 결론적으로 k-best select모듈을 통한 변수를 선택할 수 있었다.

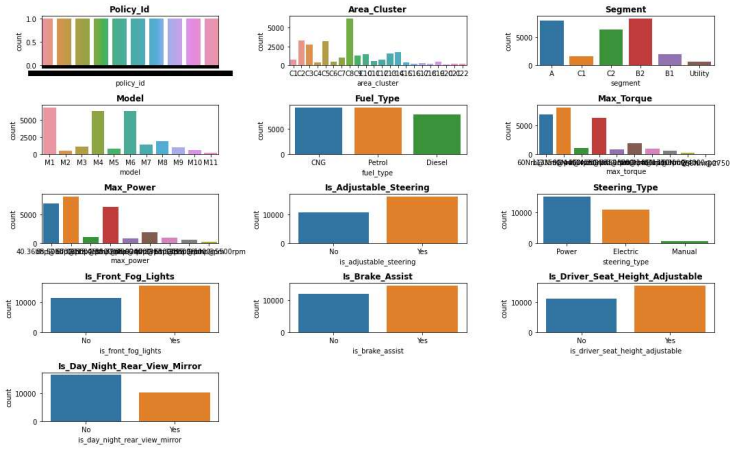
3.2.4 EDA

좋은 모델을 개발하기 위해서는 데이터의 이해가 필수적이다. 보다 더 정확한 데이터의 이해를 위해 여러 방식을 통해 시각화를 했다.

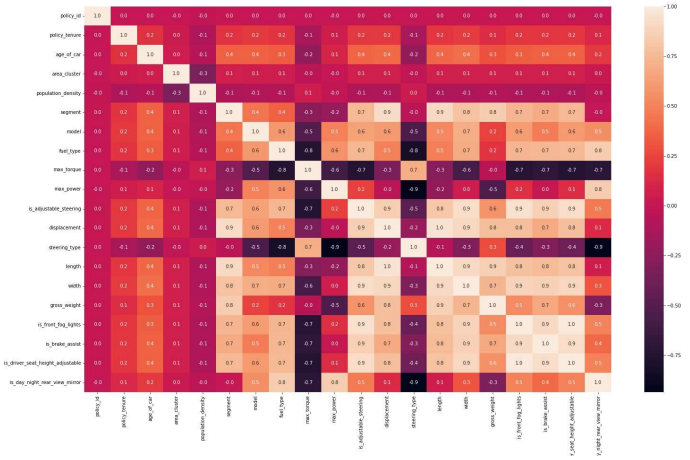
- **선택된 수치형 데이터 분포**



- 선택된 범주형 데이터 분포

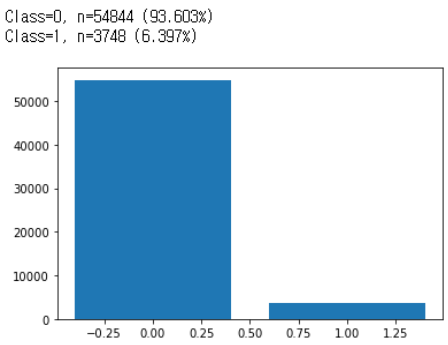


- Correlation Matrix



3.2.5 데이터 불균형해소

target변수에 해당하는 보험 청구는 현실에서 보험을 청구하는 인원보다 청구하지 않는 인원이 훨씬 많다. 해당 데이터의 target변수의 0,1 개수를 plot을 통해 확인 할 수 있었다.

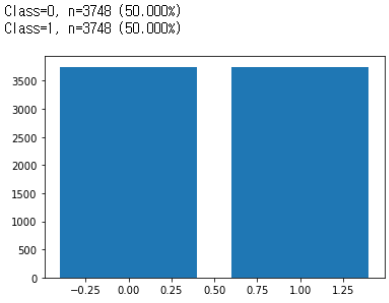


원 데이터에 대한 target count plot

target변수의 1의 값이 54844개, 0의 값이 3748개로 보험을 청구하는 사람은 데이터의 6.397%밖에 차지하지 않았다. 이렇게 데이터가 불균형한 분포를 가지는 경우, 모델의 학습이 제대로 이루어지지 않을 확률이 높기 때문에 undersampling과 oversampling 방식을 이용해 해결해보았다.

