

포도주 평가 데이터를 이용한 포도주 점수 예측하기

1491051박병민

1. 프로젝트 주제 및 배경

최근 뉴스를 보던 중 우리나라의 포도주 소비가 급성장하고 있다는 뉴스를 보았습니다. 특히 이어 홈술, 혼술등의 소비 키워드로 인하여 레스토랑, 호텔, 바 등의 소비보다 마트 와인숍등의 소비의 매출이 높다는 기사였습니다. 이 기사를 보던 중 와인병 라벨의 표기된 정보들을 가지고 좋은 점수와 평가를 받는 와인인지 아닌지를 소비자가 판별할 수 있다면 좋겠다는 생각을 하게 되었고 와인리뷰를 통한 와인점수 예측하기라는 프로젝트를 진행하게 되었습니다.

2. 데이터 전처리

country	description	designation	points	price	province	region_1	region_2	taster_name	taster_twitter_handle	title	variety	winery
Italy	Aromas include tropical fruit, broom, brimston...	Vulkà Bianco	87	NaN	Sicily & Sardinia	Etna	NaN	Kerin O'Keefe	@kerinokeefe	Nicosia 2013 Vulkà Bianco (Etna)	White Blend	Nicosia
Portugal	This is ripe and fruity, a wine that is smooth...	Avidagos	87	15.0	Douro	NaN	NaN	Roger Voss	@vossroger	Quinta dos Avidagos 2011 Avidagos Red (Douro)	Portuguese Red	Quinta dos Avidagos
US	Tart and snappy, the flavors of lime flesh and...	NaN	87	14.0	Oregon	Willamette Valley	Willamette Valley	Paul Gregutt	@paulgwine	Rainstorm 2013 Pinot Gris (Willamette Valley)	Pinot Gris	Rainstorm
US	Pineapple rind, lemon pith and orange blossom ...	Reserve Late Harvest	87	13.0	Michigan	Lake Michigan Shore	NaN	Alexander Peartree	NaN	St. Julian 2013 Reserve Late Harvest Riesling ...	Riesling	St. Julian
US	Much like the regular bottling from 2012, this...	Vintner's Reserve Wild Child Block	87	65.0	Oregon	Willamette Valley	Willamette Valley	Paul Gregutt	@paulgwine	Sweet Cheeks 2012 Vintner's Reserve Wild Child...	Pinot Noir	Sweet Cheeks

제가 사용한 데이터는 포도주 평가 데이터로 나라, 맛 설명, 포도가 생산된 포도밭, 점수, 가격, 지방, 지역_1, 지역_2, 맛을 본 사람, 맛을 본 사람 트위터, 포도주 등급, 포도주 제작자의 데이터가 들어있습니다. 이 중에서 어떠한 데이터가 점수에 영향을 주는지가 중요했기 때문에 우선 포도주 점수에 영향을 많이 미치는 요인이 무엇인지와 포도주 라벨에 표기된 내용을 조사하였습니다. 우선 라벨에 표기된 내용은 국가마다 달랐기 때문에 공통으로 포함하고 있는 값인 국가, 포도주 제작자, 포도주 생산지방을 사용하였습니다. 또한, 포도주 점수에 영향을 주는 요인 같은 경우에는 포도주 등급을 사용하였습니다. 조사한 바에 따르면 포도주 등급이 높을수록 맛이 좋고 그렇다면 점수에 큰 영향을 줄 수 있다고 생각하여 사용할 데이터로 추가하였습니다.

그리하여 제가 최종적으로 사용한 데이터값은 국가, 포도주 생산지방, 포도주 등급, 포도주 제작자입니다.

	country	points	province	variety	winery
0	Italy	87	Sicily & Sardinia	White Blend	Nicosia
1	Portugal	87	Douro	Portuguese Red	Quinta dos Avidagos
2	US	87	Oregon	Pinot Gris	Rainstorm
3	US	87	Michigan	Riesling	St. Julian
4	US	87	Oregon	Pinot Noir	Sweet Cheeks
...
129966	Germany	90	Mosel	Riesling	Dr. H. Thanisch (Erben Müller-Burggraef)
129967	US	90	Oregon	Pinot Noir	Citation
129968	France	90	Alsace	Gewürztraminer	Domaine Gresser
129969	France	90	Alsace	Pinot Gris	Domaine Marcel Deiss
129970	France	90	Alsace	Gewürztraminer	Domaine Schoffit

129971 rows × 5 columns

원본데이터에서 사용값들을 뽑아낸 후 새로운 데이터 프레임을 만들었고 결측값이 존재하지 않았기 때문에 결측치 제거는 따로 하지 않았고 one-hot encoding만 하여 사용하였습니다.

