Red Wine Quality Analysis

2조

(박상규. 안용희. 안광필. 김정모. 오준원)

2020.05.28

목차

- 데이터 정제
- ANN, SVM
- KNN
- LogisticRegression
- Decision Tree, RandomForest
- DNN, NaiveBayes, Ensemble
- 결론

사용한 데이터셋

kaggle **winequality-red.csv**

	A	B	C	D	E	F	G	H	I	J	K	L
1	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxid	total sulfur dioxic	density	pH	sulphates	alcohol	quality
2	7.4	0.7	0	1.9	0.076	11	34	0.9978	3.51	0.56	9.4	5
3	7.8	0.88	0	2.6	0.098	25	67	0.9968	3.2	0.68	9.8	5
4	7.8	0.76	0.04	2.3	0.092	15	54	0.997	3.26	0.65	9.8	5
5	11.2	0.28	0.56	1.9	0.075	17	60	0.998	3.16	0.58	9.8	6
6	7.4	0.7	0	1.9	0.076	11	34	0.9978	3.51	0.56	9.4	5
7	7.4	0.66	0	1.8	0.075	13	40	0.9978	3.51	0.56	9.4	5
8	7.9	0.6	0.06	1.6	0.069	15	59	0.9964	3.3	0.46	9.4	5
9	7.3	0.65	0	1.2	0.065	15	21	0.9946	3.39	0.47	10	7
10	7.8	0.58	0.02	2	0.073	9	18	0.9968	3.36	0.57	9.5	7
11	7.5	0.5	0.36	6.1	0.071	17	102	0.9978	3.35	0.8	10.5	5
12	6.7	0.58	0.08	1.8	0.097	15	65	0.9959	3.28	0.54	9.2	5
13	7.5	0.5	0.36	6.1	0.071	17	102	0.9978	3.35	0.8	10.5	5
14	5.6	0.615	0	1.6	0.089	16	59	0.9943	3.58	0.52	9.9	5
15	7.8	0.61	0.29	1.6	0.114	9	29	0.9974	3.26	1.56	9.1	5

Feature 소개

fixed acidity	volatile acidity	citric acid	residual sugar	chlo-rides	free sulfur dioxide	total sulfur dioxide	density	pH	sulpha-tes	alcohol	quality
주석산 농도	아세트산 농도	구연산 농도	잔류 당분 농도	염화 나트륨 농도	유리 아황산 농도	총 아황산 농도	밀도 (점성)	pH	황산 칼륨 농도	알코올 도수	와인 등급

	A	B	C	D	E	F	G	H	I	J	K	L
1	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxid	total sulfur dioxic	density	pH	sulphates	alcohol	quality
2	7.4	0.7	0	1.9	0.076	11	34	0.9978	3.51	0.56	9.4	5
3	7.8	0.88	0	2.6	0.098	25	67	0.9968	3.2	0.68	9.8	5
4	7.8	0.76	0.04	2.3	0.092	15	54	0.997	3.26	0.65	9.8	5
5	11.2	0.28	0.56	1.9	0.075	17	60	0.998	3.16	0.58	9.8	6
6	7.4	0.7	0	1.9	0.076	11	34	0.9978	3.51	0.56	9.4	5
7	7.4	0.66	0	1.8	0.075	13	40	0.9978	3.51	0.56	9.4	5
8	7.9	0.6	0.06	1.6	0.069	15	59	0.9964	3.3	0.46	9.4	5
9	7.3	0.65	0	1.2	0.065	15	21	0.9946	3.39	0.47	10	7
10	7.8	0.58	0.02	2	0.073	9	18	0.9968	3.36	0.57	9.5	7
11	7.5	0.5	0.36	6.1	0.071	17	102	0.9978	3.35	0.8	10.5	5
12	6.7	0.58	0.08	1.8	0.097	15	65	0.9959	3.28	0.54	9.2	5
13	7.5	0.5	0.36	6.1	0.071	17	102	0.9978	3.35	0.8	10.5	5
14	5.6	0.615	0	1.6	0.089	16	59	0.9943	3.58	0.52	9.9	5
15	7.8	0.61	0.29	1.6	0.114	9	29	0.9974	3.26	1.56	9.1	5

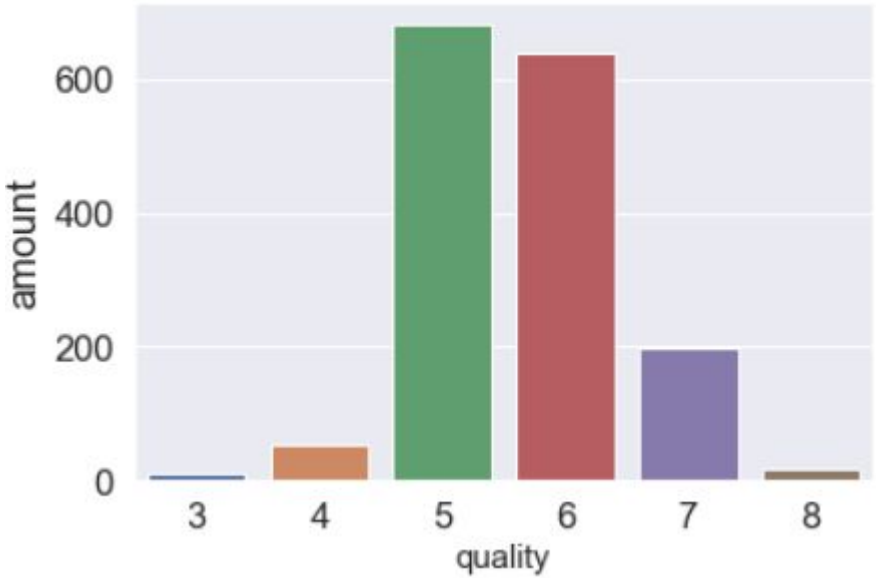
데이터 전처리

결측값, 데이터형식 확인

```
red.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1599 entries, 0 to 1598
Data columns (total 12 columns):
fixed acidity           1599 non-null float64
volatile acidity        1599 non-null float64
citric acid             1599 non-null float64
residual sugar          1599 non-null float64
chlorides               1599 non-null float64
free sulfur dioxide     1599 non-null float64
total sulfur dioxide    1599 non-null float64
density                 1599 non-null float64
pH                      1599 non-null float64
sulphates               1599 non-null float64
alcohol                 1599 non-null float64
quality                 1599 non-null int64
dtypes: float64(11), int64(1)
memory usage: 150.0 KB
```

타겟 확인



→ 결측값 없음

→ 타겟 수정 (5이하 = 0, 6이상 = 1)

데이터 전처리 피처 상관도 분석

	VIF Factor	feature
1	3.031160	alcohol
2	1.481932	chlorides
3	3.128022	citric acid
4	6.343760	density
5	7.767512	fixed acidity
6	1.963019	free sulfur dioxide
7	3.329732	pH
8	1.702588	residual sugar
9	1.429434	sulphates
10	2.186813	total sulfur dioxide
11	1.789390	volatile acidity

