




KUBIG 머신러닝 분반

와인 품질 분류

경진대회

3팀 정윤주, 이수민, 정하운





CONTENTS

01 프로젝트 소개

02 EDA

03 데이터 전처리

04 모델링

05 결과 평가

01. 프로젝트 소개

와인 품질(Quality) 분류 경진대회

정형 | 알고리즘 | 초급

₩ 상금 : 교육

🕒 2023.08.01 ~ 2023.08.31 23:59

+ Google Calendar

👤 2,934명

📅 D-23

대회안내

데이터

코드 공유

토크

리더보드

팀

제출

☰ 개요

📄 규칙

1. 개요

안녕하세요 여러분! 🍷 와인 품질(Quality) 분류 경진대회에 오신 것을 환영합니다.

- index 구분자
- quality 품질
- fixed acidity 산도
- volatile acidity 휘발성산
- citric acid 시트르산
- residual sugar 잔당 : 발효 후 와인 속에 남아있는 당분
- chlorides 염화물
- free sulfur dioxide 독립 이산화황
- total sulfur dioxide 총 이산화황
- density 밀도
- pH 수소이온농도
- sulphates 황산염
- alcohol 도수
- type 종류

02. EDA

변수 확인

```
train.isna().sum()
```

index	0
quality	0
fixed acidity	0
volatile acidity	0
citric acid	0
residual sugar	0
chlorides	0
free sulfur dioxide	0
total sulfur dioxide	0
density	0
pH	0
sulphates	0
alcohol	0
type	0
dtype:	int64

```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 5497 entries, 0 to 5496  
Data columns (total 14 columns):  
#   Column                Non-Null Count  Dtype  
---  -  
0   index                 5497 non-null   int64  
1   quality               5497 non-null   int64  
2   fixed acidity         5497 non-null   float64  
3   volatile acidity      5497 non-null   float64  
4   citric acid           5497 non-null   float64  
5   residual sugar        5497 non-null   float64  
6   chlorides             5497 non-null   float64  
7   free sulfur dioxide    5497 non-null   float64  
8   total sulfur dioxide   5497 non-null   float64  
9   density               5497 non-null   float64  
10  pH                   5497 non-null   float64  
11  sulphates             5497 non-null   float64  
12  alcohol               5497 non-null   float64  
13  type                  5497 non-null   object  
dtypes: float64(11), int64(2), object(1)  
memory usage: 601.4+ KB
```

- 결측치 없음
- type 변수 제외 모두 numerical variable

02. EDA

변수 확인

	index	quality	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
count	5497.000000	5497.000000	5497.000000	5497.000000	5497.000000	5497.000000	5497.000000	5497.000000	5497.000000	5497.000000	5497.000000	5497.000000	5497.000000
mean	2748.000000	5.818992	7.210115	0.338163	0.318543	5.438075	0.055808	30.417682	115.566491	0.994673	3.219502	0.530524	10.504918
std	1586.991546	0.870311	1.287579	0.163224	0.145104	4.756676	0.034653	17.673881	56.288223	0.003014	0.160713	0.149396	1.194524
min	0.000000	3.000000	3.800000	0.080000	0.000000	0.600000	0.009000	1.000000	6.000000	0.987110	2.740000	0.220000	8.000000
25%	1374.000000	5.000000	6.400000	0.230000	0.250000	1.800000	0.038000	17.000000	78.000000	0.992300	3.110000	0.430000	9.500000
50%	2748.000000	6.000000	7.000000	0.290000	0.310000	3.000000	0.047000	29.000000	118.000000	0.994800	3.210000	0.510000	10.300000
75%	4122.000000	6.000000	7.700000	0.400000	0.390000	8.100000	0.064000	41.000000	155.000000	0.996930	3.320000	0.600000	11.300000
max	5496.000000	9.000000	15.900000	1.580000	1.660000	65.800000	0.610000	289.000000	440.000000	1.038980	4.010000	2.000000	14.900000

- 각 단위 값들의 차이가 많이 나는 것으로 보아 추후 scaling 필요할 것
- quality 변수는 continuous보다는 ordinal variable로 보임

