

인공지능

# 머신러닝을 이용한 물의 음용 가능성 분류

2021108249  
컴퓨터공학전공  
김성운





# CONTENTS

## 01. 개요 및 필요성

- 실험 개요
- 실험 필요성

## 02. 관련 연구

## 03. 본론

- 내용 요약
- 데이터 로드 및 시각화
- 데이터 전처리
- 학습 및 테스트
- 결과 분석

## 04. 결론 및 소감



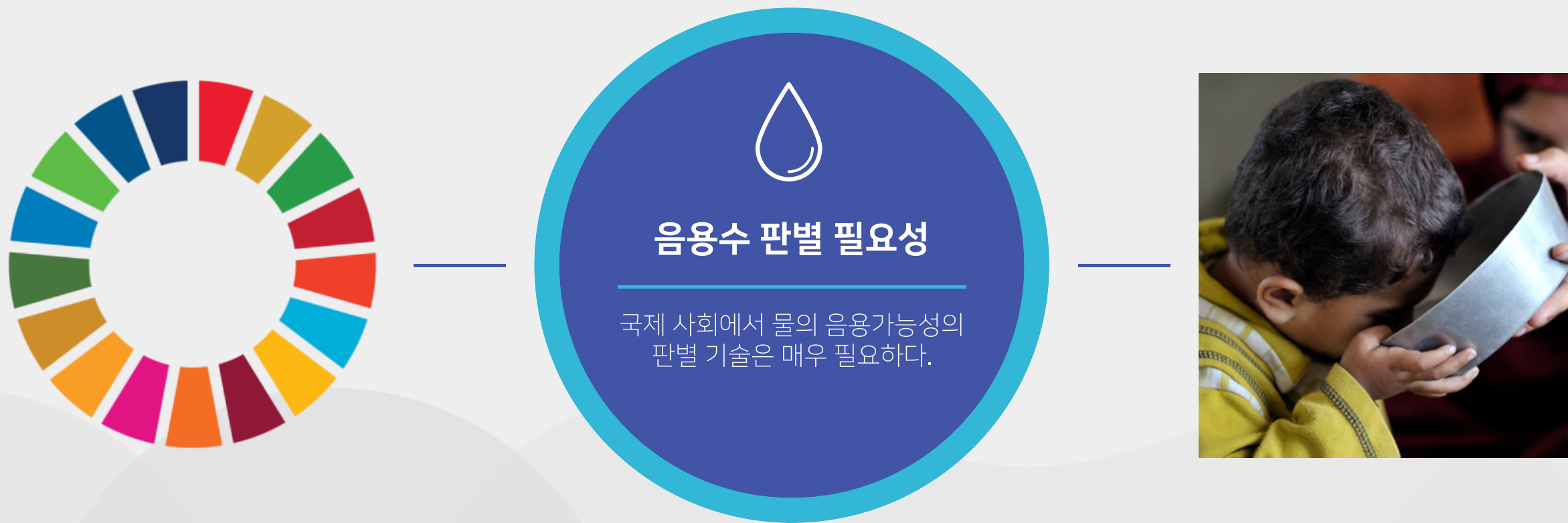
# 01. 개요 및 필요성

# 실험 개요

kaggle의 오픈소스 데이터 water\_potability.csv를 학습하여 물의 음용 가능성(Potability)를 구별할 수 있게 한다.

## 실험 필요성

- 2015년 UN회원국에 의해 지속가능발전목표(SDGs)를 채택하였다.
- 그 중 Goal 6은 Ensure access to water and sanitation for all이라는 제목으로 2030년까지 모든 사람이 안전하고 저렴한 식수에 대한 보편적 접근을 보장하기 위해 노력하고 있다.
- 하지만 여전히 2022년 기준으로 22억명의 사람들은 여전히 안전하게 관리되는 식수를 충분히 제공받지 못한다.
- 이러한 노력의 일환으로 현재 오염된 물을 정화하는 기술이 많이 개발되고 있다.
- 따라서 오염된 물에 대해 음용이 가능한지 판별이 필요하며 오염수 정화 기술의 결과를 판별해야 한다.