다중공선성

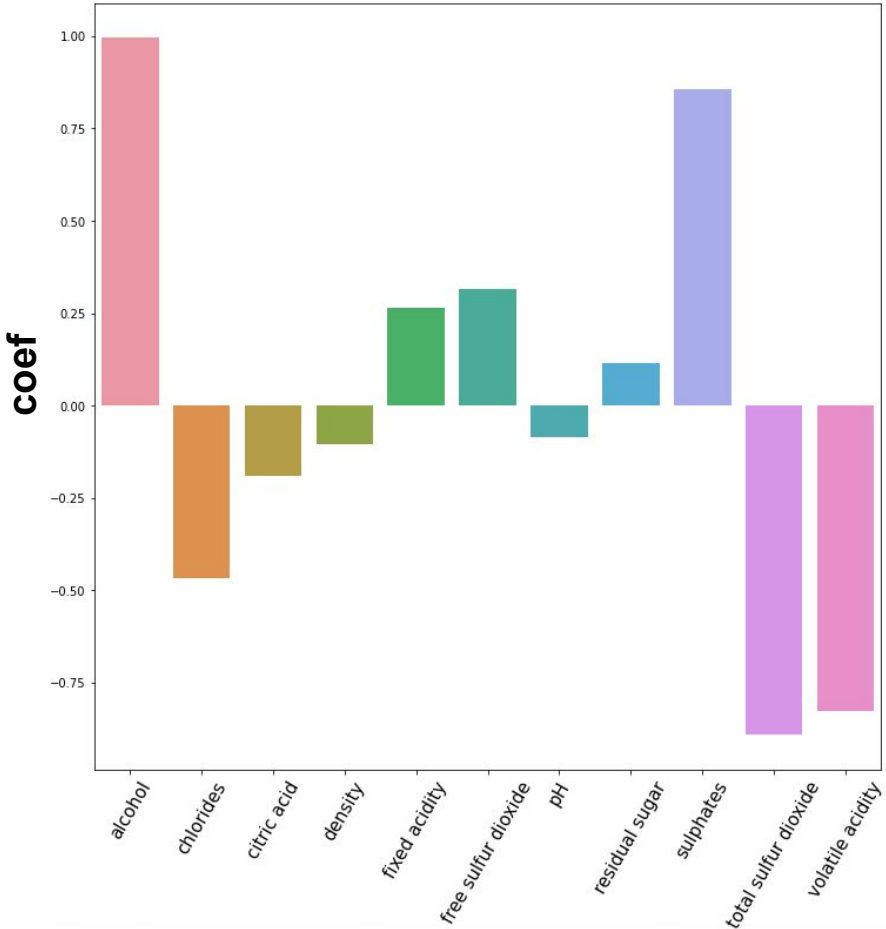
- 변수 간 상관 관계가 높아 분석에 부정적인 영향을 미치는 것
- 다중 공산성은 분산팽창요인(VIF)이라는 계수로 평가한다.
- VIF 계수가 10 ~ 15 정도를 넘으면 그 피처는 다중 공선성의 문제가 발생했다고 판단한다.
- 다중 공선성의 문제를 해결하는 일반적인 방법은 해당 피처를 drop

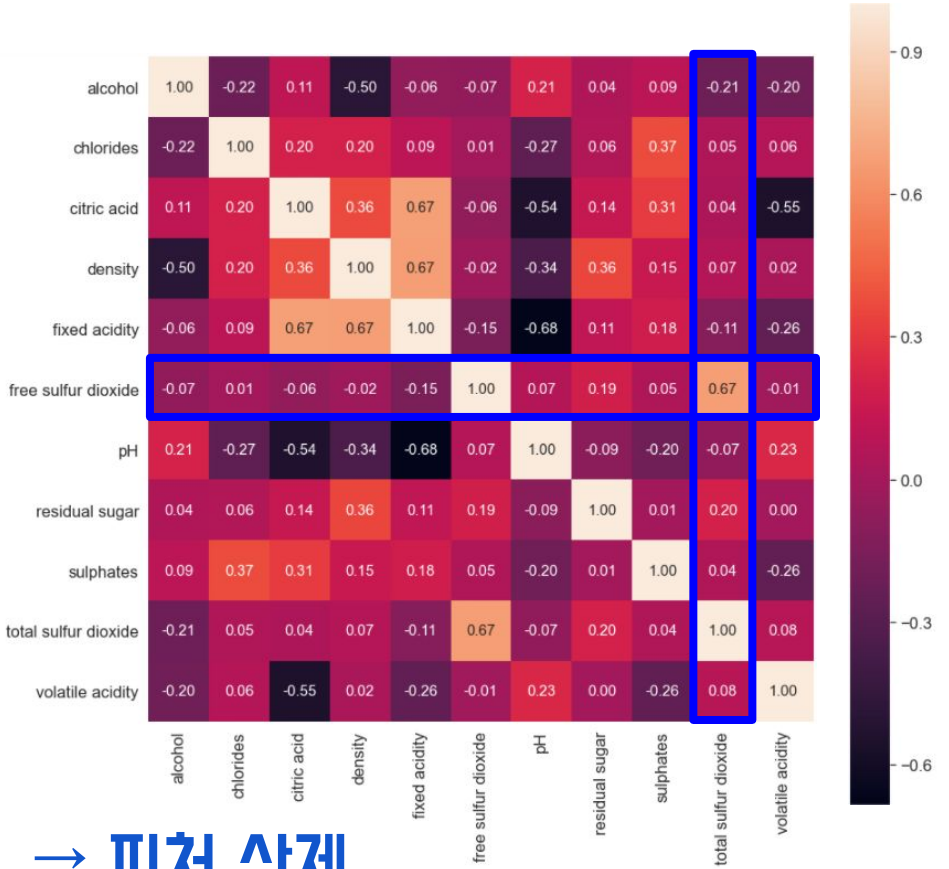
VIF 계수 10이상 존재 X

(피처 간 상관 관계를 자세히 알지 못함)