- **undersampling**

undersampling은 불균형한 데이터에서 높은 비율을 차지하던 class의 데이터 수를 줄임으로써 데이터 불균형을 해소하는 아이디어이다. 하지만 이 방법은 학습에 사용되는 전체 데이터 수를 급격하게 감소시켜 좋은 질의 데이터 개수를 줄인다는 점과 오히려 모델의 성능이 떨어질 수 있다는 단점이 존재한다.



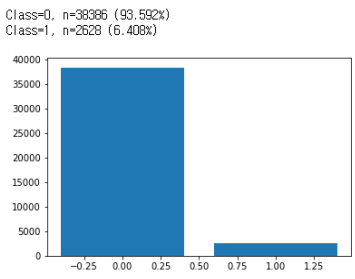
undersampling 이후 target data

다음과 같이 데이터 불균형을 해소해주었고 이 데이터를 바탕으로 분석을 진행해보았다. 약 6만개의 데이터에서 4000개로 데이터의 양을 급격하게 줄여 모델의 성능이 우려되었지만 데이터의 불균형을 undersampling으로 해소하고 난 후 모델의 성능이 좋아진 것을 확인할 수 있었다.

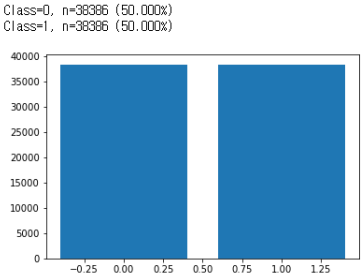
- **oversampling**

oversampling은 undesampling과 반대로 낮은 비율 class의 데이터 수를 늘림으로써 데이터 불균형을 해소하는 아이디어이다. 우리는 대표적인 oversampling 방법인 smote개념을 이용해 oversampling을 진행해보았다.

smote는 낮은 비율 class데이터들의 최근접 이웃을 이용하여 새로운 데이터를 생성하는 방식이다. 완전히 똑같은 특성을 가진 데이터를 복사하는 것은 의미가 없기 때문에, 근접해 있는 데이터들과 일정한 거리에 떨어진 위치에 데이터를 생성하는 것이다. smote 개념을 적용하기 위해 test set과 train set으로 나눈 후 train set에 대해서 smote를 적용하였다.



train set



smote를 적용한 후 train set

위의 사진과 같이 smote를 진행해준 후 여러 모델들에 적용시켰다.
oversampling을 통해 모델을 구현한 결과 undersampling을 이용했을 때 보다 여러 모델에 대해서 성능이 현저히 낮아졌다.

3.2.6 data scaling

데이터 스케일링을 하기 이전 데이터를 test set과 train set으로 데이터를 분류해 줄 필요가 있다. 앞서 oversampling한 데이터는 test set과 train set으로 분할되었지만 undersampling한 데이터는 그렇지 않아 분할을 추가로 진행하였다.
이후 가장 보편적인 Min-Max scaling을 사용해 스케일링을 진행하였다.

Min-Max scaling에 대한 수식은 다음과 같다. $z_i = \frac{X_i - X_{\min}}{X_{\max} - X_{\min}}$

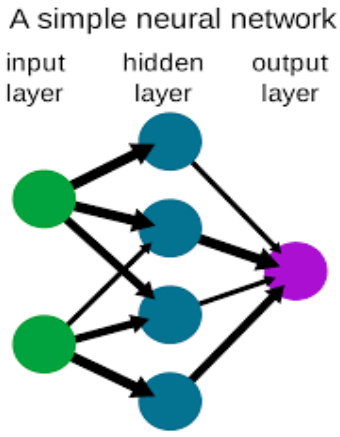
앞으로 사용할 모델에 스케일링 된 train set을 학습시키고 test set에 대해서 예측을 진행하였다.

3.3 활용 알고리즘

이번 프로젝트를 진행하는데 있어 지도학습 데이터를 사용하기 때문에 classifier 방식의 모델을 많이 사용하였다. 그 중 Neural Network, XGboost, Decition tree, Logistic regression, Support Vector Machine, Gradient boosting, Random forest, ADA Boost 을 사용하기로 결정 하였다. 다음으로 알고리즘에 대한 간단하게 알아보았다.