02. EDA

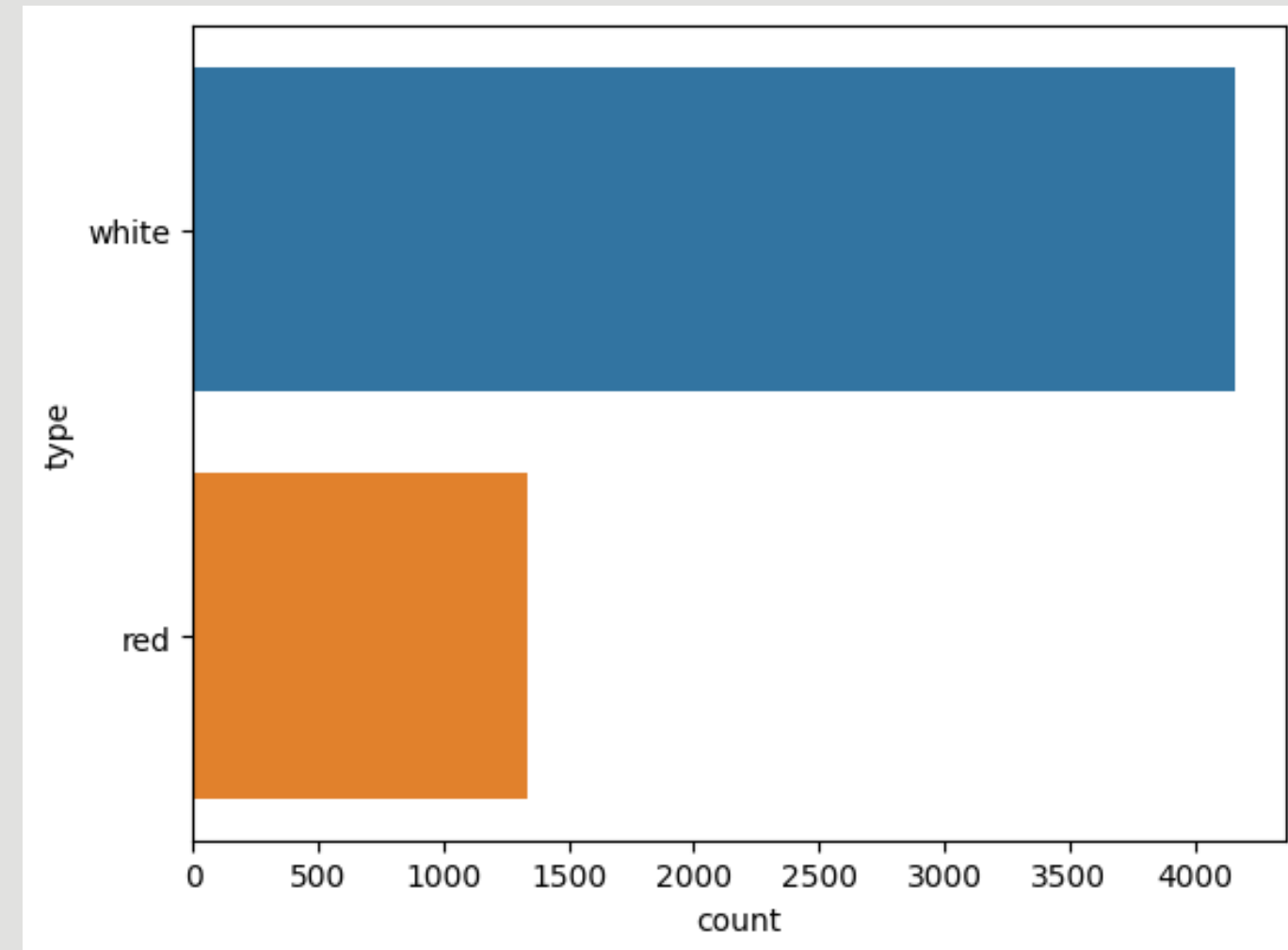
불균형 체크

```
train['type'] = train['type'].eq('red').mul(1).values
train['type'].value_counts()
```

```
0    4159
1     1338
Name: type, dtype: int64
```