상관 계수 그래프 및 히트맵

데이터 전처리 피처 상관도 분석

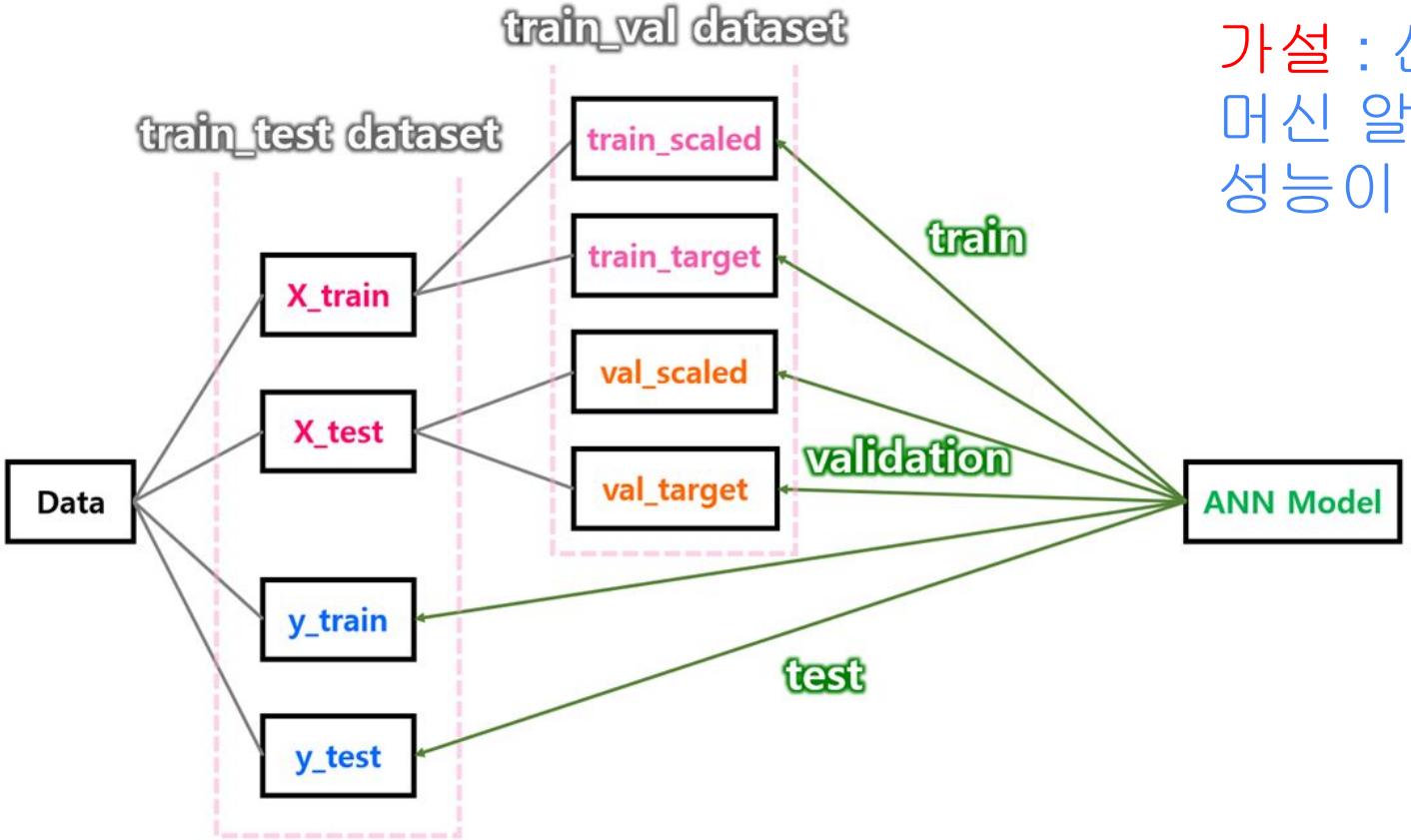




→ **피처 삭제**

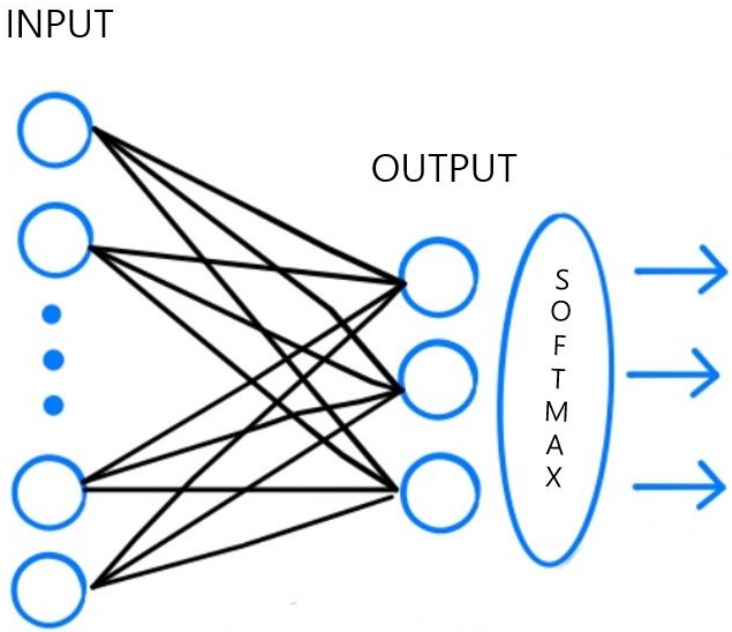
free sulfur dioxide , total sulfur dioxide

데이터 예측 ANN 모델 기법



가설 : 신경망이 다른 머신 알고리즘보다 성능이 좋지 않을까?

데이터 예측 ANN 모델 기법



optimizer = 'adam'

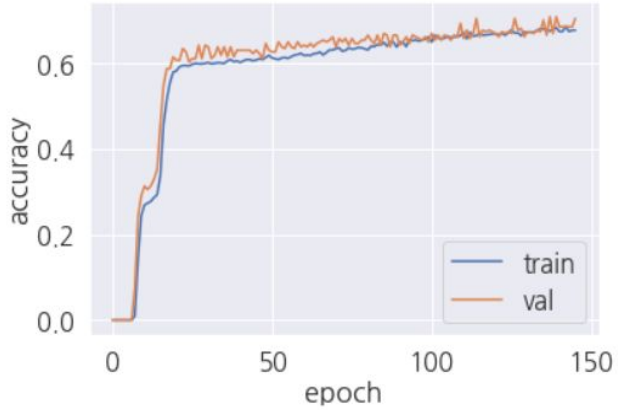
loss = 'sparse_categorical_crossentropy'

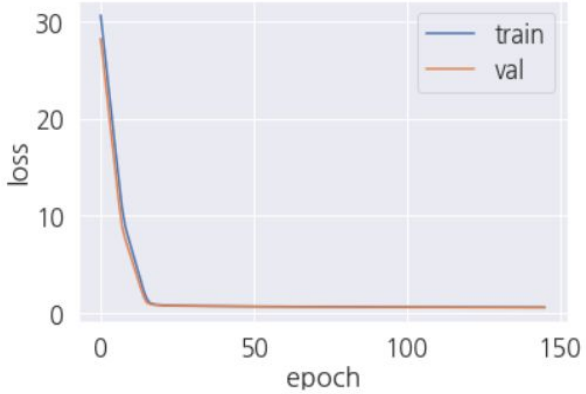
metrics = 'accuracy'