3.3.1 Neural Network

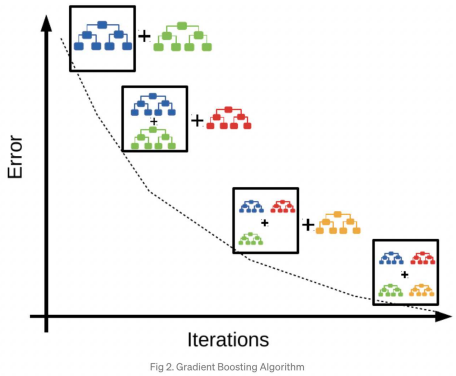
인공신경망은 사람의 두뇌의 신경세포를 모방하여 수학적으로 모델링한 모델이다. 데이터를 학습시켜 분류 및 최적화를 하는 머신러닝의 방법이다. input layer, hidden layer, output layer로 구성되어 있으며 여러 가지 인공신경망 종류가 있다. 각 노드들은 다른 노드들에 연결이 되어있고 노드가 일정 임계치를 넘으면 다음 계층으로 데이터를 보내는 구조이다. 해당 신경망은 시간이 지남에 따라 모델을 학습, 개선해 나간다. NN의 장점으로는 범주형, 수치형 변수에 상관없이 분석이 가능하단 점이 있고, data의 양이 많아질수록 모델의 성능이 좋아진다. 하지만 hidden layer의 개수에 따라서 결과 값이 많이 달라질 수 있기에 hidden layer의 개수를 최적화 할 필요가 있고 학습에 오랜 시간이 걸리는 단점이 있다.



[simplified view of artificial nerual network]

3.3.2 Gradient Boosting

Gradient Boosting알고리즘은 분류분석 또는 회귀분석을 수행할 수 있는 예측모형이며 예측모형의 앙상블 방법론 중 Boosting계열에 속하는 알고리즘이다. Gradient Boosting은 Tabular format 데이터에 대한 예측에서 엄청난 성능을 보여준다. Gradient Boosting을 이해하는 가장 쉬운 방법은 Residual fitting으로 이해하는 것이다. 아주 간단한 모델 A를 통해 y를 예측하고 남은 잔차를 다시 B라는 모델을 통해 예측하고 A+B모델을 통해 y를 예측한다면 A보다 나은 B모델을 만들 수 있게 되는 것이다. 이러한 방법을 계속하면 잔차는 계속해서 줄어들게 되고 training set을 잘 설명하는 예측모형을 만들 수 있게 된다. 하지만 이러한 방식은 bias는 줄일 수 있어도 overfitting이 일어날 수 있다는 단점이 있다.
Boosting 기법을 이용하여 구현한 알고리즘은 Gradient Boost가 대표적인데, 이 알고리즘을 병렬 학습이 지원되도록 구현한 라이브러리가 XGBoost이다.



[Gradient Boosting for classification]

3.3.3 XGBoost

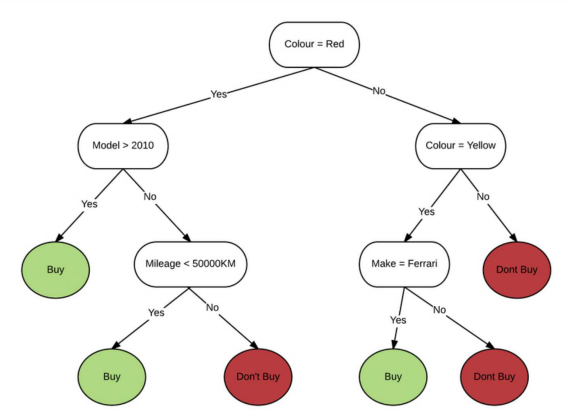
XGBoost란 Extreme gradient boosting의 약자로 Gradient boost 알고리즘을 병렬 학습이 가능하도록 구현한 알고리즘이다. XGBoost의 기본 개념인 Boosting은 여러 개의 약한 분류기 (Decision tree)를 조합해서 정확도를 예측하는 방법을 일컫는다. 간단한 설명을 곁들이자면 xgboost는 greedy 알고리즘을 사용해 분류기 M,G,H를 발견, 분산 처리를 이용해 빠르게 적합한 비중 파라미터를 찾는다. 또한 xgboost는 트리를 만들 때 CART(classification and regression Tree)라는 앙상블 모델을 사용한다. XGboost는 분류,회귀 문제를 모두 지원하고 성능이 좋아 대중적으로 사용되는 알고리즘 모델이다.
장점으로는 빠른 수행시간, overfitting 방지가 있다. 하지만 적은양의 데이터에 대해서 과적합이 일어날 수 있기 때문에 해당 부분을 조심해야 한다.