## 02. 관련 연구





LAKSIKA THARMALINGAM · 2MO AGO · 317 VIEWS



Copy &amp; Edit

8



## Models for water potability - Baselines





Python · [Water Quality and Potability](#)

Hindawi  
Computational Intelligence and Neuroscience  
Volume 2022, Article ID 9283293, 15 pages  
<https://doi.org/10.1155/2022/9283293>



### *Research Article*

## **A Machine Learning-Based Water Potability Prediction Model by Using Synthetic Minority Oversampling Technique and Explainable AI**

**Jinal Patel,<sup>1</sup> Charmi Amipara,<sup>1</sup> Tariq Ahamed Ahanger ,<sup>2</sup> Komal Ladhva,<sup>1</sup>  
Rajeev Kumar Gupta,<sup>1</sup> Hashem O. Alsaab ,<sup>3,4</sup> Yusuf S. Althobaiti ,<sup>4,5</sup>  
and Rajnish Ratna <sup>6</sup>**



## 03. 본론



## 사용 라이브러리

1. pycaret
2. 수업 중 사용한 myai 라이브러리 수정하여 사용



## 과정 요약



## 데이터 로드

```

1  from myai import *
2
3  ai = MyDataFrame()
4  # 1. csv 파일 로드
5  ai.load_csv('water_potability.csv')
6
7  # 2. csv 파일 정보 표시
8  ai.show_file_info()
9  ai.show_cols()

```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3276 entries, 0 to 3275
Data columns (total 10 columns):
#   Column                Non-Null Count  Dtype
---  -
0   ph                    2785 non-null   float64
1   Hardness              3276 non-null   float64
2   Solids               3276 non-null   float64
3   Chloramines          3276 non-null   float64
4   Sulfate              2495 non-null   float64
5   Conductivity         3276 non-null   float64
6   Organic_carbon       3276 non-null   float64
7   Trihalomethanes      3114 non-null   float64
8   Turbidity            3276 non-null   float64
9   Potability           3276 non-null   int64
dtypes: float64(9), int64(1)
memory usage: 256.1 KB
Index(['ph', 'Hardness', 'Solids', 'Chloramines', 'Sulfate', 'Conductivity',
      'Organic_carbon', 'Trihalomethanes', 'Turbidity', 'Potability'],
      dtype='object')

```

## 결측치 확인 및 제거

```
# 결측치 확인  
ai.check_null()
```

```
# 결측치 확인  
1개의 사용 위치  
def check_null(self):  
    print(self.data_frame.isnull().sum())
```

```
ph      491  
Hardness      0  
Solids      0  
Chloramines      0  
Sulfate      781  
Conductivity      0  
Organic_carbon      0  
Trihalomethanes      162  
Turbidity      0  
Potability      0  
dtype: int64
```

## 결측치 확인 및 제거

```
# 결측치 삭제  
ai.remove_null()
```

```
# 결측치 삭제  
1개의 사용 위치  
def remove_null(self):  
    self.data_frame.dropna(inplace=True)  
    self.data_frame.reset_index(inplace=True, drop=True)  
    print(self.data_frame.isnull().sum())
```