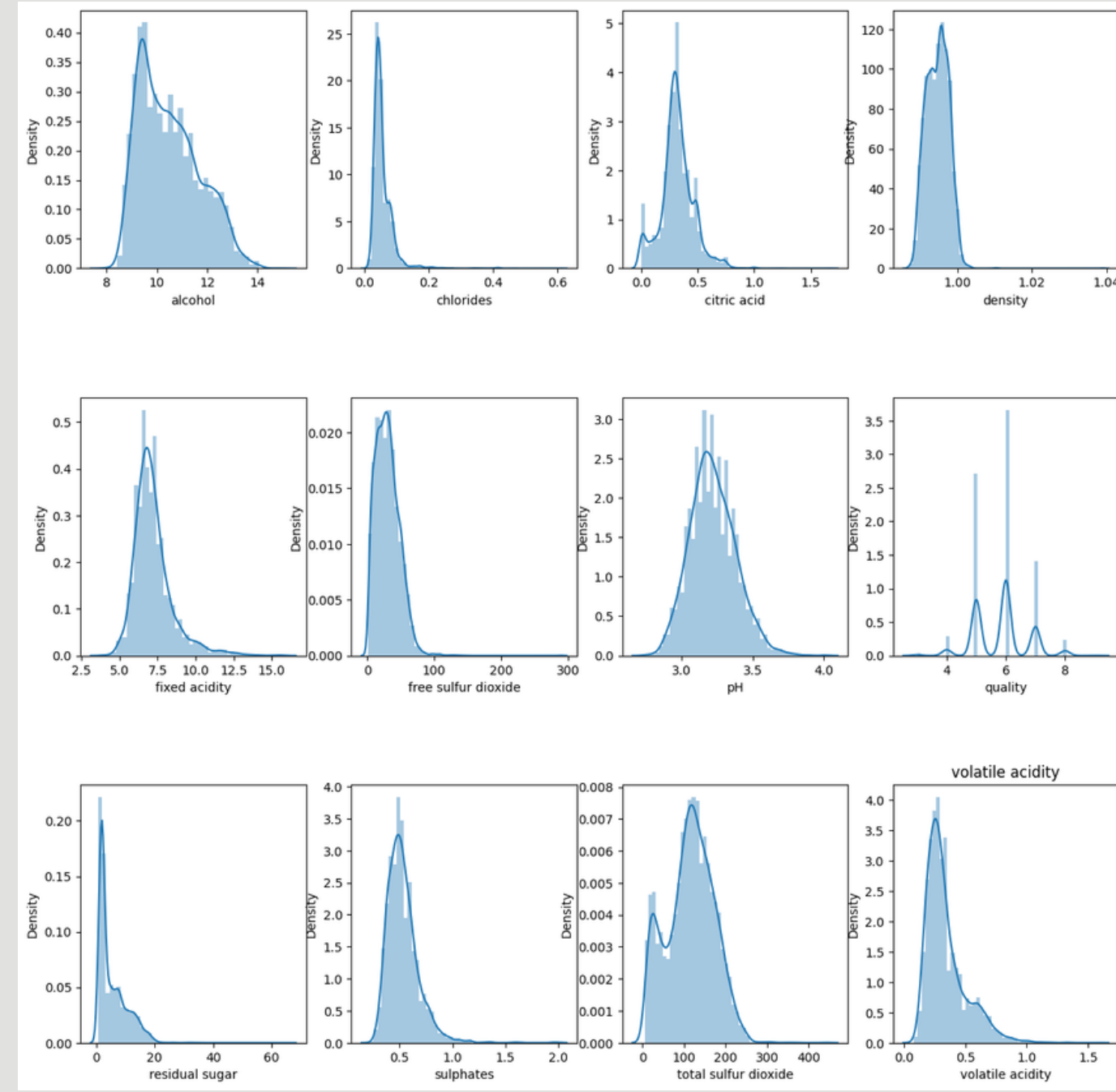
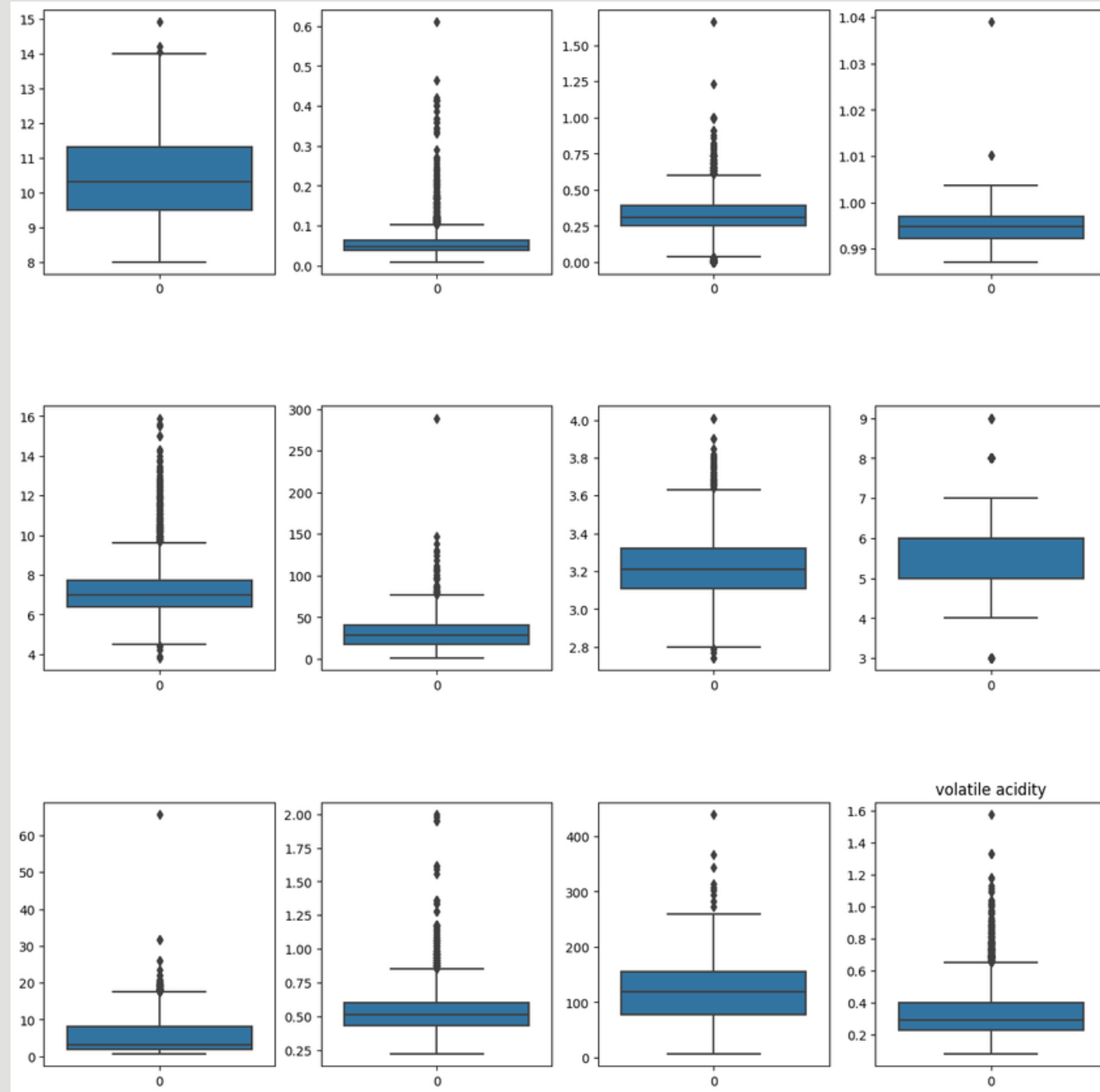
```
#불균형도 n(majority) / n(minority)
major = train['type'].value_counts().index[0]
minor = train['type'].value_counts().index[1]
num_major = train['type'].value_counts().iloc[0]
num_minor = train['type'].value_counts().iloc[1]
imbalance_rate = num_major / num_minor
print(imbalance_rate)
```

```
3.108370702541106
```