3.3.4 ADA Boosting

Ada Boosting은 adaptive boosting로 간단한 약분류기들이 상호 보완하도록 단계적으로 학습하고 이들을 조합하여 최종 강분류기의 성능을 증폭시키는 방식이다. 이전 분류기가 잘못 분류한 샘플의 가중치를 adaptive하게 바꾸어가며 잘못 분류되는 데이터에 더 집중하여 잘 학습하고 분류할 수 있도록 한다. 최종 분류기는 개별 약분류기들에게 각각 가중치를 적용, 조합하여 얻을 수 있다.

3.3.5 Decision Tree

Decision Tree는 우리가 흔히 아는 의사결정트리 모델이다. 지도학습 방법의 하나로 데이터의 특성에서 간단한 결정 규칙을 정해 값을 예측하는 모델이다. 의사결정트리 모델은 해석이 간편하며 나무를 시각화 할 수 있고 범주형 데이터를 모두 처리할 수 있는 장점이 있다. 하지만 현재 scikit-learn 에서는 범주형 데이터는 지원하지 않는다. 단점으로는 잘못하면 트리가 너무 복잡해져 과적합이 일어날 수 있다는 점이 있다.



[Example of Decision Tree model]

3.3.6 Logistic Regression

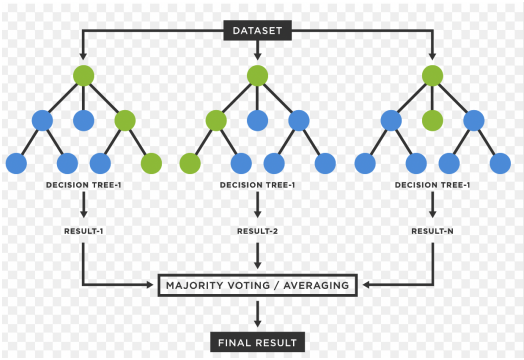
Logistic 회귀분석은 종속변수와 독립 변수간의 관계를 구체적인 함수로 나타내어 향후 예측 모델에 사용하는 것이다. 하지만 일반 선형 회귀 분석과 다른 점은 종속 변수가 범주형 데이터를 대상으로 한다는 점이다. 회귀 분석이지만 일종의 분류 기법이라고 볼 수 있다.

3.3.7 KNN

knn은 K-Nearest Neighbor의 약자로 지도 학습 알고리즘 중 하나이다. knn의 알고리즘 방식은 어떤 데이터가 주어지면 그 주변의 데이터를 살펴본 뒤 더 많은 데이터가 포함되어 있는 범주로 분류하는 방식이다. 새로운 데이터가 주어질 때 바로 분류를 시작하는 것이라 knn 훈련이 따로 필요 없다는 것이 knn의 특징이다. knn의 데이터들끼리의 거리를 구해야한다. 거리를 구하는 방식은 대표적으로 유클리디안 방식과 맨해튼방식이 있다.

3.3.8 Random Forest

Decision tree는 overfitting될 가능성이 높다는 약점을 가지고 있다. 가지치기를 통해 트리의 최대 높이를 설정해 줄 수 있지만 이로써는 overfitting을 충분히 해결할 수 없다. Random forest는 decision tree에서 좀 더 일반화된 트리를 만드는 기법이다. Random forest는 ensemble machine learning 모델로 여러 개의 decision tree를 형성하고 새로운 데이터 포인트를 각 트리에 동시에 통과시키며, 각 트리가 분류한 결과에서 투표를 실시하여 가장 많이 득표한 결과를 최종 분류 결과로 선택한다. 랜덤포레스트가 생성한 일부 트리는 overfitting될 수 있지만, 많은 수의 트리를 생성함으로써 overfitting이 예측하는데에 있어 큰 영향을 미치지 못하도록 예방한다.