```
ph      0  
Hardness  0  
Solids  0  
Chloramines  0  
Sulfate  0  
Conductivity  0  
Organic_carbon  0  
Trihalomethanes  0  
Turbidity  0  
Potability  0  
dtype: int64
```

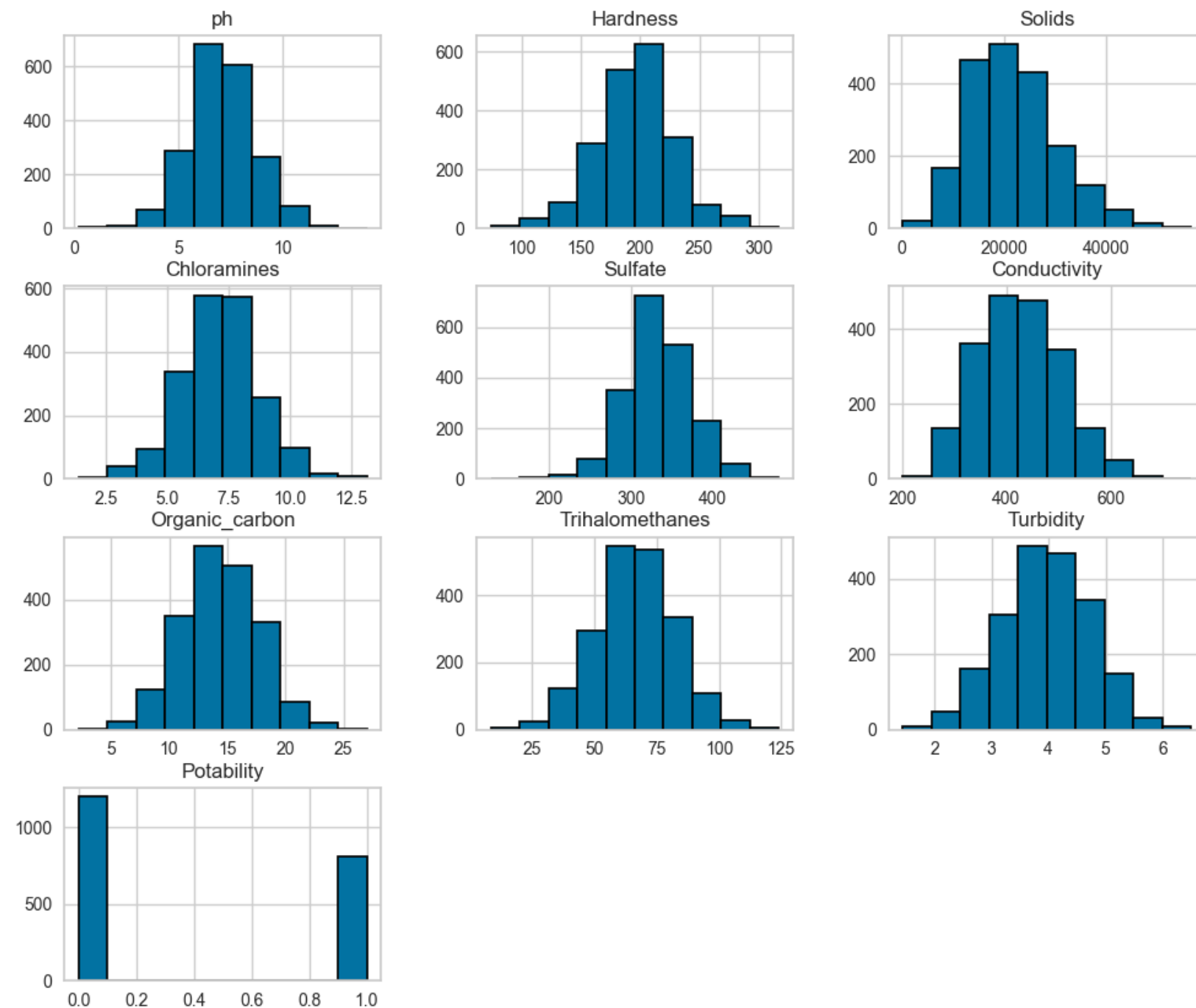
## 시각화