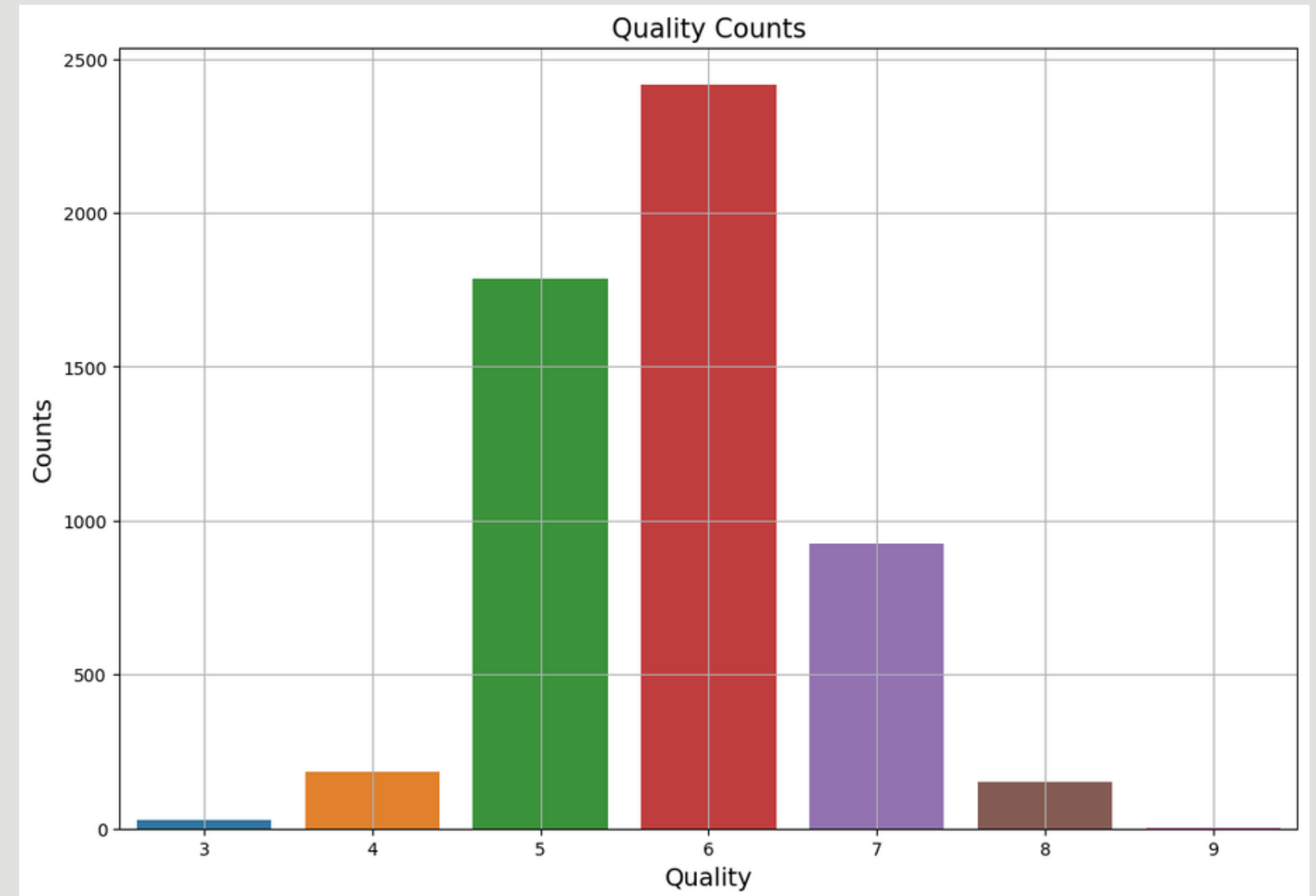
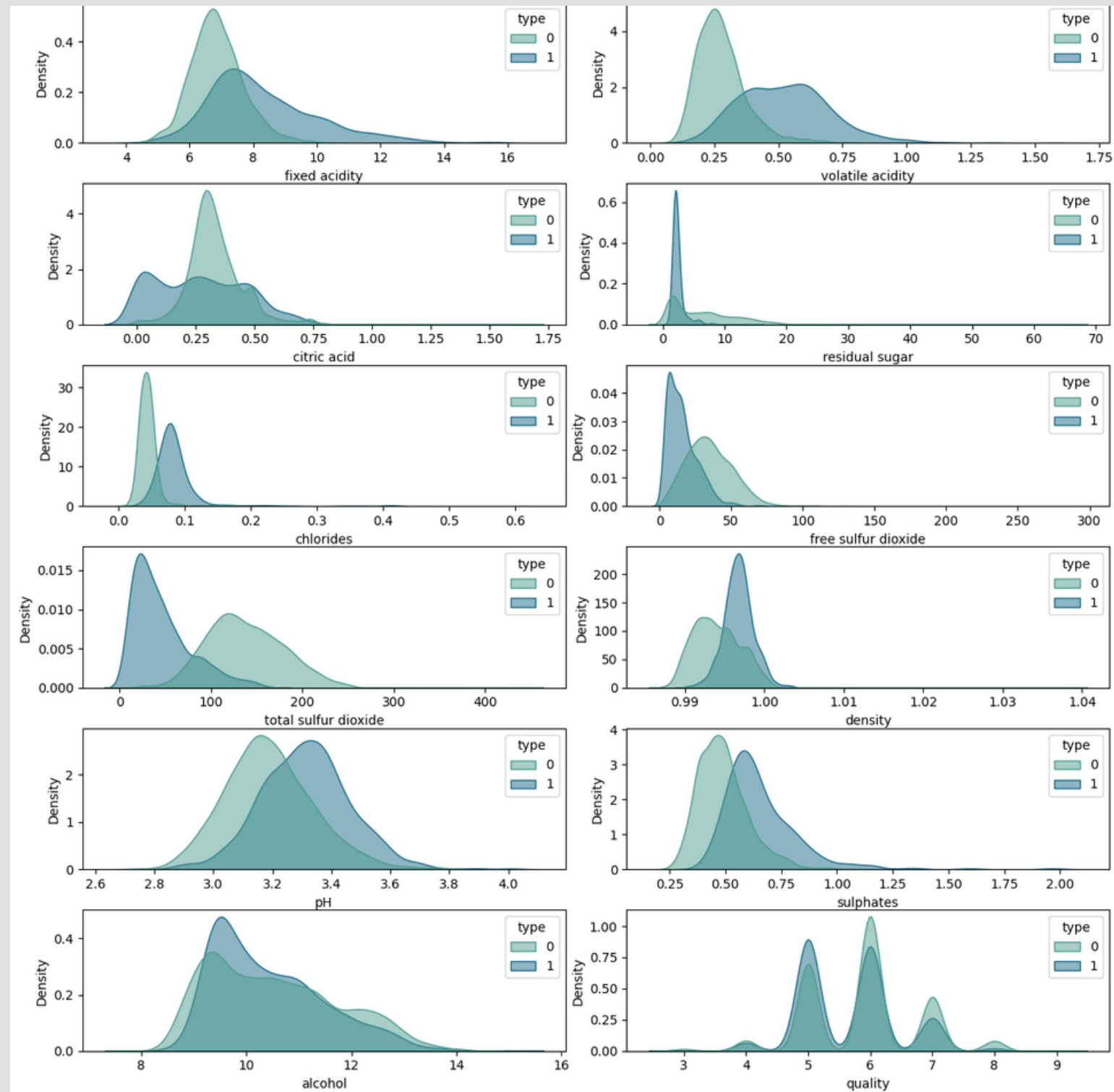
- type 변수 'red'를 1로, 'white'를 0으로 변환
- 와인 type 간 분포의 불균형이 심한 것으로 확인됨
- 'red' 데이터가 많지 않아 oversampling으로 불균형 처리 해주어야 할 것 같음