[Random Forest for classification]

4. 평가 및 분석

4.1 모델평가

이번 프로젝트에서 모델의 성능을 평가할 때 confusion matrix를 이용한 f1 score를 평가척도로 결정했다. 평가척도에는 전체 예측 건수에서 정답을 맞힌 True 건수의 비율을 나타내는 값인 accuracy도 있다. 하지만 본 프로젝트에 사용되는 데이터는 target변수가 매우 불균형했다. 보험에 가입한 사람의 수는 3748명으로 데이터의 6.397%밖에 차지하지 않았다. 보험에 가입하지 않은 사람이 거의 없기 때문에 true positive는 하나도 없을 수 있고 true negative의 값만 매우 커질 수 있다.

이에 실제 데이터에 Negative 비율이 너무 높아서 희박한 가능성으로 발생할 상황에 대해 제대로 된 분류를 해주는지 평가해줄 지표인 precision과 recall의 조화평균인 f1 score를 평가지표로 정했다. precision은 positive로 예측한 데이터 중 실제 positive를 찾아낸 비율이고 recall은 실제 positive 중 올바르게 positive를 예측해낸 비율이다. precision, recall, f1-score의 수식은 다음과 같다.

$$\text{precision} = \frac{TP}{TP+FP}, \quad \text{Recall} = \frac{TP}{TP+FN}, \quad \text{F1 score} = (2\text{xprecision x recall})/(\text{precision}+\text{recall})$$

4.1.1 모델 성능비교

여러 가지 전처리 방식을 통해 8가지의 모델을 돌리고 가장 좋은 성능을 이끌어내는 전처리 방식, 모델을 최종적으로 선택했다. 변수 차원 축소를 주성분 분석, Select-K-Best 방식으로 진행하고 데이터 불균형에 대한 문제는 oversampling, undersampling 모두 진행해 총 3가지 전처리 방식 조합을 시도해보았다.

- **주성분 분석, Undersampling**

```
The test f1 score of Neural Network is 0.575811471765229
The test f1 score of XGBoost is 0.5491329479768786
The test f1 score of Decision Tree is 0.5673632725655847
The test f1 score of Logistic Regression is 0.5629168519341929
The test f1 score of KNN is 0.5300133392618942
The test f1 score of Gradient Boosting is 0.5460204535349044
The test f1 score of Random Forest is 0.588706091596265
The test f1 score of Ada Boosting is 0.5460204535349044
            [주성분 분석, undersampling]
```

[주성분분석, Undersampling 이용한 f1_score]

- Select-K-Best, Oversampling(smote)

```
The test f1_score of Neural Network is 0.0
The test f1_score of XGBoost is 0.12391269544799614
The test f1_score of Decision Tree is 0.12391269544799614
The test f1_score of Logistic Regression is 0.12482538973012239
The test f1_score of KNN is 0.12562814070351758
The test f1_score of Gradient Boosting is 0.0
The test f1_score of Random Forest is 0.0017182130584192437
The test f1_score of Ada Boosting is 0.0
            [Select-K-Best,Oversampling]
```

[Select-K-Best, Oversampling 이용한 f1_score]

- Select-K-Best, Undersampling

```
The test f1_score  of Neural Network is 0.0
The test f1_score  of XGBoost is 0.5022466300549177
The test f1_score  of Decision Tree is 0.6461318051575932
The test f1_score  of Logistic Regression is 0.018965517241379307
The test f1_score  of KNN is 0.5468025949953661
The test f1_score  of Gradient Boosting is 0.5493406093678945
The test f1_score  of Random Forest is 0.36321031048623315
The test f1_score  of Ada Boosting is 0.5493406093678945
            [Select-K-Best,undersampling]
```

[Select-K-Best, Undersampling 이용한 f1_score]

4.2 결과분석

- **최적의 hyperparameter**

여러 가지의 전처리 방식과, 다양한 모델들을 적용해본 결과 Select-K-Best 방식을 이용해 변수선택을 한 후 Undersampling을 한 후 Decsion Tree 모델에 적용시켰을 때 F1-Score가 가장 높은 값을 가졌다. 이후 Decision Tree의 성능을 높이기 위해 GridSearchCV 함수를 사용하여 가장 좋은 성능을 가지는 파라미터를 선정했다.