	원본 데이터	정제된 데이터
정확도	0.688 →	0.753
손실	0.58536 →	0.53624

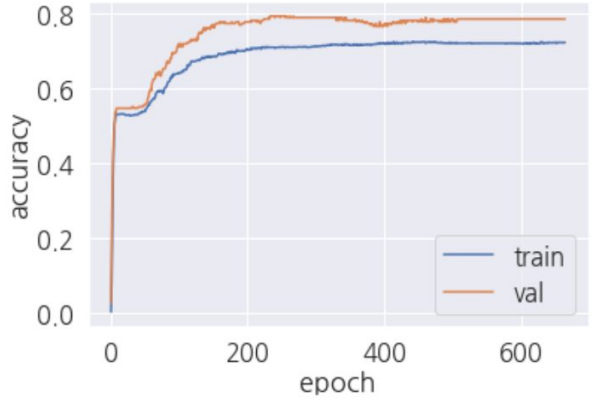
데이터 예측 ANN 모델 기법

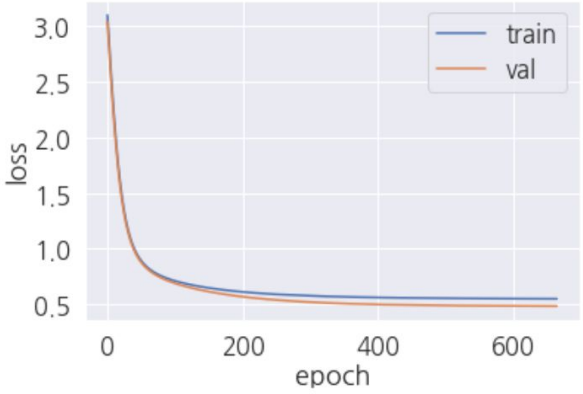
원본 데이터를 이용한 ANN



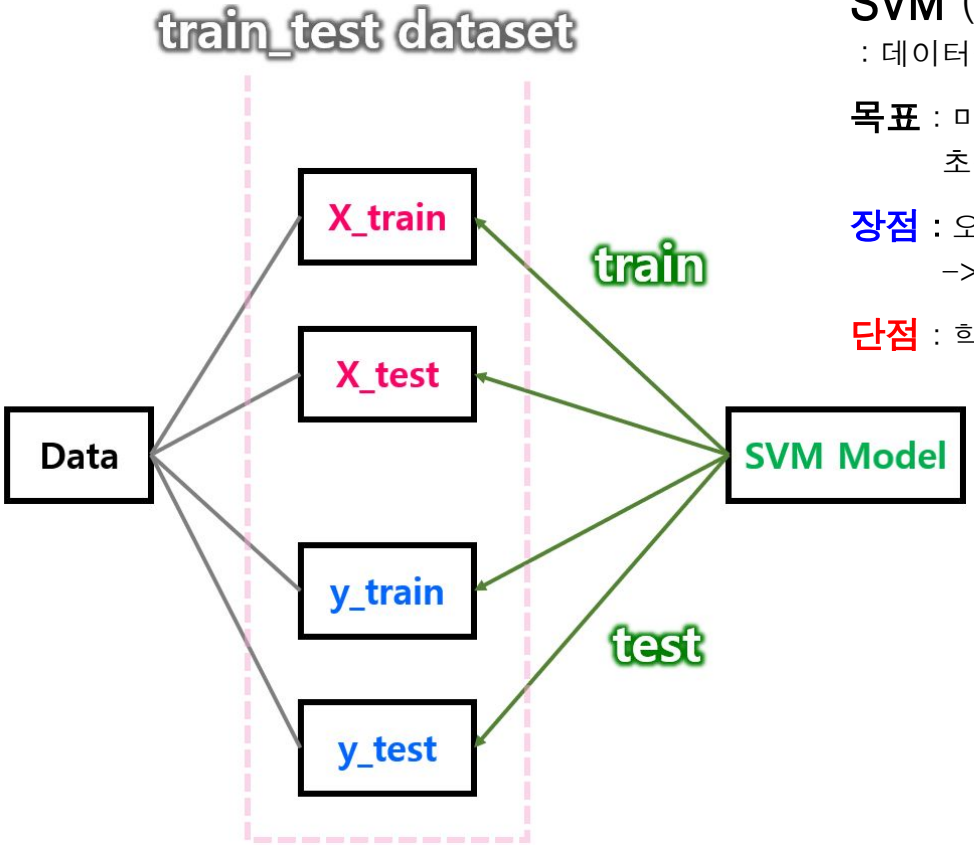


정제된 데이터를 이용한 ANN





데이터 예측 SVM 모델 기법



SVM (support vector machine)

: 데이터를 선형으로 분리하는 최적의 선형 결정 경계를 찾는 알고리즘

목표 : 마진(양쪽 support vector와 판간의 거리 합)을 최대화하는 초평판(데이터등을 분류할 수 있는 칸막이)을 찾는 것

장점 : 오류데이터 영향이 적음, 과적합되는 경우가 적음, 사용하기 쉬움
-> 분류, 수치예측 에서 많이 사용

단점 : 학습 속도 느림, 설명력이 떨어지고 복잡함.

목표 : 하이퍼 파라미터 튜닝

성능 비교

데이터 예측 SVM 모델 기법

하이퍼 파라미터 오토 튜닝 : GridSearchCV

SVM Parameter	C	: float, default=1.0
	kernel	:{'linear','poly','rbf','sigmoid','precomputed'} default='rbf'
	degree	: int, default=3
	gamma	: {'scale', 'auto'} or float, default='scale'
	coef0	: float, default=0.0
	shrinking	: bool, default=True
	probability	: bool, default=False
	tol	: float, default=1e-3
	cache_size	: float, default=200
	class_weight	: dict or 'balanced', default=None
	verbose	: bool, default=False
	max_iter	: int, default=-1
	decision_function_shape	: {'ovo', 'ovr'}, default='ovr'
	break_ties	: bool, default=False
	random_state	: int, RandomState instance or None, default=None