<코드>

```
import csv
import pandas as pd

df = pd.read_csv('review1(원본).csv')
wine_review = df[['country', 'province', 'variety', 'winery']]
wine_review.to_csv("wine_review.csv")
```

3. 분석 결과의 정확도 및 다양한 분석 방법 시도

3.1 SVM 방식을 사용한 정확도 측정

1) 데이터 5000개로 테스트

<코드>

```
import csv
import pandas as pd
from sklearn import svm, metrics
from sklearn.model_selection import train_test_split
wine_review = pd.read_csv('wine_review.csv')
wine_review = wine_review.drop(columns = ['Unnamed: 0'], axis = 1)
wine_review = wine_review.iloc[0:5000]
wine_review_encode = pd.get_dummies(wine_review, prefix=['country',
'province', 'variety', 'winery'])
wine_data = wine_review_encode
wine_label = wine_review_encode["points"]
train_data, test_data, train_label, test_label = \
train_test_split(wine_data, wine_label)
clf = svm.SVC(gamma='auto')
clf.fit(train_data, train_label)
pre = clf.predict(test_data)
result = pd.DataFrame({"label": test_label, "prediction":pre})
ac_score = metrics.accuracy_score(test_label, pre)
print("\n정답률 =", ac_score)
```

```
1 wine_data = wine_review_encode
2 wine_label = wine_review_encode["points"]
3
4 train_data, test_data, train_label, test_label = \
5 train_test_split(wine_data, wine_label)
6
7 clf = svm.SVC(gamma='auto')
8 clf.fit(train_data, train_label)
9 pre = clf.predict(test_data)
10
11 result = pd.DataFrame({"label": test_label, "prediction":pre})
12 ac_score = metrics.accuracy_score(test_label, pre)
13 print("\n정답률 =", ac_score)
```

정답률 = 0.1352

2) 데이터 10000개로 테스트

<코드>

```
import csv
import pandas as pd
from sklearn import svm, metrics
from sklearn.model_selection import train_test_split
wine_review = pd.read_csv('wine_review.csv')
wine_review = wine_review.iloc[0:10000]
wine_review_encode = pd.get_dummies(wine_review, prefix=['country',
'province', 'variety', 'winery'])
wine_data = wine_review_encode
wine_label = wine_review_encode["points"]
train_data, test_data, train_label, test_label = \
train_test_split(wine_data, wine_label)
clf = svm.SVC(gamma='auto')
clf.fit(train_data, train_label)
pre = clf.predict(test_data)
result = pd.DataFrame({"label": test_label, "prediction":pre})
ac_score = metrics.accuracy_score(test_label, pre)
print("\n정답률 =", ac_score)
```

```
: 1 wine_review = pd.read_csv('wine_review.csv')
2
3 wine_review = wine_review.iloc[0:10000]
4
5 wine_review_encode = pd.get_dummies(wine_review, prefix=['country', 'province',
6
7 wine_data = wine_review_encode
8 wine_label = wine_review_encode["points"]
9
10 train_data, test_data, train_label, test_label = \
11 train_test_split(wine_data, wine_label)
12
13 clf = svm.SVC(gamma='auto')
14 clf.fit(train_data, train_label)
15 pre = clf.predict(test_data)
16
17 result = pd.DataFrame({"label": test_label, "prediction":pre})
18 ac_score = metrics.accuracy_score(test_label, pre)
19 print("\n정답률 =", ac_score)
```