```
# 시각화  
ai.show_hist()  
ai.show_heatmap()
```

```
def show_hist(self):  
    self.data_frame.hist(edgecolor='black', linewidth=1.2)  
    fig = plt.gcf()  
    fig.set_size_inches(12, 10)  
    plt.show()
```

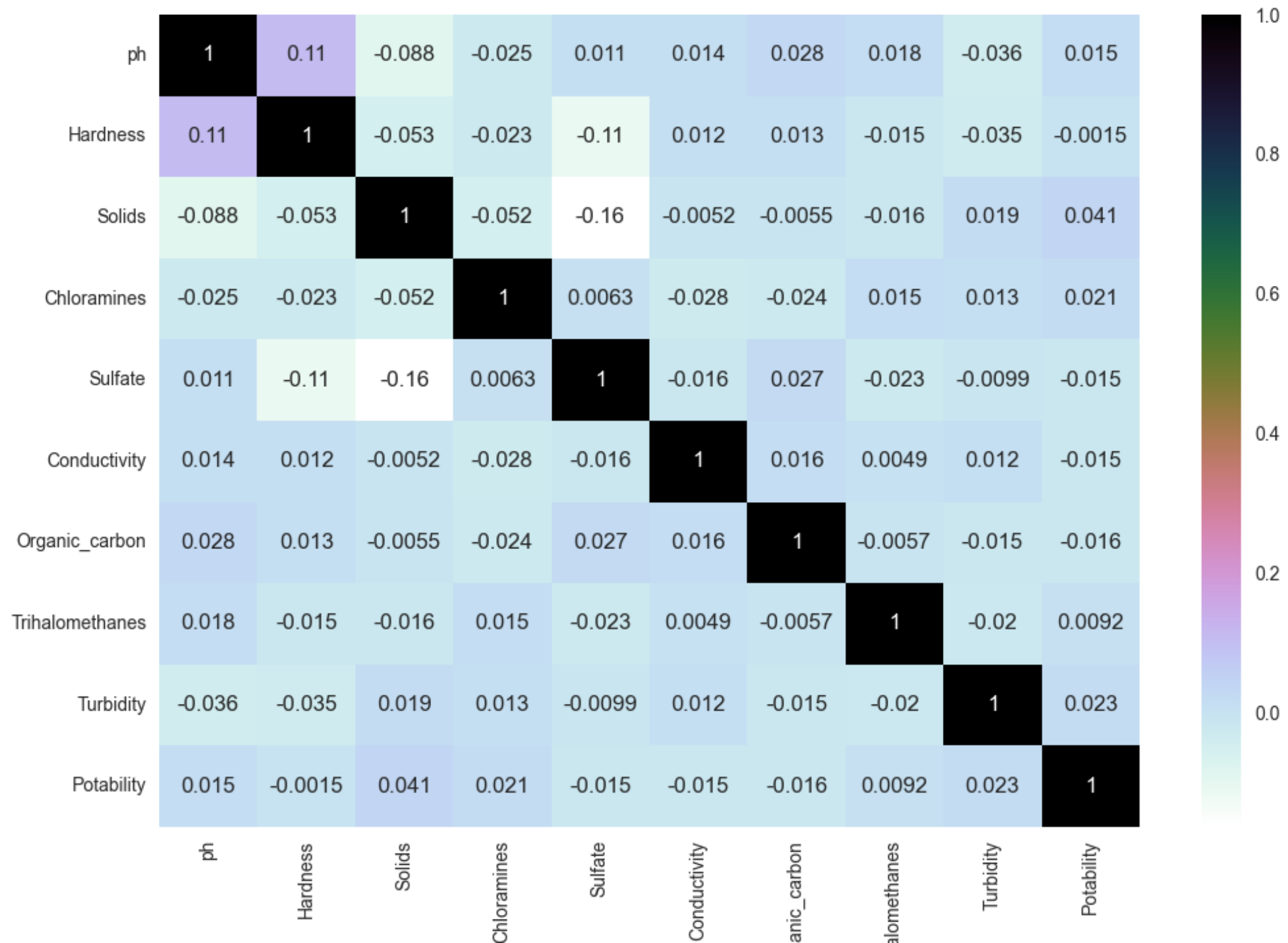
```
def show_heatmap(self):  
    plt.figure(figsize=(12, 8))  
    sns.heatmap(self.data_frame.corr(), annot=True, cmap='cubehelix_r')  
    plt.show()
```

# 시각화



모든 column의 값에 대한 분포를 나타내는 histogram

## 시각화



column 간의 연관성을 나타내는 heatmap

# Pycaret setting

```
ai.setting_up(target_col: 'Potability', ratio: 0.3)
```

```
def setting_up(self, target_col, ratio):
    train, test = train_test_split(*arrays: self.data_frame, test_size=ratio)
    setup(data=train, target=target_col, test_data=test, fix_imbalance=True, normalize=True,
          normalize_method="zscore", remove_outliers=True)
```

	Description	Value
0	Session id	7525
1	Target	Potability
2	Target type	Binary
3	Original data shape	(2011, 10)
4	Transformed data shape	(2230, 10)
5	Transformed train set shape	(1626, 10)
6	Transformed test set shape	(604, 10)
7	Numeric features	9
8	Preprocess	True
9	Imputation type	simple
10	Numeric imputation	mean
11	Categorical imputation	mode
12	Remove outliers	True
13	Outliers threshold	0.05
14	Fix imbalance	True
15	Fix imbalance method	SMOTE
16	Normalize	True
17	Normalize method	zscore
18	Fold Generator	StratifiedKFold
19	Fold Number	10
20	CPU Jobs	-1
21	Use GPU	False
22	Log Experiment	False
23	Experiment Name	clf-default-name
24	USI	6d68



## train and test