GridSearchCV로 C, kernel, gamma 값 튜닝

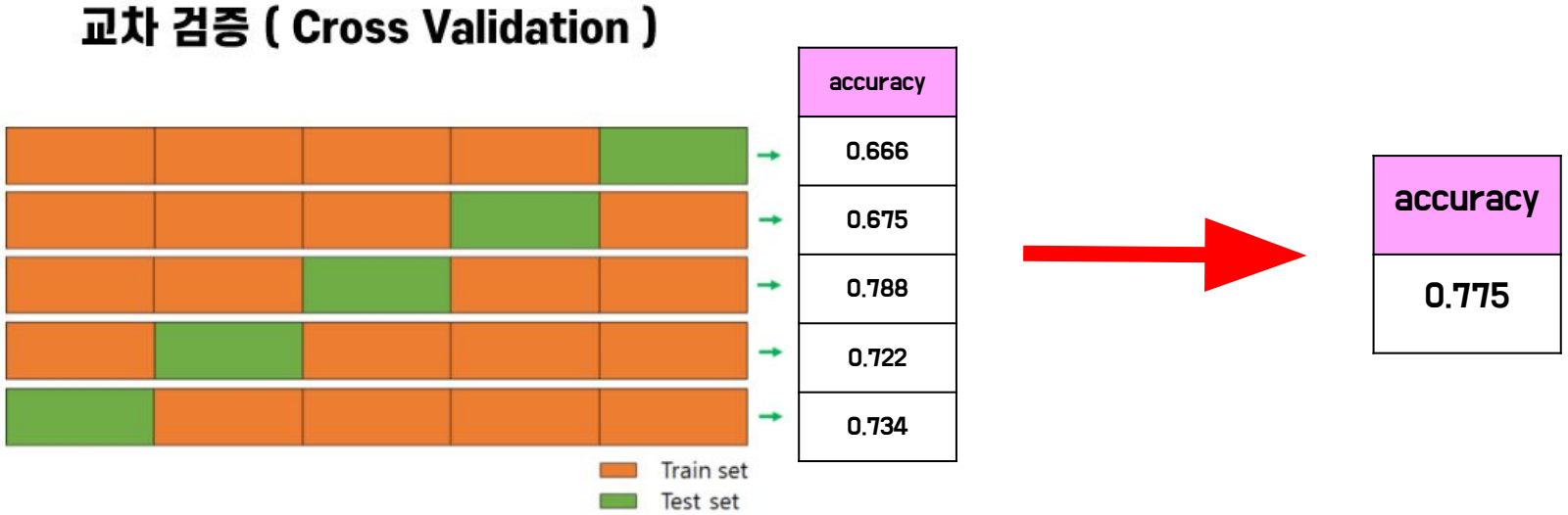
C : 정규화 매개 변수. 정규화의 강도는 C에 반비례

kernel : 커널 유형

gamma : 커널 계수

하이퍼 파라미터	수정 전	수정 후
원본 데이터	0.755 →	0.775
정제된 데이터	0.740 →	0.773

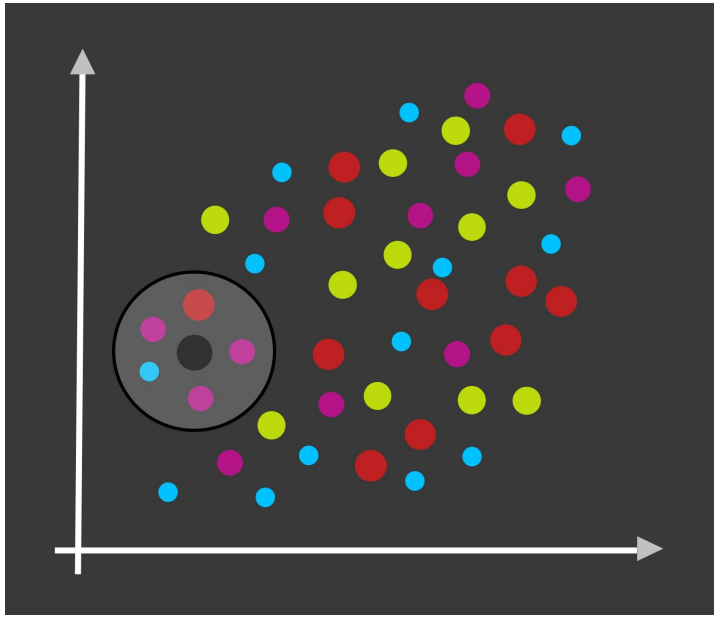
데이터 예측 SVM 모델 기법



분류에서 ANN보다 SVM의 성능이 좋다.

K-Nearest Neighbor(최근접 이웃)

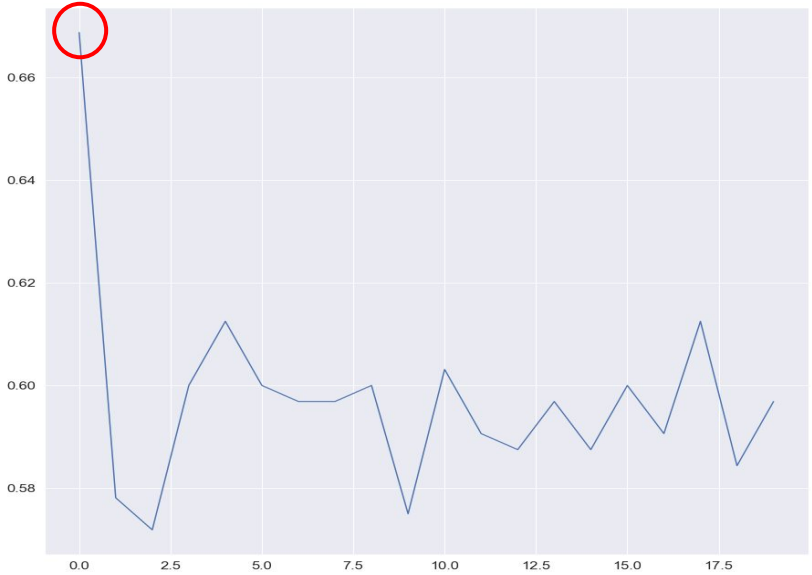
- 특별한 예측 모델없이 가장 가까운 데이터 포인트를 기반으로 예측을 수행하는 방법
- 분류와 회귀 모두 지원



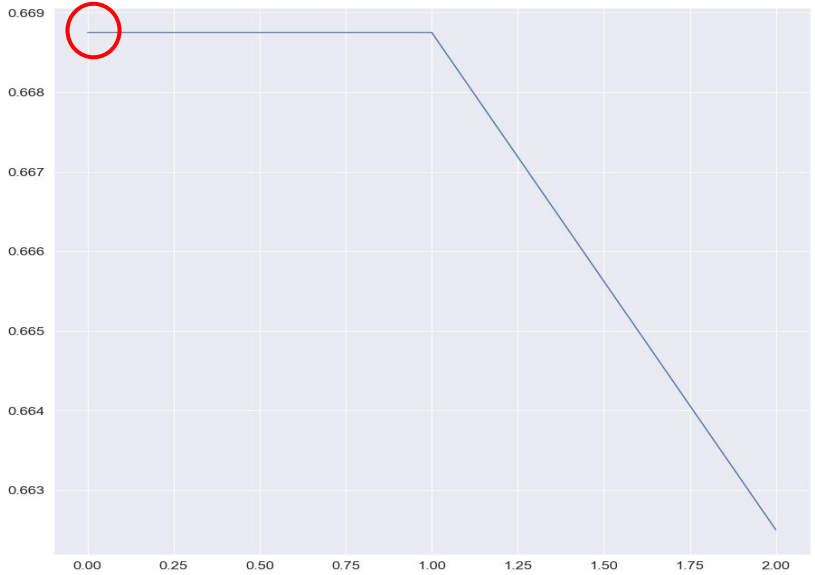
장점)
높은 정확도
오류 데이터가 결과에 영향을 미치지 않음
데이터에 대한 가정이 없음

단점)
느린속도
고용량 메모리사용

하이퍼 파라미터 튜닝



n_neighbors = 1



metric = 'minkowski', 'euclidean'

etc. weights, algorithm , leaf_size, p, n_jobs (변화 없음)

하이퍼 파라미터 튜닝 결과

	수정 전	수정 후
하이퍼 파라미터	0.6125	0.6688

0.0563 증가 ⇧

Logistic Regression

 # red_final , y 데이터 split

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = \
train_test_split(red_final2, y, stratify=y, test_size=0.2, random_state=42)
```

red_final 데이터 LogisticReression 모델 훈련 및 평가

```
x1 = x_train.values
y1 = y_train.values
from sklearn.linear_model import LogisticRegression
clf = LogisticRegression(C=10, solver='newton-cg', max_iter=100, random_state=42).fit(x1, y1)
predict = clf.predict(x_test)
accuracy= (predict == y_test).sum() /len(y1)
print(accuracy)      = >   0.2501954652071931
```

sklearn accuracy score 추가

```
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test, predict)
print(accuracy)    => 0.740625
```

경사하강법 클래스

```
from sklearn.linear_model import SGDClassifier

sgd = SGDClassifier(loss='log', max_iter=100, tol=1e-3, random_state=42)

sgd.fit(x_train, y_train)

sgd.score(x_test,y_test)
```

=> 0.6125

Logistic Regression

 # X_scaled, y 데이터 split