02. EDA



- 전체적으로 outlier가 많으며 분포가 치우쳐져 있음
- 각 값에 대한 white, red 분포가 크게 차이가 나므로 원하는 성능이 나오지 않는다면 red, white를 나눠 모델 제작도 고려해볼 것

02. EDA



- quality별 red, white 와인 데이터 수도 불균형
- 99 percentile을 기준으로 이상치를 제거

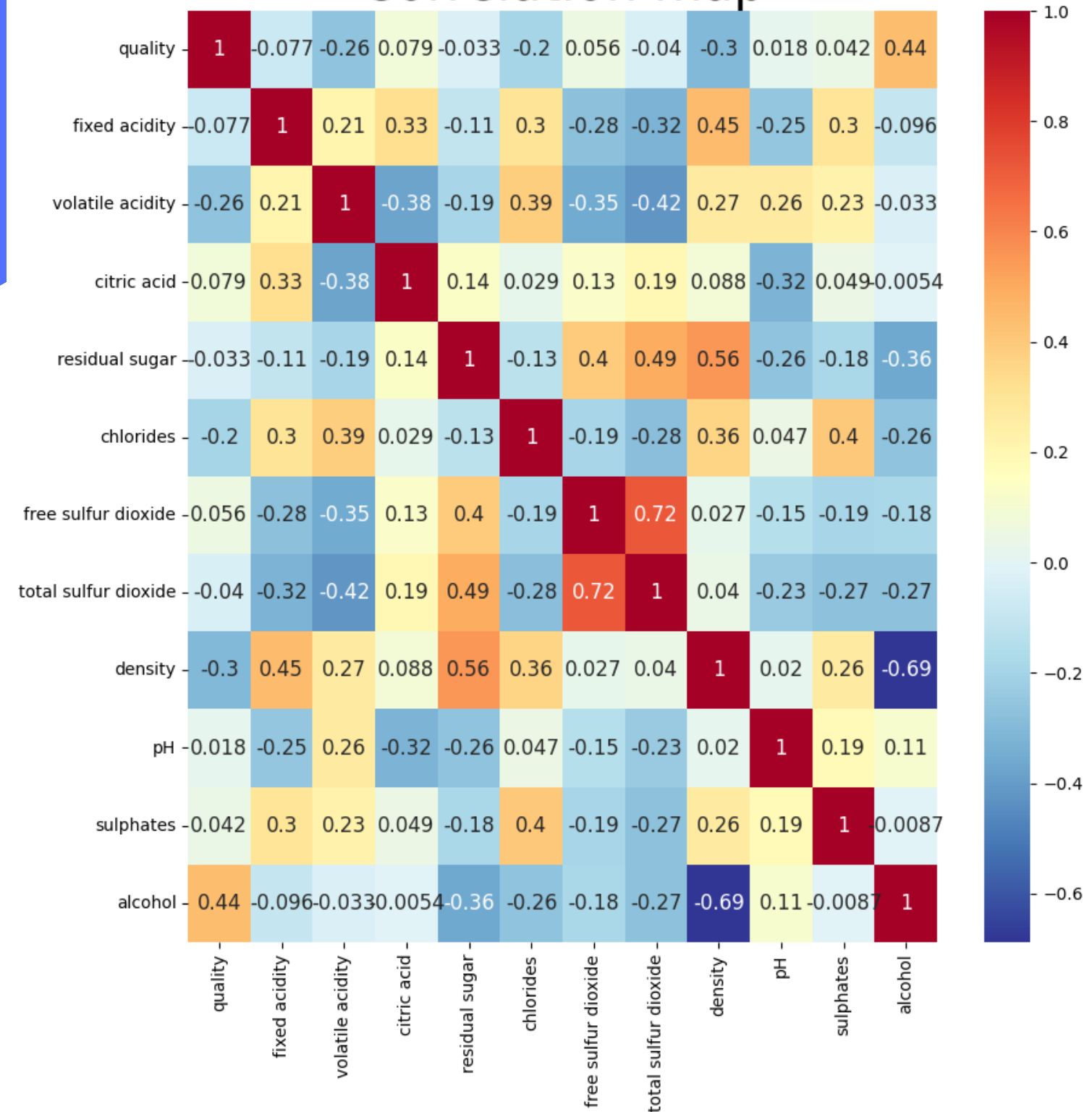
03. 데이터 전처리

다중공선성 확인

```
print(vif)
```

	Features	VIF
0	fixed acidity	59.135444
1	volatile acidity	9.030114
2	citric acid	9.259499
3	residual sugar	3.561881
4	chlorides	5.712596
5	free sulfur dioxide	8.341172
6	total sulfur dioxide	14.686250
7	density	933.739871
8	pH	582.667592
9	sulphates	18.276105
10	alcohol	108.030343