정답률 = 0.7644

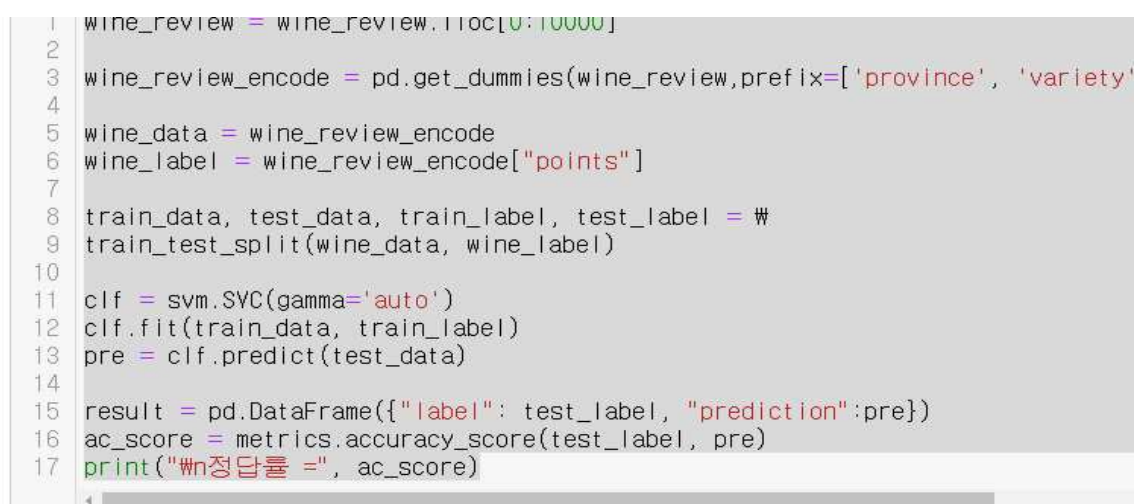
데이터의 수를 5000개로 하였을 때 정확도가 너무 낮아 데이터의 수를 올려 다시 테스트하였습니다.

3) 사용하는 데이터 중 나라를 제외하고 데이터 수는 10000개 테스트

<코드>

```
import csv
import pandas as pd
from sklearn import svm, metrics
from sklearn.model_selection import train_test_split

wine_review = pd.read_csv('wine_review.csv')
wine_review = wine_review.drop(columns = ['Unnamed: 0', 'country'], axis = 1)
wine_review = wine_review.iloc[0:10000]
wine_review_encode = pd.get_dummies(wine_review, prefix=['province', 'variety', 'winery'])
wine_data = wine_review_encode
wine_label = wine_review_encode["points"]
train_data, test_data, train_label, test_label = \
train_test_split(wine_data, wine_label)
clf = svm.SVC(gamma='auto')
clf.fit(train_data, train_label)
pre = clf.predict(test_data)
result = pd.DataFrame({"label": test_label, "prediction":pre})
ac_score = metrics.accuracy_score(test_label, pre)
print("\n정답률 =", ac_score)
```



```
1 wine_review = wine_review.iloc[0:10000]
2
3 wine_review_encode = pd.get_dummies(wine_review, prefix=['province', 'variety', 'winery'])
4
5 wine_data = wine_review_encode
6 wine_label = wine_review_encode["points"]
7
8 train_data, test_data, train_label, test_label = #
9 train_test_split(wine_data, wine_label)
10
11 clf = svm.SVC(gamma='auto')
12 clf.fit(train_data, train_label)
13 pre = clf.predict(test_data)
14
15 result = pd.DataFrame({"label": test_label, "prediction":pre})
16 ac_score = metrics.accuracy_score(test_label, pre)
17 print("\n정답률 =", ac_score)
```

정답률 = 0.252

정확도를 올리기 위해 사용하는 데이터의 종류를 바꿔서 테스트해보았습니다.

4)사용하는 데이터 중 나라를 제외하고 데이터 수는 20000개 테스트

<코드>

```
import csv
import pandas as pd
from sklearn import svm, metrics
from sklearn.model_selection import train_test_split
wine_review = wine_review.iloc[0:20000]
wine_review_encode = pd.get_dummies(wine_review,prefix=['province',
'variety', 'winery'])
wine_data = wine_review_encode
wine_label = wine_review_encode["points"]
train_data, test_data, train_label, test_label = \
train_test_split(wine_data, wine_label)
clf = svm.SVC(gamma='auto')
clf.fit(train_data, train_label)
pre = clf.predict(test_data)
result = pd.DataFrame({"label": test_label, "prediction":pre})
ac_score = metrics.accuracy_score(test_label, pre)
print("\n정답률 =", ac_score)
```