```
from sklearn.model_selection import train_test_split
x_train2, x_test2, y_train2, y_test2 = train_test_split(X_scaled, y, stratify=y, test_size=0.2, random_state=42)
```

X_scaled 데이터 LogisticRegression 모델 훈련 및 평가

```
x3 = x_train2.values
y3 = y_train2.values
from sklearn.linear_model import LogisticRegression
clf = LogisticRegression(solver='newton-cg', max_iter=100).fit(x3, y3)
predict = clf.predict(x_test2)
accuracy= (predict == y_test2).sum() /len(y3)
print(accuracy)   =>  0.18451915559030493
```

sklearn accuracy_score 추가

```
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test2, predict)
print(accuracy)   =>  0.71875
```

경사하강법 클래스

```
from sklearn.linear_model import SGDClassifier


sgd = SGDClassifier(loss='log', max_iter=100, tol=1e-3, random_state=42)
sgd.fit(x_train2, y_train2)
sgd.score(x_test2,y_test2)   =>   0.71875
```

의사결정 나무(Decision Tree)

최적의 depth 찾기

```
score_li = []
for depth in range(1,11):
    tree = DecisionTreeClassifier(criterion="entropy",
max_depth = depth, random_state = 42)
    tree.fit(X_train, y_train)
    score = tree.score(X_test, y_test)
    score_li.append(score)
    print(depth, '번째 : ' ,score, end=" | ")
print('\n최고점수 : ', max(score_li) )
```

최적의 depth : 7

```
01 번째 :  0.7175 | 02 번째 :  0.7175 |
03 번째 :  0.6775 | 04 번째 :  0.7225 |
05 번째 : 0.71250 | 06 번째 :  0.715 |
07 번째 :  0.7275 | 08 번째 :  0.7050 |
09 번째 :  0.7100 | 10 번째 :  0.7000 |
최고점수 :  0.7275
```

```
훈련 세트 정확도: 0.818
테스트 세트 정확도: 0.728
```

랜덤 포레스트(Random Forest)

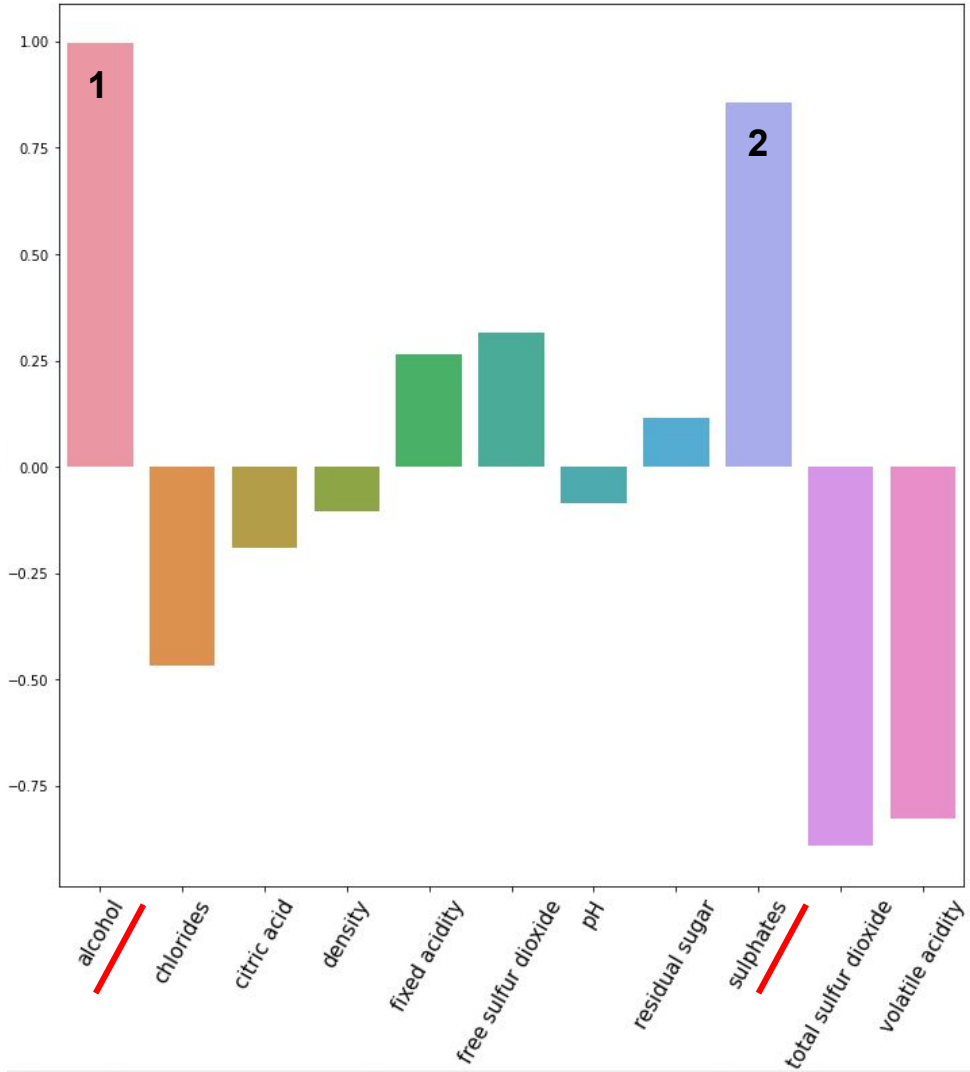
최적의 n_estimators(결정트리) 찾기

```
score_li = []
for num in range(1,24,2):
    tree_r = RandomForestClassifier(
    n_estimators = num, max_depth = 7,
    random_state=42)
    tree_r.fit(X_train, y_train)
    score = tree_r.score(X_test, y_test)
    print(num,' : ', score, end=" | ")
    score_li.append(score)