```
best_models = compare_models(n_select=5)
```

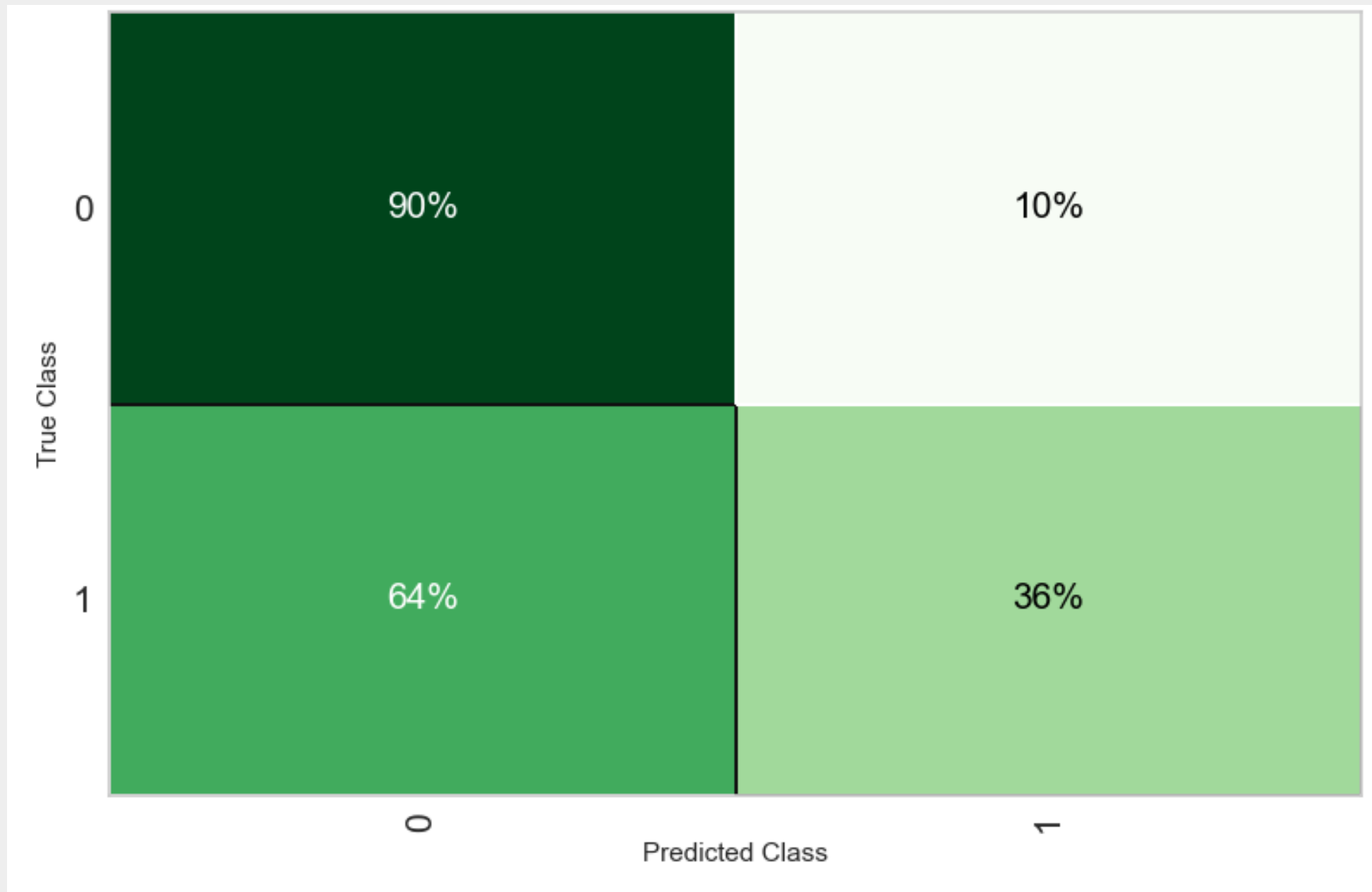
	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
catboost	CatBoost Classifier	0.6489	0.6936	0.5837	0.5540	0.5679	0.2728	0.2735	0.705
et	Extra Trees Classifier	0.6447	0.6777	0.5013	0.5540	0.5252	0.2433	0.2444	0.064
qda	Quadratic Discriminant Analysis	0.6403	0.6926	0.6248	0.5401	0.5789	0.2682	0.2706	0.034
rf	Random Forest Classifier	0.6397	0.6703	0.5264	0.5468	0.5354	0.2418	0.2423	0.096
lightgbm	Light Gradient Boosting Machine	0.6219	0.6469	0.5547	0.5216	0.5371	0.2182	0.2189	0.101
xgboost	Extreme Gradient Boosting	0.6177	0.6461	0.5300	0.5168	0.5215	0.2040	0.2050	0.053
dummy	Dummy Classifier	0.6041	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000	0.032
knn	K Neighbors Classifier	0.5949	0.6279	0.5962	0.4932	0.5392	0.1844	0.1867	0.233
gbc	Gradient Boosting Classifier	0.5899	0.6295	0.5712	0.4838	0.5229	0.1678	0.1703	0.090
dt	Decision Tree Classifier	0.5735	0.5641	0.5187	0.4655	0.5229	0.1678	0.1703	0.090
nb	Naive Bayes	0.5466	0.5699	0.5514	0.4408	0.4903	0.1257	0.1263	0.032
ada	Ada Boost Classifier	0.5310	0.5374	0.5101	0.4235	0.4615	0.0529	0.0537	0.052
lda	Linear Discriminant Analysis	0.5054	0.4845	0.4813	0.3985	0.4354	0.0029	0.0026	0.027
ridge	Ridge Classifier	0.4954	0.0000	0.4795	0.3897	0.4295	-0.0137	-0.0143	0.027
lr	Logistic Regression	0.4947	0.4945	0.4686	0.3880	0.4237	-0.0182	-0.0190	0.343
svm	SVM - Linear Kernel	0.4683	0.0000	0.4683	0.3628	0.4061	-0.0614	-0.0630	0.028

# blending