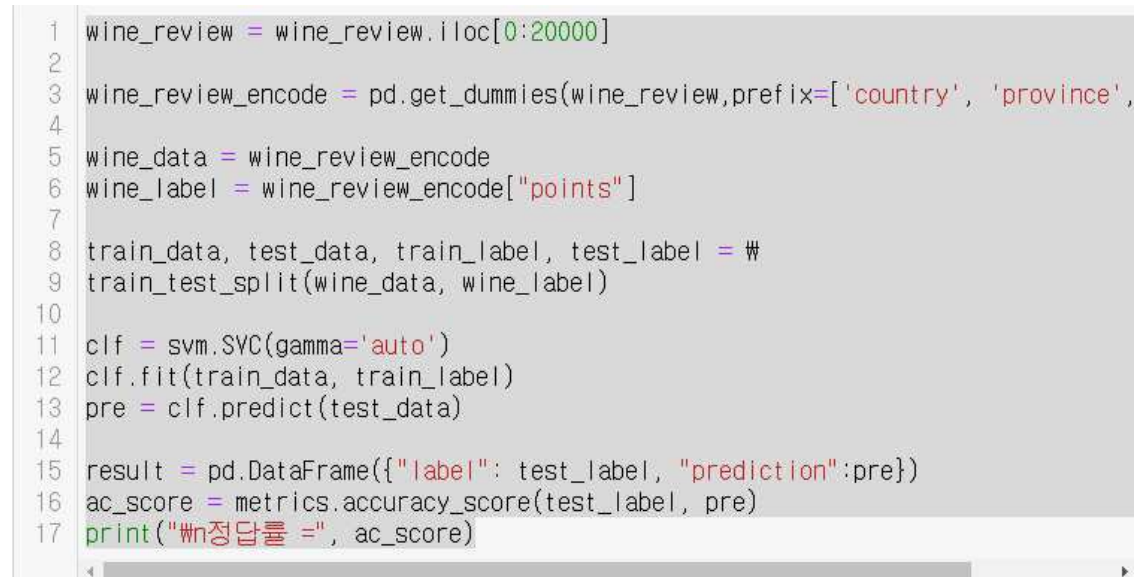
정답률 = 0.3332

정확도가 너무 낮아 데이터를 올리기 위해 데이터의 수를 높여서 테스트해보았습니다.

5)사용하는 데이터 중 포도주 제작자를 제외하고 데이터 수는 20000개 테스트

<코드>

```
import csv
import pandas as pd
from sklearn import svm, metrics
from sklearn.model_selection import train_test_split
wine_review = pd.read_csv('wine_review.csv')
wine_review = wine_review.drop(columns = ['Unnamed: 0', 'winery'], axis =
1)
wine_review = wine_review.iloc[0:20000]
wine_review_encode = pd.get_dummies(wine_review,prefix=['country',
'province', 'variety'])
wine_data = wine_review_encode
wine_label = wine_review_encode["points"]
train_data, test_data, train_label, test_label = \
train_test_split(wine_data, wine_label)
clf = svm.SVC(gamma='auto')
clf.fit(train_data, train_label)
pre = clf.predict(test_data)
result = pd.DataFrame({"label": test_label, "prediction":pre})
ac_score = metrics.accuracy_score(test_label, pre)
print("\n정답률 =", ac_score)
```



```
1 wine_review = wine_review.iloc[0:20000]
2
3 wine_review_encode = pd.get_dummies(wine_review,prefix=['country', 'province',
4
5 wine_data = wine_review_encode
6 wine_label = wine_review_encode["points"]
7
8 train_data, test_data, train_label, test_label = \
9 train_test_split(wine_data, wine_label)
10
11 clf = svm.SVC(gamma='auto')
12 clf.fit(train_data, train_label)
13 pre = clf.predict(test_data)
14
15 result = pd.DataFrame({"label": test_label, "prediction":pre})
16 ac_score = metrics.accuracy_score(test_label, pre)
17 print("\n정답률 =", ac_score)
```

