프로젝트 1 보고서

2022171063 김경민

1.모델을 이용한 사전 예측없이 모든 칩에 대하여 실제 테스트를 수행한다고 하자. 훈련 데이터 셋을 바탕으로 총 이득을 계산하면 얼마인가? 데이터 분석을 통하여 불량을 완벽히 판별하는 이상적인 모델을 만들었다고 가정했을 때의 총 이득은 얼마인가? 만일, 모델이 90%의 Recall과 90%의 Precision을 가질 때의 총 이득은 얼마인가?

Confusion Matrix에서, 각각의 확률에 대해 이득은 다음과 같이 계산된다.

TP= -P=-100

TN=S-P-Q=900

FP=-P=-100

FN=-P-Q=-1100

(1) No model - 훈련데이터셋을 바탕으로 총 이득은

모든 칩을 검사하였을 때 정상인 칩이 900원의 이득을, 불량인 칩이 1100원의 손해를 낸다.

➔ 8,861,900원

(2) Perfect model 의 총 이득

정상인 칩이 900원의 이득을, 불량인 칩이 100원의 손해를 낸다.

➔ 9,601,900원

(3) 90% recall과 precision 을 가진다면

90% recall 일 경우 90%의 불량 칩을 판별해내므로 TP=불량 개수의 90%

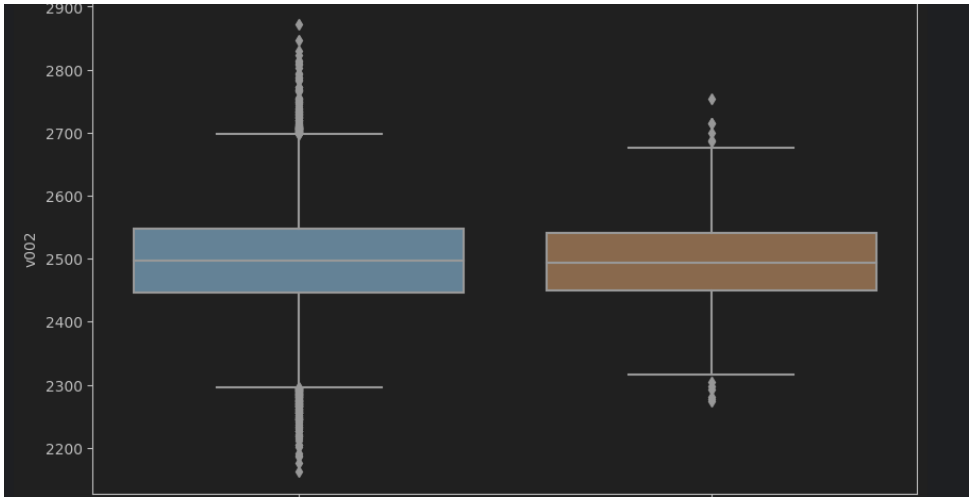
90% precision일 경우 불량 칩 중 90%가 실제 불량이므로 FP=TP*(1/9)이다.

➔ 9,453,900원

2. 필요한 EDA를 수행하고 훈련 데이터를 분석에 사용할 수 있는 데이터로 전처리하시오. 각 변

수의 분포 확인 및 시각화, 데이터의 변환, 이상치 탐색 및 결측치 처리, 통계 분석을 통한 변수의 선별 등을 포함한다. EDA와 전처리 과정을 설명하고, 특이 사항을 보고하시오. 최종적으로 전처리된 데이터의 표본의 수와 변수의 수를 보고하시오.

Data_training.describe()를 사용하여 각 변수의 대략적인 정보를 확인하였다. 변수 590개, 샘플 11491개로 이루어진 데이터이이고, 정규분포를 따르기보다는 이상값이 많았다.



boxplot example

EDA에서 standardscalar, fit_transform을 사용하여 정규화하였고, variable에서 zero variance 를 가진 column은 다음과 같이 분류했다. (nunique=1 사용). Zero variance column은 116개로 확인됐다.

- 만약 column이 dominantly zero라면, 결측치를 1로 처리
- 아니라면 결측치를 0으로 처리
- NA가 50%보다 적은 variable일 경우 NA는 평균으로 대치
- Column이 dominantly NA라면, NA를 0으로 대치한다. / drop column의 경우 performance가 그렇지 않은 경우보다 떨어졌기 때문에 변수 축소를 최소화했다.

Boxplot으로 이상치와 Q값들의 분포를 확인하였는데 결과는 다음과 같다. 전체적으로 값이 극단적으로 치우쳐 있음을 알 수 있다. 전반적으로 값이 쏠려 있는 경우 비선형 데이터를 처리하는 KNN 모델이 강할 것으로 예상하여 가장 먼저 model training 에 포함시켰다.