max(score_li)
```

최적의 n_estimators : 19

```
01 : 0.7050 | 03 : 0.7375 |
05 : 0.7550 | 07 : 0.7575 |
09 : 0.7275 | 11 : 0.7375 |
13 : 0.7425 | 15 : 0.7600 |
17 : 0.7650 | 19 : 0.7675 |
21 : 0.7650 | 23 : 0.7675 |
```

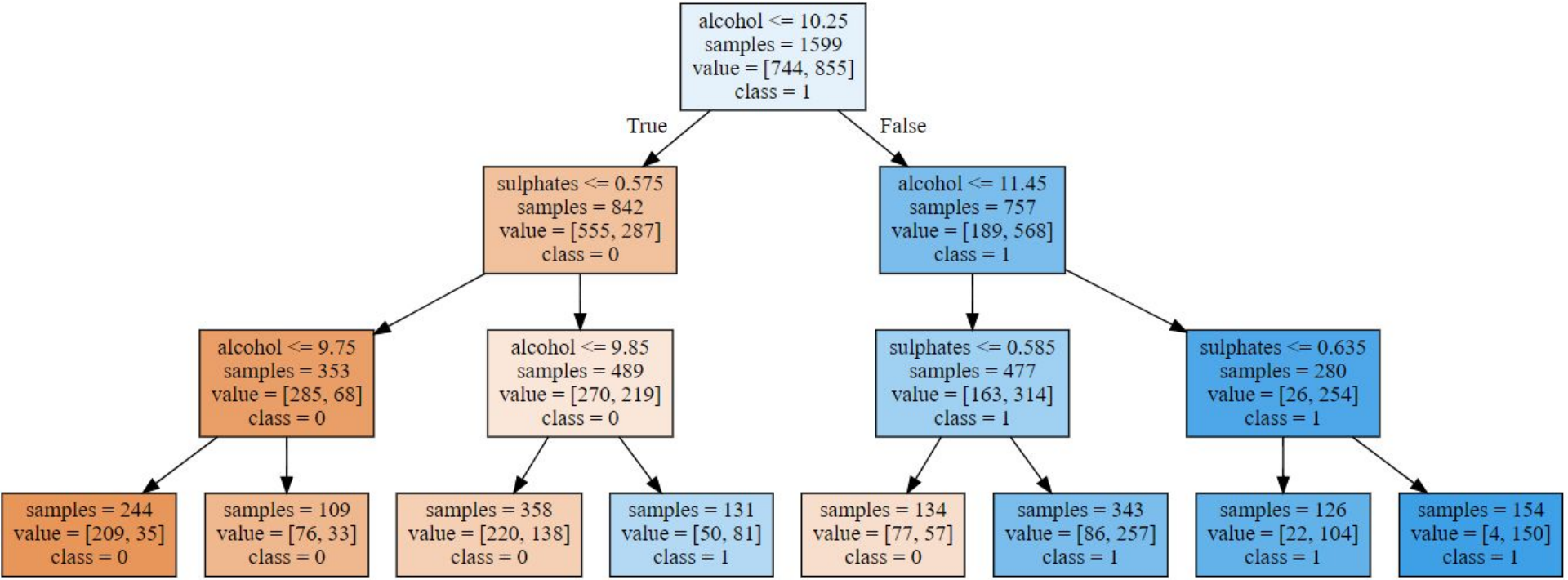
```
훈련 세트 정확도: 0.880
테스트 세트 정확도: 0.767
```



와인의 등급과 상관관계

- Alcohol - 알코올
- Sulphates - 황산염
- Free sulfur dioxide
- Fixed dioxide
- pH

graphviz를 이용한 시각화



정규화 데이터 사용

최적의 depth : 10 / n_estimators(결정트리) : 17

```
1 print("훈련 세트 정확도: {:.3f}".format(tree.score(X_train2, y_train2)))
2 print("테스트 세트 정확도: {:.3f}".format(tree.score(X_test2, y_test2)))
```

```
훈련 세트 정확도: 0.919
테스트 세트 정확도: 0.740
```

```
1 print("훈련 세트 정확도: {:.3f}".format(tree_r.score(X_train2, y_train2)))
2 print("테스트 세트 정확도: {:.3f}".format(tree_r.score(X_test2, y_test2)))
```

```
훈련 세트 정확도: 0.969
테스트 세트 정확도: 0.782
```

	의사결정 나무	랜덤 포레스트
train	0.818	0.880
test	0.728	0.767

train	0.919	0.969
test	0.740	0.782

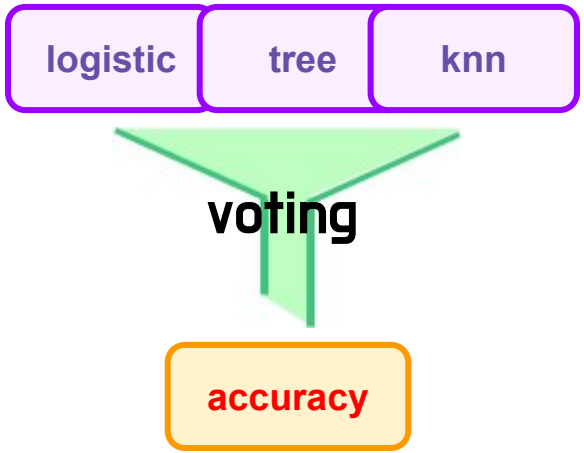
- 의사결정나무보다 랜덤포레스트일때 예측도가 상승하였음.
- 정규화된 데이터는 예측도가 더 높으나 과적합의 가능성이 크다.

데이터 예측 앙상블 기법

- 약한 분석 알고리즘 묶어서 한 번에 분석(예측)

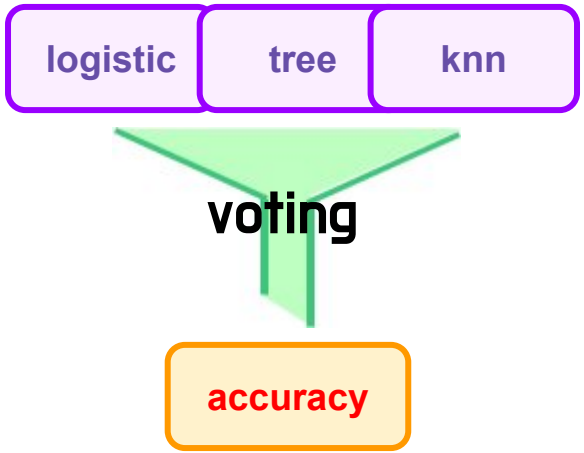
- 투표(Voting) 시스템으로 점수가 높은 것
  (많이 선택한 것)으로 선택 (다수결, 평균)

- 방식 : 배깅, 부스팅
  배깅 : 샘플을 여러번 뽑아 학습시킨 후 집계 (병렬)
  부스팅 : 가중치를 부여하여 개선 (직렬)



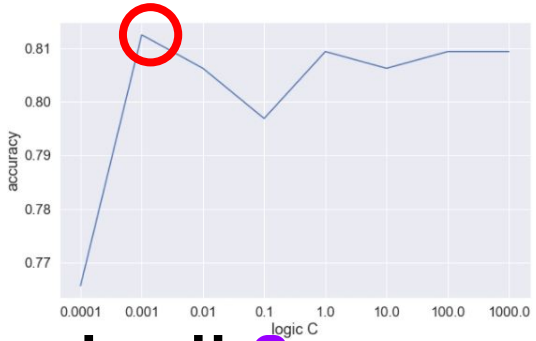
데이터 예측 앙상블 기법