정답률 = 0.9382

6)사용하는 데이터 중 포도주 제작자를 제외하고 데이터 수는 30000개 테스트

<코드>

```
import csv
import pandas as pd
from sklearn import svm, metrics
from sklearn.model_selection import train_test_split
wine_review = pd.read_csv('wine_review.csv')
wine_review = wine_review.drop(columns = ['Unnamed: 0', 'winery'], axis =
1)
wine_review = wine_review.iloc[0:30000]
wine_review_encode = pd.get_dummies(wine_review,prefix=['country',
'province', 'variety'])
wine_data = wine_review_encode
wine_label = wine_review_encode["points"]
train_data, test_data, train_label, test_label = \
train_test_split(wine_data, wine_label)
clf = svm.SVC(gamma='auto')
clf.fit(train_data, train_label)
pre = clf.predict(test_data)
result = pd.DataFrame({"label": test_label, "prediction":pre})
ac_score = metrics.accuracy_score(test_label, pre)
print("\n정답률 =", ac_score)
```

```
: 1 wine_review = wine_review.iloc[0:30000]
2
3 wine_review_encode = pd.get_dummies(wine_review,prefix=['country', 'province',
4
5 wine_data = wine_review_encode
6 wine_label = wine_review_encode["points"]
7
8 train_data, test_data, train_label, test_label = ₩
9 train_test_split(wine_data, wine_label)
10
11 clf = svm.SVC(gamma='auto')
12 clf.fit(train_data, train_label)
13 pre = clf.predict(test_data)
14
15 result = pd.DataFrame({"label": test_label, "prediction":pre})
16 ac_score = metrics.accuracy_score(test_label, pre)
17 print("\n정답률 =", ac_score)
```

정답률 = 0.9617333333333333

7)사용하는 데이터 중 포도주 제작자를 제외하고 데이터 수는 50000개 테스트

```
import csv
import pandas as pd
from sklearn import svm, metrics
from sklearn.model_selection import train_test_split
wine_review = pd.read_csv('wine_review.csv')
wine_review = wine_review.drop(columns = ['Unnamed: 0', 'winery'], axis = 1)
wine_review = wine_review.iloc[0:30000]
wine_review_encode = pd.get_dummies(wine_review,prefix=['country', 'province', 'variety'])
wine_data = wine_review_encode
wine_label = wine_review_encode["points"]
train_data, test_data, train_label, test_label = \
train_test_split(wine_data, wine_label)
clf = svm.SVC(gamma='auto')
clf.fit(train_data, train_label)
pre = clf.predict(test_data)
result = pd.DataFrame({"label": test_label, "prediction":pre})
ac_score = metrics.accuracy_score(test_label, pre)
print("\n정답률 =", ac_score)
```

```
1 wine_review = pd.read_csv('wine_review.csv')
2
3 wine_review = wine_review.drop(columns = ['Unnamed: 0', 'winery'], axis = 1)
4 wine_review = wine_review.iloc[0:50000]
5
6 wine_review_encode = pd.get_dummies(wine_review,prefix=['country', 'province',
7
8 wine_data = wine_review_encode
9 wine_label = wine_review_encode["points"]
10
11 train_data, test_data, train_label, test_label = #
12 train_test_split(wine_data, wine_label)
13
14 clf = svm.SVC(gamma='auto')
15 clf.fit(train_data, train_label)
16 pre = clf.predict(test_data)
17
18 result = pd.DataFrame({"label": test_label, "prediction":pre})
19 ac_score = metrics.accuracy_score(test_label, pre)
20 print("#정답률 =", ac_score)
```