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import GridSearchCV

params={'max_depth':[2,3,4,6,8],'min_samples_split' : [2, 3],'min_samples_leaf':[4,6]}

gs=GridSearchCV(DecisionTreeClassifier(random_state=0,criterion = 'entropy'), params,scoring='f1',cv=5,verbose=1)
gs.fit(df_scaled2,train_y)
scores_df = pd.DataFrame(gs.cv_results_)
```

	params	mean_test_score	rank_test_score	split0_test_score	split1_test_score	split2_test_score	split3_test_score	split4_test_score
0	{'max_depth': 2, 'min_samples_leaf': 4, 'min_s...	0.676060	1	0.666157	0.692771	0.653291	0.689291	0.678788
1	{'max_depth': 2, 'min_samples_leaf': 4, 'min_s...	0.676060	1	0.666157	0.692771	0.653291	0.689291	0.678788
2	{'max_depth': 2, 'min_samples_leaf': 6, 'min_s...	0.676060	1	0.666157	0.692771	0.653291	0.689291	0.678788
3	{'max_depth': 2, 'min_samples_leaf': 6, 'min_s...	0.676060	1	0.666157	0.692771	0.653291	0.689291	0.678788
4	{'max_depth': 3, 'min_samples_leaf': 4, 'min_s...	0.618375	17	0.574735	0.617811	0.670295	0.638298	0.590734
5	{'max_depth': 3, 'min_samples_leaf': 4, 'min_s...	0.618375	17	0.574735	0.617811	0.670295	0.638298	0.590734
6	{'max_depth': 3, 'min_samples_leaf': 6, 'min_s...	0.618375	17	0.574735	0.617811	0.670295	0.638298	0.590734
7	{'max_depth': 3, 'min_samples_leaf': 6, 'min_s...	0.618375	17	0.574735	0.617811	0.670295	0.638298	0.590734
8	{'max_depth': 4, 'min_samples_leaf': 4, 'min_s...	0.655102	5	0.656766	0.667774	0.645045	0.649913	0.656013
9	{'max_depth': 4, 'min_samples_leaf': 4, 'min_s...	0.655102	5	0.656766	0.667774	0.645045	0.649913	0.656013
10	{'max_depth': 4, 'min_samples_leaf': 6, 'min_s...	0.654563	7	0.654067	0.667774	0.645045	0.649913	0.656013
11	{'max_depth': 4, 'min_samples_leaf': 6, 'min_s...	0.654563	7	0.654067	0.667774	0.645045	0.649913	0.656013
12	{'max_depth': 6, 'min_samples_leaf': 4, 'min_s...	0.646998	9	0.624573	0.669797	0.660457	0.651243	0.628920
13	{'max_depth': 6, 'min_samples_leaf': 4, 'min_s...	0.646998	9	0.624573	0.669797	0.660457	0.651243	0.628920
14	{'max_depth': 6, 'min_samples_leaf': 6, 'min_s...	0.644181	11	0.641221	0.664582	0.657627	0.627556	0.629921
15	{'max_depth': 6, 'min_samples_leaf': 6, 'min_s...	0.644181	11	0.641221	0.664582	0.657627	0.627556	0.629921
16	{'max_depth': 8, 'min_samples_leaf': 4, 'min_s...	0.627747	13	0.573557	0.641156	0.660535	0.620690	0.642796