```
blender_specific = blend_models(estimator_list=best_models, choose_better=True)
```

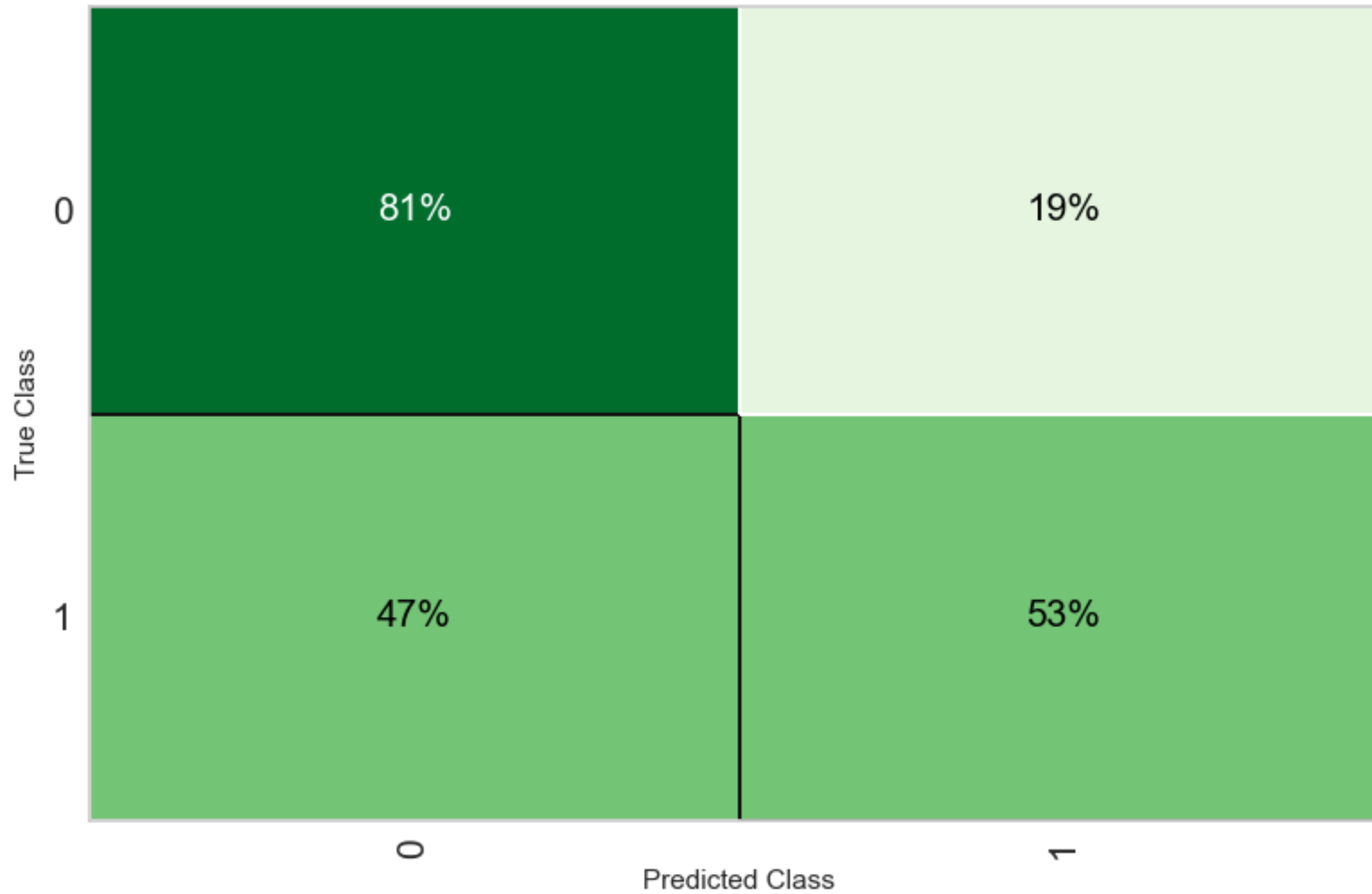
Original model was better than the blended model, hence it will be returned.

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
Fold							
0	0.6028	0.5952	0.4107	0.5000	0.4510	0.1445	0.1462
1	0.6099	0.6508	0.5179	0.5088	0.5133	0.1879	0.1879
2	0.7234	0.7645	0.6071	0.6667	0.6355	0.4134	0.4146
3	0.6738	0.6718	0.6071	0.5862	0.5965	0.3228	0.3230
4	0.6383	0.6662	0.5536	0.5439	0.5487	0.2469	0.2470
5	0.6454	0.7130	0.5893	0.5500	0.5690	0.2684	0.2688
6	0.6383	0.6649	0.4821	0.5510	0.5143	0.2282	0.2295
7	0.6857	0.7373	0.6545	0.5902	0.6207	0.3536	0.3550
8	0.5929	0.6759	0.5455	0.4839	0.5128	0.1653	0.1661
9	0.6500	0.7149	0.5818	0.5517	0.5664	0.2733	0.2736
Mean	0.6460	0.6855	0.5550	0.5532	0.5528	0.2604	0.2612
Std	0.0380	0.0458	0.0670	0.0500	0.0537	0.0807	0.0807



정규화 등의 전처리를 하지 않았을 경우  
한쪽에 치중된 결과가 나옴

ExtraTreesClassifier Confusion Matrix



정규화 등의 전처리를 한 후



## 04. 결론 및 소감



- 결론

- CatBoost Classifier 모델이 가장 정확도가 높았음(0.6489)
- 정규화 등의 전처리를 하기 전과 전처리를 한 후 결과가 상당히 다름

- 소감

- 전처리의 매우 중요하다는 것을 알게 되었습니다.
- 인공지능에 대한 관심이 많아졌습니다.
- 더 좋은 데이터셋을 구하지 못해 아쉽습니다.
- 부족한 점이 많아 아쉬웠습니다.

감사합니다