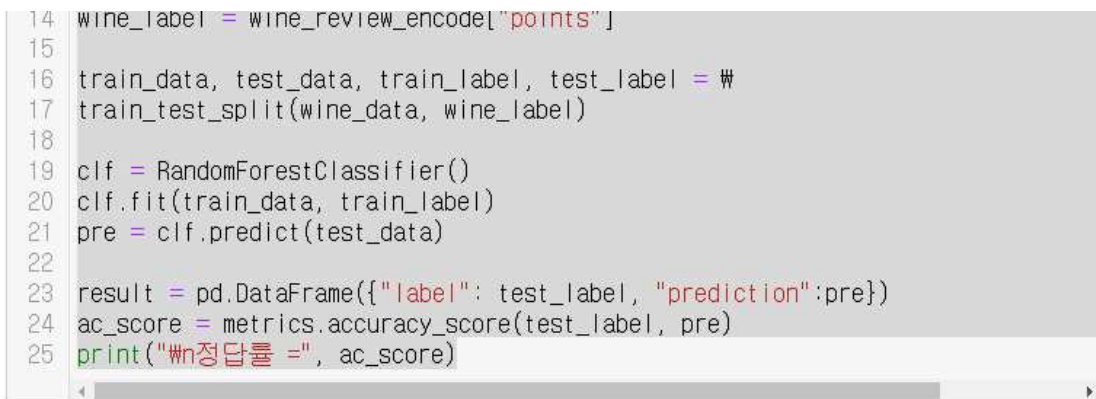
정답률 = 0.97584

3.2 랜덤포레스트

<코드>

```
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn import metrics
from sklearn.model_selection import train_test_split

wine_review = pd.read_csv('wine_review.csv')
wine_review = wine_review.drop(columns = ['Unnamed: 0', 'winery'], axis =
1)
wine_review = wine_review.iloc[0:50000]
wine_review_encode = pd.get_dummies(wine_review,prefix=['country',
'province', 'variety'])
wine_data = wine_review_encode
wine_label = wine_review_encode["points"]
train_data, test_data, train_label, test_label = \
train_test_split(wine_data, wine_label)
clf = RandomForestClassifier()
clf.fit(train_data, train_label)
pre = clf.predict(test_data)
result = pd.DataFrame({"label": test_label, "prediction":pre})
ac_score = metrics.accuracy_score(test_label, pre)
print("\n정답률 =", ac_score)
```



```
14 wine_label = wine_review_encode["points"]
15
16 train_data, test_data, train_label, test_label = #
17 train_test_split(wine_data, wine_label)
18
19 clf = RandomForestClassifier()
20 clf.fit(train_data, train_label)
21 pre = clf.predict(test_data)
22
23 result = pd.DataFrame({"label": test_label, "prediction":pre})
24 ac_score = metrics.accuracy_score(test_label, pre)
25 print("\n정답률 =", ac_score)
```

정답률 = 0.95272

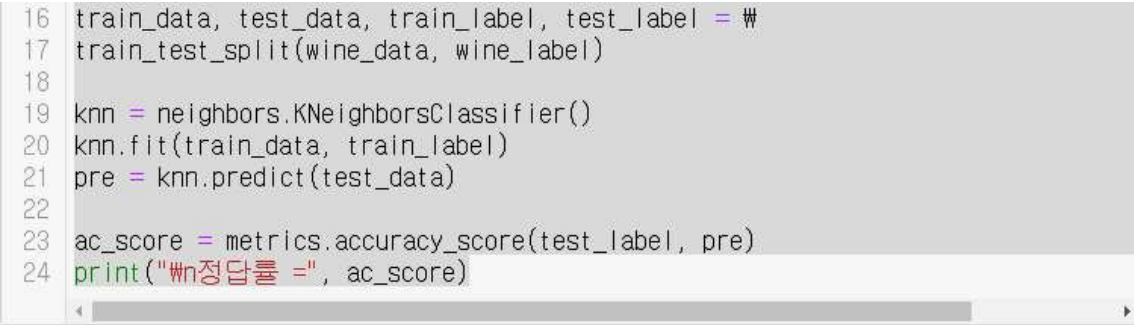
3.3 KNN

1) 데이터 50000개

<코드>

```
import pandas as pd
from sklearn import neighbors
from sklearn import metrics
from sklearn.model_selection import train_test_split

wine_review = pd.read_csv('wine_review.csv')
wine_review = wine_review.drop(columns = ['Unnamed: 0', 'winery'], axis =
1)
wine_review = wine_review.iloc[0:50000]
wine_review_encode = pd.get_dummies(wine_review, prefix=['country',
'province', 'variety'])
wine_data = wine_review_encode
wine_label = wine_review_encode["points"]
train_data, test_data, train_label, test_label = \
train_test_split(wine_data, wine_label)
knn = neighbors.KNeighborsClassifier()
knn.fit(train_data, train_label)
pre = knn.predict(test_data)
ac_score = metrics.accuracy_score(test_label, pre)
print("\n정답률 =", ac_score)
```