```
GridSearchCV 최고 평균 정확도 수치: 0.6761
GridSearchCV 최적 하이퍼파라미터:  {'max_depth': 2, 'min_samples_leaf': 4, 'min_samples_split': 2}
```

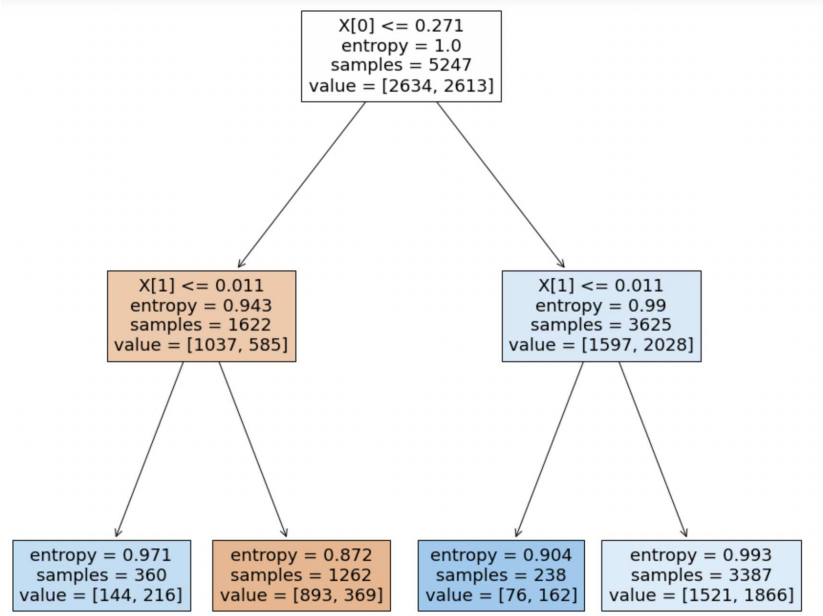
```
dt_precision:  0.5446853516657852
dt_recall:  0.9074889867841409
dt_f1-score:  0.6807666886979511