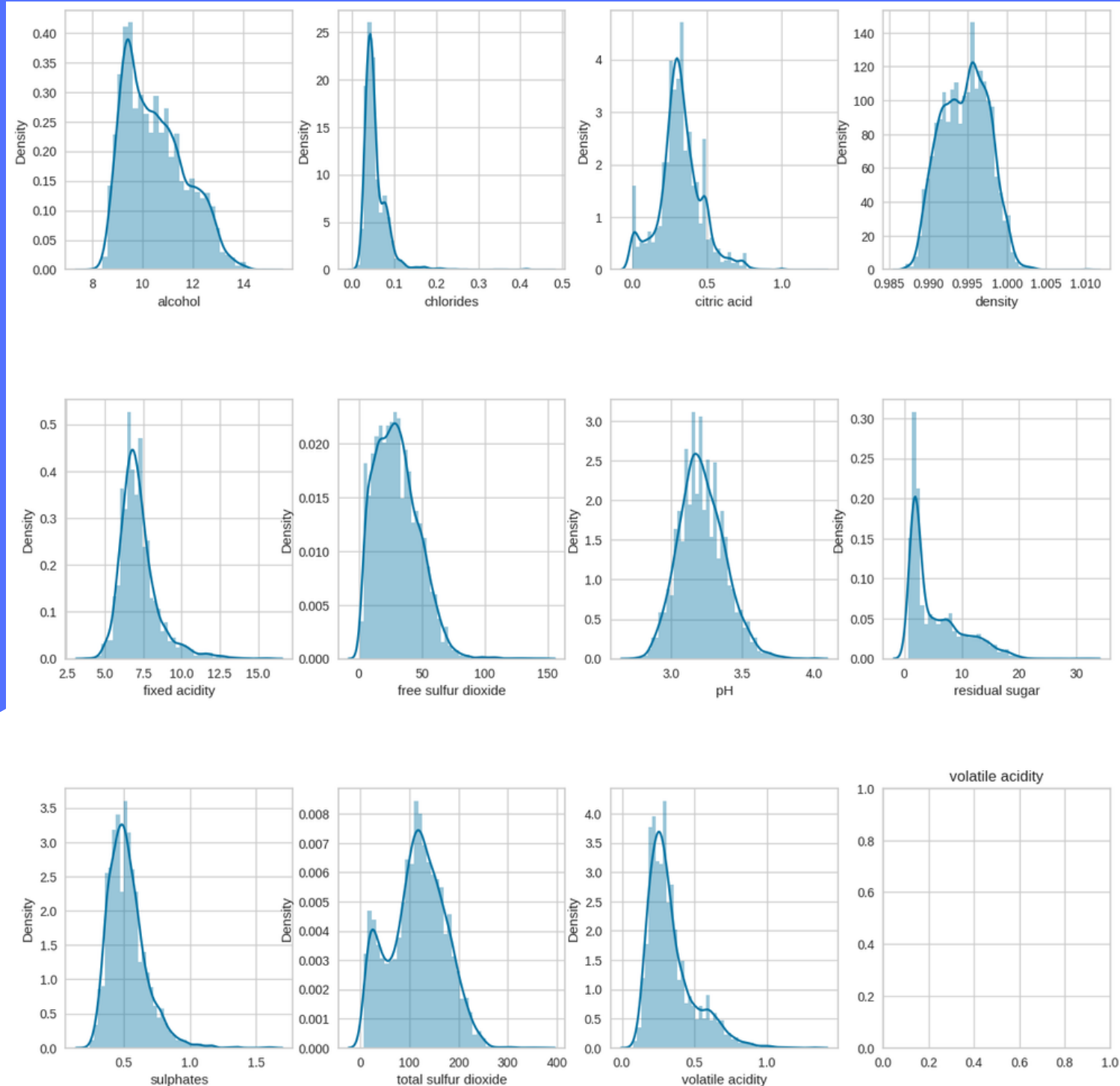
Correlation map



- VIF 10을 넘는 변수를 모두 제거한다면 변수들이 매우 적어지기 때문에 히트맵 기반으로 제거

03. 데이터 전처리

스케일링



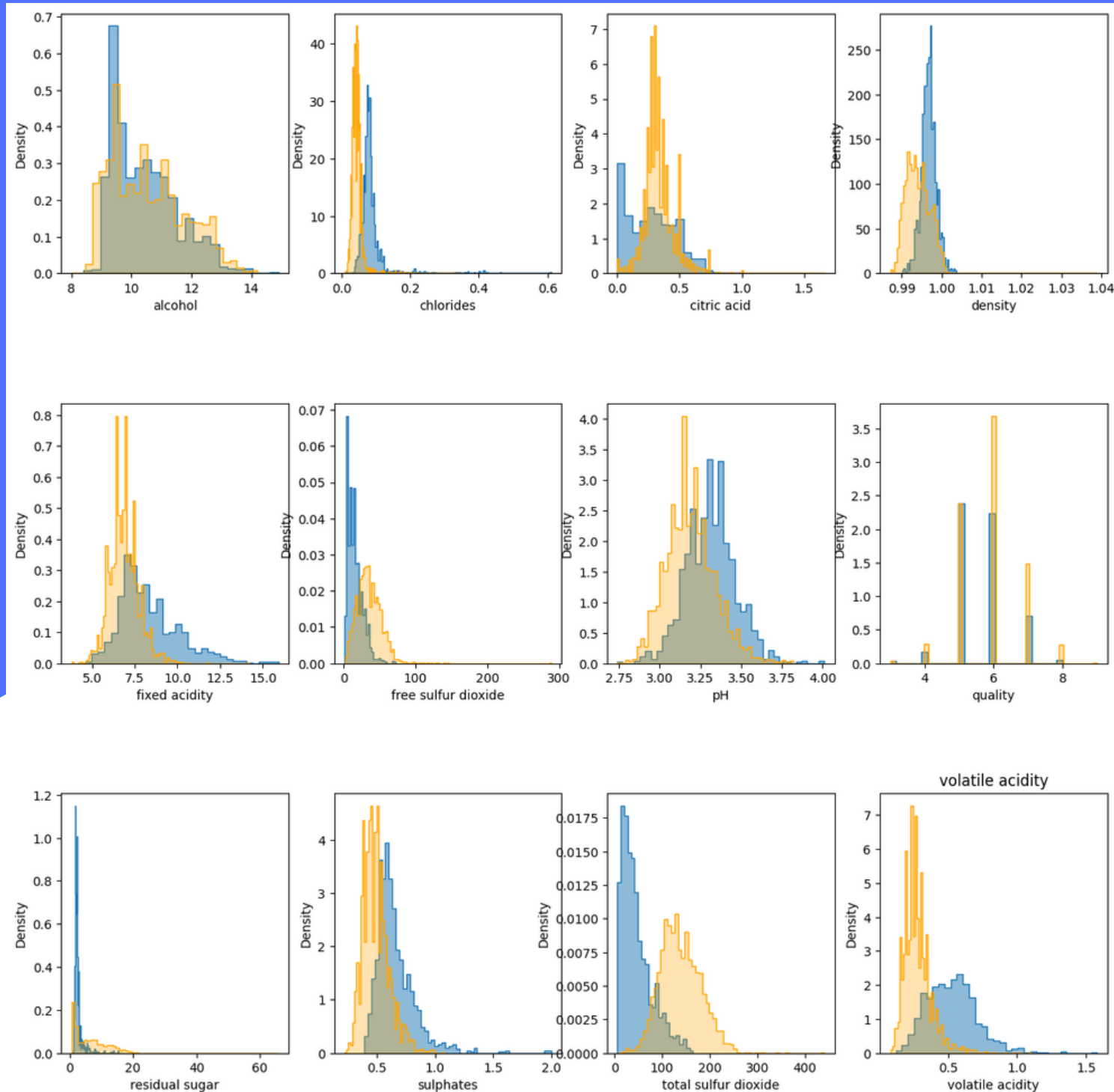
- shapiro test, ks-test 두가지 검정으로 정규성 가정 확인
- 모두 정규성 검정이 기각되어 standard scaler가 큰 의미 없을 것으로 판단
- 정규성을 따르지 않기 때문에 robust scaler 혹은 min-max scaler가 적절하다고 판단
- 성능 확인 후 robust scaler로 확정