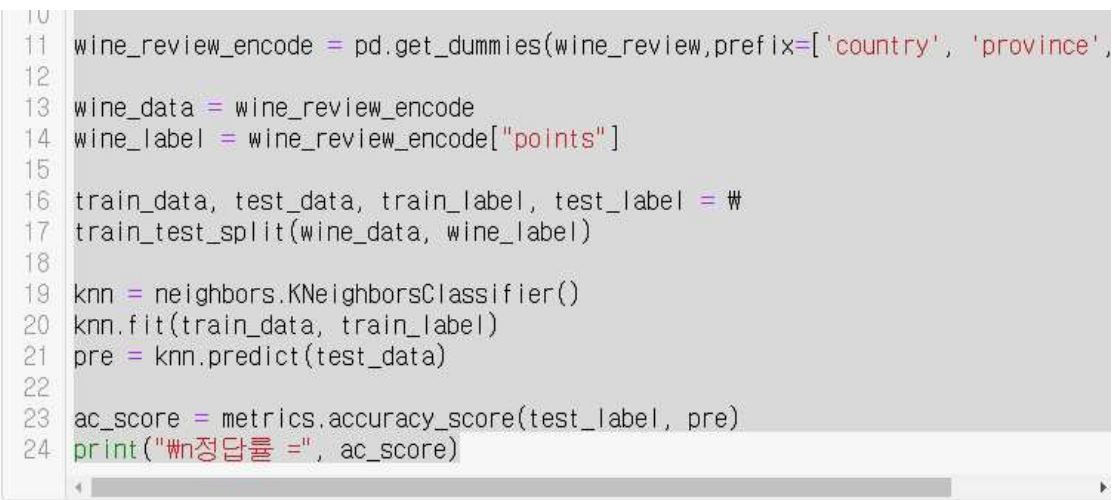
```
16 train_data, test_data, train_label, test_label = \
17 train_test_split(wine_data, wine_label)
18
19 knn = neighbors.KNeighborsClassifier()
20 knn.fit(train_data, train_label)
21 pre = knn.predict(test_data)
22
23 ac_score = metrics.accuracy_score(test_label, pre)
24 print("\n정답률 =", ac_score)
```

정답률 = 0.9356

2) 데이터 70000개

```
import pandas as pd
from sklearn import neighbors
from sklearn import metrics
from sklearn.model_selection import train_test_split

wine_review = pd.read_csv('wine_review.csv')
wine_review = wine_review.drop(columns = ['Unnamed: 0', 'winery'], axis =
1)
wine_review = wine_review.iloc[0:70000]
wine_review_encode = pd.get_dummies(wine_review, prefix=['country',
'province', 'variety'])
wine_data = wine_review_encode
wine_label = wine_review_encode["points"]
train_data, test_data, train_label, test_label = \
train_test_split(wine_data, wine_label)
knn = neighbors.KNeighborsClassifier()
knn.fit(train_data, train_label)
pre = knn.predict(test_data)
ac_score = metrics.accuracy_score(test_label, pre)
print("\n정답률 =", ac_score)
```



```
10
11 wine_review_encode = pd.get_dummies(wine_review, prefix=['country', 'province',
12
13 wine_data = wine_review_encode
14 wine_label = wine_review_encode["points"]
15
16 train_data, test_data, train_label, test_label = #
17 train_test_split(wine_data, wine_label)
18
19 knn = neighbors.KNeighborsClassifier()
20 knn.fit(train_data, train_label)
21 pre = knn.predict(test_data)
22
23 ac_score = metrics.accuracy_score(test_label, pre)
24 print("\n정답률 =", ac_score)
```

정답률 = 0.9429714285714286

3.4 나이브베이지

1) 데이터 50000개

<코드>

```
import csv
import pandas as pd
from sklearn import metrics
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split

wine_review = pd.read_csv('wine_review.csv')
wine_review = wine_review.drop(columns = ['Unnamed: 0', 'winery'], axis =
1)
wine_review = wine_review.iloc[0:50000]
wine_review_encode = pd.get_dummies(wine_review,prefix=['country',
'province', 'variety'])
wine_data = wine_review_encode
wine_label = wine_review_encode["points"]
train_data, test_data, train_label, test_label = \
train_test_split(wine_data, wine_label)
nb = GaussianNB()
nb.fit(train_data, train_label)
pre = nb.predict(test_data)
ac_score = metrics.accuracy_score(test_label, pre)
print("\n정답률 =", ac_score)
```

```
12 wine_review_encode = pd.get_dummies(wine_review,prefix=['country', 'province',
13
14 wine_data = wine_review_encode
15 wine_label = wine_review_encode["points"]
16
17 train_data, test_data, train_label, test_label = \
18 train_test_split(wine_data, wine_label)
19
20 nb = GaussianNB()
21 nb.fit(train_data, train_label)
22 pre = nb.predict(test_data)
23
24 ac_score = metrics.accuracy_score(test_label, pre)
25 print("\n정답률 =", ac_score)
```