[[ 253  861]
 [ 105 1030]]
```

[최적의 hyperparameter를 적용시켰을 때의 precision, recall, f1-score, confusion matrix]

GridSearchCV 함수를 통해 최적의 parameters를 찾은 결과 트리의 최대깊이를 의미하는 max_depth=2, 리프노드가 되기 위해 필요한 최소한의 샘플 데이터 수의 개수를 의미하는 min_samples_leaf=4,노드를 분할하기 위한 최소한의 샘플 데이터 수를 의미하는 min_samples_split=2, 값으로 나왔다.
이 파라미터를 test set에 적용시켰을 때 precision=0.54 recall=0.91, f1_score=0.68이 나왔다.

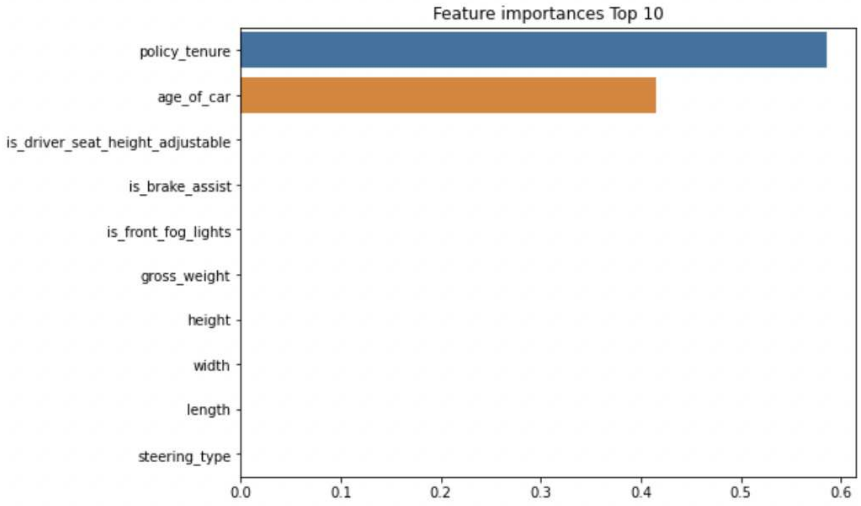
• Decision Tree 시각화



[max_depth=2, min_samples_leaf=4, min_samples_split=2 인 decision tree]

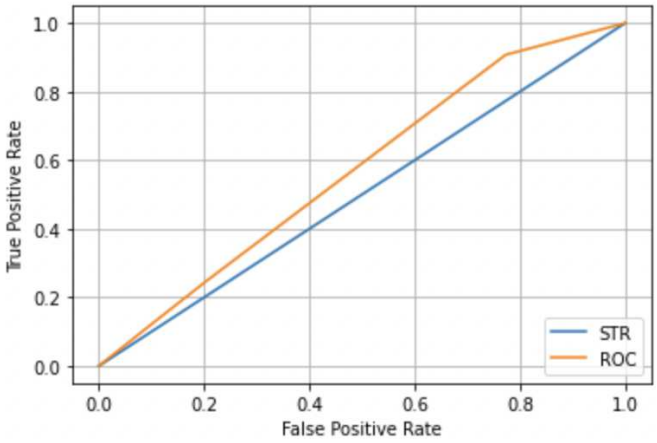
• **변수 중요도**

다음의 그림은 Decision Tree모델의 feature importance이다. Seaborn을 사용하여 예측 값들이 어떤 feature에 가장 많은 영향을 받는지를 시각화했다. 보험증서가 남은 기간과 차의 나이가 가장 중요한 변수임을 확인했다.



• Roc Curve

ROC Curve는 분류 분석 모형의 평가를 쉽게 비교할 수 있도록 시각화한 그래프이다. FPR이 변할 때 TPR이 어떻게 변하는지를 나타내는 곡선이다.



Roc Curve에서는 0.5와 가까운 값을 보여 아쉬운 점이 있었다.

5. conclusion

5.1 문제점 및 한계

• Undersampling, Oversampling

58592개의 데이터에서 자동차 보험을 청구한 숫자인 3748개로 줄여주었다. Undersampling의 가장 큰 단점은 제거하는 과정에서 유용한 정보가 버려지게 될 수 있다는 것이다. undersampling으로 인해 없어진 약 5만개의 데이터 중 자동차 보험 청구를 예측하는데 중요한 데이터가 있었을 것이다. 하지만 undersampling을 하며 많은 데이터가 버려졌기 때문에 모델의 정확도에 한계가 있었을 것이라고 예상한다.

데이터 불균형을 해결하기 위해 Undersampling과 smote oversampling 이외의 다른 resampling, adasyn, GAN 등 다양한 방식을 시도해보아야 할 필요성이 있다.

• 변수 선택

최종적으로 선택된 모델은 Decision Tree이다. Decision Tree 모델의 변수 중요도를 확인해본 결과 보험 증서가 남은 기간을 의미하는 policy_tenure와 차의 나이를 의미하는 age of car을 제외하고는 변수의 중요도가 0에 가까웠다. 이는 Decision tree 모델에 적용할 때 중요한 변수가 데이터 전처리 과정의 변수선택 과정에서 빠졌을 수 있다고 생각한다.

select-k-best 방식이 아닌 자동차 보험 가입 여부와 관련된 논문을 통해 연관관계가 높은 변수들을 직접 뽑아내는 방식도 시도해보아야 할 필요성이 있다.

5.2 향후 발전 방향

문제점 및 한계들을 해결하기 위해서는 아직 사용해보지 못한, 알지 못하는 알고리즘이 많아 다양한 예측모델을 사용해 예측의 모델의 성능을 높이는 작업이 더 필요하다고 생각한다. 모든 모델을 사용할 때 최적의 parameter 값을 찾거나, 세부 속성 값들을 변경해나갔다면 더욱 좋은 예측모델을 만들 수 있었을 것이다. 또한 이번 데이터에는 보험과 관련된 운전자의 정보보단 자동차의 기능적인 것에 대한 정보가 대부분이었다. 자동차 보험 청구는 운전자가 하는 것이기 때문에 운전자의 개인적인 정보에 대한 변수나 과거의 운전자의 보험 가입에 대한 정보를 더 추가한다면 자동차 보험 청구에 대한 예측에 현재보다 더 좋은 성능을 낼 수 있을 것이라고 생각한다.

6. 참고문헌

- 기승도, 황진태. (2011.03). 충성도를 고려한 자동차보험 마케팅 전략 연구. 보험 연구원 정책 보고서[권호 : 11-3].

- Hui Dong Wang.(2019).Research on the Features of Car Insurance Data Based on Machine Learning.procedia Computer Science 166 (2020) 582-587

- Endalew Alamir, Teklu Urgessa, Ashebir Hunegnaw, Tiruveedula Gopikrish. (2021). Motor Insurance Claim Status Prediction using Machine Learning Techniques. International Journal of Advanced Computer Science and Applications,Vol. 12, No. 3.