```
# 로지스틱, 결정트리, 최근접이웃
logistic = LogisticRegression(C=0.001, random_state=42)
tree = DecisionTreeClassifier(max_depth=None, criterion='entropy', random_state=42)
knn = KNeighborsClassifier(n_neighbors=1, p=2, metric='manhattan')

voting_estimators = [('logistic', logistic), ('tree', tree), ('knn', knn)]

voting = VotingClassifier(estimators = voting_estimators, voting='soft')

clf_labels = ['Logistic regression', 'Decision tree', 'KNN', 'Majority voting']
all_clf = [logistic, tree, knn, voting]


voting.fit(X_train, y_train)
pred = voting.predict(X_test)
print('보팅 분류기의 정확도 : {0:.4f}'.format(accuracy_score(y_test, pred)))
```



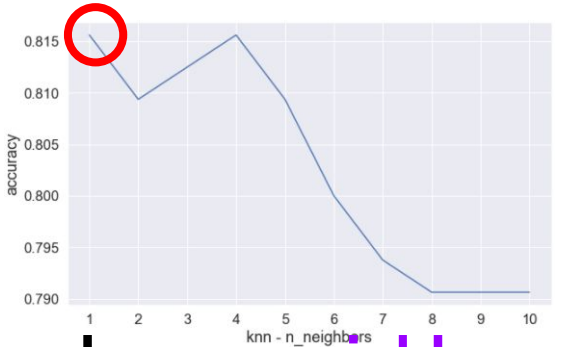
보팅 분류기의 정확도 : 0.8156

데이터 예측 앙상블 기법 - 하이퍼 파라미터 튜닝

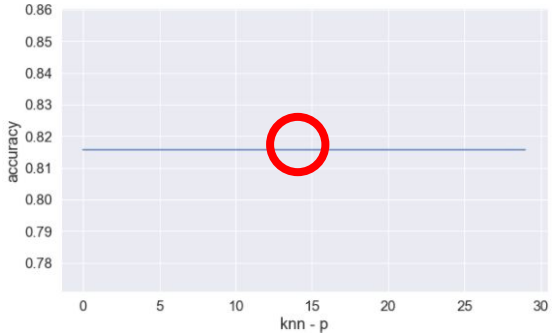


Logit C (penalty)

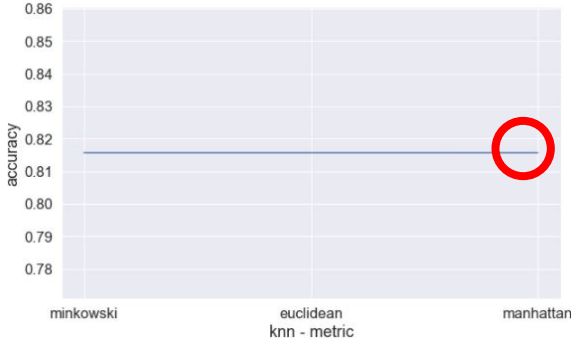
Tree max_depth



knn n_neighbors



knn p



knn metric

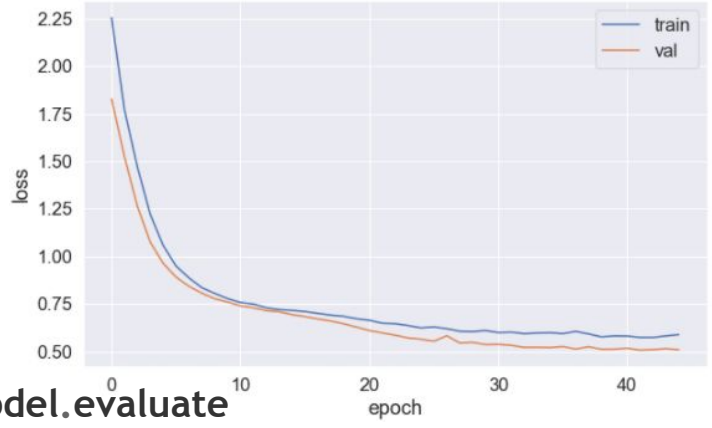
voting soft/hard

데이터 예측 DNN 모델 (심층신경망)

Model: "sequential"

Layer (type)	Output Shape	Param #	activation
dense (Dense)	(None, 100)	1000	relu
dense_1 (Dense)	(None, 32)	3232	relu
dropout (Dropout)	(None, 32)	0	
dense_2 (Dense)	(None, 16)	528	sigmoid
dense_3 (Dense)	(None, 10)	170	softmax

Total params: 4,930
Trainable params: 4,930
Non-trainable params: 0



model.evaluate (X_train, y_train)

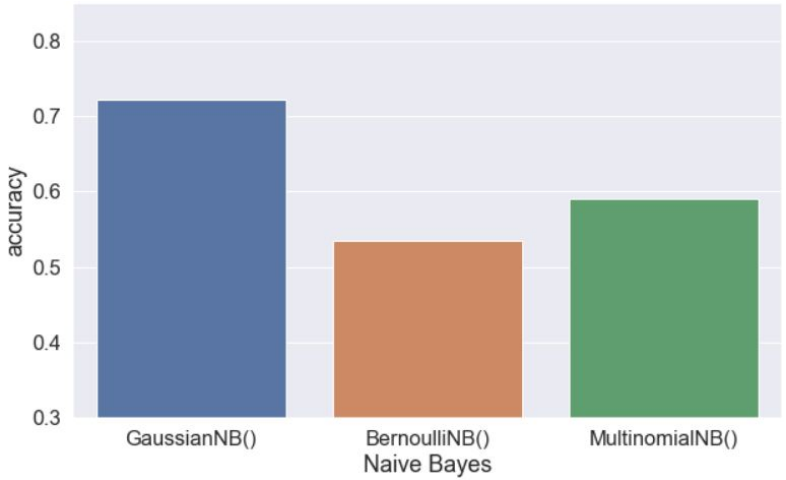
[0.5501018331981059, 0.7419859]

```
model.compile(optimizer = 'adam', loss = 'sparse_categorical_crossentropy', metrics = ['accuracy'])

# 복구점
checkpoint_cb = keras.callbacks.ModelCheckpoint('wine-model.h5')
# 조기종료
early_stopping_cb = keras.callbacks.EarlyStopping(patience=3, restore_best_weights=True)

# 모델 학습
history = model.fit(X_train, y_train, epochs = 300, callbacks=[checkpoint_cb, early_stopping_cb],
validation_split=0.2)
```

데이터 예측 나이브 베이즈 분류 기법

- 언어와 관련된 부분에서 많이 사용
- 이산적인 속성(0, 1)에 특화
- 대표적인 문제 : 메일 구분(스팸/햄)

```
gnb = GaussianNB()
brn = BernoulliNB()
mn = MultinomialNB()

acc_li = []

for nb in [gnb, brn, mn]:
    y_pred = nb.fit(X_train, y_train).predict(X_test)
    acc = (y_test == y_pred).sum() / len(y_pred)
    acc_li.append(acc)
    print(nb, 'pred accuracy :', acc)
```



0.721875 0.534375 0.590625

- 결론

- 신경망 보다는 머신러닝의 알고리즘이 분류의 효과가 높았음.
- 하이퍼 파라미터 튜닝 보단 정제한 데이터를 사용하는게 효과적
- 앙상블이 예측율 가장 높음 = 81.56%

- 향후 과제

- 과대/ 과소 적합 완화

색 붉고 반짝반짝 빛이 남
갈색, 혼탁함 → 변질된 와인

향 풀, 과일 향 등 여러 미묘한 향이 오래 지속
잔을 돌려 살짝 파도치게 한 후 코 밑에 대고 향을 맡음

맛 공기를 빨아들여 입 안에서 서서히 굴리며 맛 봄
단 맛, 신 맛, 떫은 맛(탄닌)이 함께 조화를 이뤄야 함

좋은 와인은
탄닌, 산, 단 맛, 과일 향과 다른 여러 성분들이
조화와 균형을 이루고 있어야 합니다.


감 사 합 니 다