정답률 = 0.99504

2)데이터 60000개

<코드>

```
import csv
import pandas as pd
from sklearn import metrics
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split

wine_review = pd.read_csv('wine_review.csv')
wine_review = wine_review.drop(columns = ['Unnamed: 0', 'winery'], axis =
1)
wine_review = wine_review.iloc[0:60000]
wine_review_encode = pd.get_dummies(wine_review,prefix=['country',
'province', 'variety'])
wine_data = wine_review_encode
wine_label = wine_review_encode["points"]
train_data, test_data, train_label, test_label = \
train_test_split(wine_data, wine_label)
nb = GaussianNB()
nb.fit(train_data, train_label)
pre = nb.predict(test_data)
ac_score = metrics.accuracy_score(test_label, pre)
print("\n정답률 =", ac_score)
```

```
17 train_data, test_data, train_label, test_label = \
18 train_test_split(wine_data, wine_label)
19
20 nb = GaussianNB()
21 nb.fit(train_data, train_label)
22 pre = nb.predict(test_data)
23
24 ac_score = metrics.accuracy_score(test_label, pre)
25 print("\n정답률 =", ac_score)
```

정답률 = 0.9956

3) 모델 테스트

<코드>

```
import csv
import pandas as pd
from sklearn import metrics
from sklearn.naive_bayes import GaussianNB
from sklearn import model_selection
import random, re

wine_review = pd.read_csv('wine_review.csv')
wine_review = wine_review.drop(columns = ['Unnamed: 0', 'winery'], axis = 1)
wine_review = wine_review.iloc[0:50000]
wine_review_encode = pd.get_dummies(wine_review,prefix=['country', 'province', 'variety'])
wine_data = wine_review_encode
wine_label = wine_review_encode["points"]
nb = GaussianNB()
scores = model_selection.cross_val_score(nb, wine_data, wine_label, cv = 5)
print("각각의 정답률 = ", scores)
print("평균 정답률 = ", scores.mean())
```

```
8 wine_review = pd.read_csv('wine_review.csv')
9
10 wine_review = wine_review.drop(columns = ['Unnamed: 0', 'winery'], axis = 1)
11 wine_review = wine_review.iloc[0:50000]
12
13 wine_review_encode = pd.get_dummies(wine_review,prefix=['country', 'province',
14
15 wine_data = wine_review_encode
16 wine_label = wine_review_encode["points"]
17
18 nb = GaussianNB()
19 scores = model_selection.cross_val_score(nb, wine_data, wine_label, cv = 5)
20
21 print("각각의 정답률 = ", scores)
22 print("평균 정답률 = ", scores.mean())
```

```
각각의 정답률 = [0.9954 0.9934 0.9943 0.994 0.9953]
평균 정답률 = 0.99448
```


4. 분석 결과의 시사점 및 결론

정확도 예측은 데이터의 수, 데이터의 종류, 분석기법에 영향을 받는다는 것을 알 수 있습니다. 첫 번째 데이터의 수 경우에는 많을수록 정확도가 올라갔지만 50000개 이상에서는 크게 차이가 나지 않는다는 것을 알 수 있습니다. 두 번째 데이터의 종류 같은 경우에는 포도주를 생산한 국가가 점수예측에 큰 영향을 미치고 포도주 생산자의 경우에는 영향을 거의 없다는 것을 알 수 있었습니다. 세 번째 분석기법의 경우에는 나이브베이즈 기법이 평균 정확도가 99.4%가 나올 정도로 가장 높은 정확도를 가진다는 것을 알 수 있었습니다.

그러므로 높은 점수의 포도주를 고르기 위해서는 포도주를 생산한 국가, 포도주를 만든 지방, 포도주의 등급을 아는 것이 중요합니다. 또한, 이 정보들과 나이브베이즈 기법을 활용한다면 거의 100%의 확률로 높은 점수의 포도주를 고를 수 있습니다.