3. 전처리된 데이터를 바탕으로 불량에 대한 예측 모델을 만드시오. 다양한 모델을 시도하고 적절한 검증 과정을 거쳐 최종적으로 하나의 모델을 선택하시오. 어떠한 모델을 시도하였고, 어떠한 검증 방식을 사용하였는지 설명하시오. 특히, 모델 선택을 위한 성능 지표를 어떻게 결정하였는지 설명하시오. 선택된 모델에 대하여 훈련 데이터에서의 성능과 검증 데이터에서의 성능을 보고하

시오. (중요: 대부분의 이진 분류에서는 확률값 0.5를 기준으로 하여 분류하지만, 실제로는 꼭 그럴 필요는 없다. 다른 값을 사용하여 분류하여 원하는 성능을 달성할 수도 있다.)

평가 기준: true positive 및 true negative 가 다른 두 parameter 보다 이득에 큰 영향을 주기 때문에(각각 이득 900원, 손해 100원) recall/precision이 모두 중요하다. 따라서 그 둘을 아울러 설명할 수 있는 f1 스코어를 사용하여 모델을 평가하였다.

Validation set은 Training set의 0.2로 하였다.

비교에 사용한 모델은 KNN, Logistic Regression(L2 loss function), Random Forest, Gradient Boosting 그리고 이중 점수가 높은 셋을 조합해 ensemble model을 만들었다.

Gradient Boosting은 매트릭스로 depth 및 여러 파라메터를 조합해 보았으나 컴파일에 시간이 너무 오래 걸려 적당한 값으로 파라메터를 설정했다.

	KNN(k=5)	L2 Regression	Random Forest	Gradient Boosting	Ensemble
F1	0.9448	0.9085	0.8562	0.8791	0.9722
AUC	0.9883	0.9924	0.9911	0.9886	0.9996
Optimal Threshold	0.4	0.890	0.36	0.21	0.43

Optimal threshold method를 사용하여 이진분류에서 확률값 0.5를 사용하여 분류하는데 0.01부터 0.99까지를 한번씩 시험해 보면서 높은 F1 스코어가 나오는 값을 채택하였다. Ensemble model은 f1-0.9722의 점수를 가지고, 시험한 모델 중에서 가장 최적화되어 있다.

4. 위의 모델을 이용하여 평가 데이터를 분류하고 그 결과를 파일에 저장하여 제출하시오. (주의: 평가 데이터도 전처리가 필요한데, 이때는 평가데이터의 요약 정보를 이용할 수 없고 훈련 데이터의 정보를 이용해서만 전처리를 수행하여야 한다. 평가 데이터는 실제로 공장에서 하나씩 발생하기 때문에 특정 변수의 평균이나 분산을 구할 수 없다.)

평가 데이터는 훈련 데이터에서 fit 했던 것을 그대로 transform 하였고, 훈련데이터에서 했던 것처럼 정규화 및 동일한 규칙으로 NA 제거를 거친다. (preprocess_data(is_train=False)이용)

결과적으로

5. 위의 모델을 훈련 데이터에 적용하여 총 이득을 산정하고, 모델이 효용성을 갖는지 확인하시오. 모델을 사용하지 않는 경우에 비해 얼마나 이득이 증가하였는가? 이상적인 모델에 비하여 얼마나 이득을 올릴 수 있는가?

Confusion matrix를 사용하였을 때의 결과와 총 이득은 다음과 같다.

이는 No model situation에 비하여 의 성능 증가를 보인다.

➔ 8,861,900원

또한 Perfect model situation에 비하여 의 성능 증가를 보인다.

➔ 9,601,900원

6. 마케팅 부서에서는 반도체 가격 경쟁으로 인하여 내년도 판매가격(S)이 1,200원이 될 것으로 예상하였다. 이 때, 모델을 쓰지 않는 경우, 이상적이 모델을 쓰는 경우, 위에서 개발한 모델을 쓰는 경우 각각의 총 이득은 얼마인가? 모델을 수정하여 이득을 더 높일 수 있는 방안이 있는지 확인하시오. 전체 훈련 데이터를 이용하여 확인하시오.

No model case, Perfect model case, developed model case 에서 1번에서 사용했던 계산 방법을 그대로 적용하면 각각 261,100원, 1,001,100원, 998,700원의 이득을 얻을 수 있다.

S가 감소하였으므로 FN이 갖는 손실값은 perfect model case에 비해서 1000원이 증가한다. 따라서 FN이 최소화되어야 이득을 더 높일 수 있다.

(7) 어제 밤 경쟁 회사의 공장에서 대규모 화재가 발생하였다. 내년에 반도체 품귀 현상이 발생할 것으로 예상되었다. 마케팅 부서는 판매가격(S)을 5,000원으로 인상하였다. 이 때, 모델을 쓰지 않는 경우, 이상적이 모델을 쓰는 경우, 위에서 개발한 모델을 쓰는 경우 각각의 총 이득은 얼마인가? 모델을 수정하여 이득을 더 높일 수 있는 방안이 있는지 확인하시오. 전체 훈련 데이터를 이용하여 확인하시오.

No model case, Perfect model case, developed model case 에서 1번에서 사용했던 계산 방법을 그대로 적용하면 각각 41,114,900원, 41,854,900원, 41,844,900원의 이득이 생긴다.

S가 증가하였으므로 FP는 perfect model 과 비교했을 때 4000원의 손해를, FN의 경우 1000원의 손해를 본다.

6,7의 결과는 코드 실행결과 참조.