** robust scaler

1. 모든 피처가 같은 크기를 갖는다는 점이 standard와 유사
2. 그러나 평균과 분산이 아닌 중위수(median)와 IQR(사분위수)를 사용함
3. Standard scaler에 비해 이상치의 영향을 적게 받음

03. 데이터 전처리

SMOTE



- quality 값이 불균형하여 oversampling 진행
 - oversampling의 경우 과적합이 일어날 수 있다는 단점
 - 그러나 quality 별 개수 차이가 심하기 때문에 undersampling 했을 때의 정보 손실이 크다고 판단
 - 과적합 가능성을 낮추기 위해 smote, borderline smote, adasyn 세가지 방법으로 각각 성능 확인
- smote data 최종 선택

04. 모델링

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
et	Extra Trees Classifier	0.9017	0.9890	0.9017	0.8993	0.8996	0.8853	0.8857	1.5230
rf	Random Forest Classifier	0.8907	0.9871	0.8907	0.8873	0.8879	0.8724	0.8728	2.2500
lightgbm	Light Gradient Boosting Machine	0.8803	0.9832	0.8803	0.8782	0.8789	0.8603	0.8605	2.6010
xgboost	Extreme Gradient Boosting	0.8790	0.9834	0.8790	0.8781	0.8783	0.8589	0.8590	10.1160
dt	Decision Tree Classifier	0.8265	0.8988	0.8265	0.8218	0.8235	0.7976	0.7978	0.2680
knn	K Neighbors Classifier	0.8227	0.9562	0.8227	0.8071	0.8070	0.7932	0.7960	0.2700
gbc	Gradient Boosting Classifier	0.8012	0.9610	0.8012	0.7934	0.7954	0.7680	0.7688	18.5240
lr	Logistic Regression	0.5310	0.8497	0.5310	0.5175	0.5180	0.4528	0.4548	1.2800
lda	Linear Discriminant Analysis	0.5125	0.8390	0.5125	0.4934	0.4958	0.4312	0.4333	0.2110
ridge	Ridge Classifier	0.4747	0.0000	0.4747	0.4577	0.4344	0.3871	0.3953	0.2780
svm	SVM - Linear Kernel	0.4656	0.0000	0.4656	0.4554	0.4482	0.3765	0.3813	0.5590
nb	Naive Bayes	0.4178	0.7786	0.4178	0.4021	0.3785	0.3208	0.3305	0.1890
ada	Ada Boost Classifier	0.3199	0.6690	0.3199	0.3186	0.2683	0.2065	0.2208	1.1880
qda	Quadratic Discriminant Analysis	0.1428	0.0000	0.1428	0.0204	0.0357	0.0000	0.0000	0.1900
dummy	Dummy Classifier	0.1420	0.5000	0.1420	0.0202	0.0353	0.0000	0.0000	0.1960

- pycaret을 통해 모델별 성능 확인
- Extra Trees Classifier가 0.9의 accuracy를 보이는 것을 확인

05. 결과 평가

test 데이터 예측

```
check_metric(y_test, prediction['prediction_label'], metric = 'Accuracy')
```

```
0.6467
```

```
check_metric(y_test, prediction['prediction_label'], metric='F1')
```

```
0.642
```

- 그러나 test 데이터로 모델의 최종 성능을 확인하자 정확도가 크게 낮아짐
- overfitting의 발생 원인 분석

⇒ SMOTE가 기존의 데이터 기반으로 가짜 데이터를 만들어 불균형을 맞추는데,
quality 9의 경우 train set에서 4개만 존재했으나 smote 이후 1600개 정도로 증가함.
이 과정에서 overfitting이 일어났다고 판단

05. 결과 평가

보완할 점

1. smote로 인해 overfitting이 일어남

*데이터 불균형이 심한데 oversampling을 진행하면 over fitting이 될 때 어떤 방법을 사용해야 할까요?

2. PCA를 하면 변수 삭제가 아니어도 다중공선성을 해결할 수 있을 것 같아, PCA같은 차원 축소 기법을 적용을 해보지 않은 점

3. 어떤 feature가 성능에 큰 영향을 미치는지 확인해서 성능 개선에 적극적으로 이용할 수 있었을 것 같은데 그 부분에 대한 공부가 부족해 적용해보지 못한 점

4. 변수간 VIF가 높은 것이 많아서 다른 변수를 찾아 추가할 수 있는 환경이었다면 좋았을 것 같음

5. 와인에 대한 선행지식이 얇아 타닌 등 어떤 지표가 와인 품질에 큰 영향을 미치는지 고려하기 어려운 점

감사합니다
