빅데이터 분석 시각화

Homework#13

학번 : 2016039029
이름 : 이종영
마감일자 : 23/12/04

Contents

1. Data Analysis Processing

1.1 데이터 분석에 사용될 라이브러리 및 데이터를 호출

```
# 1. Import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sb

from matplotlib import style
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
from sklearn.ensemble import AdaBoostRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
```

✓ 0.0s

```
# 2. Import the dataset (same directory - py)
df = pd.read_csv('winequality-red.csv')
```

✓ 0.0s

1.2 데이터 정보 확인 크기, 데이터 형태, 분포

```
# 3. Exploratory data analysis: Data Information
print(df.shape)
print(df.info())
print(df.describe())
```

✓ 0.0s

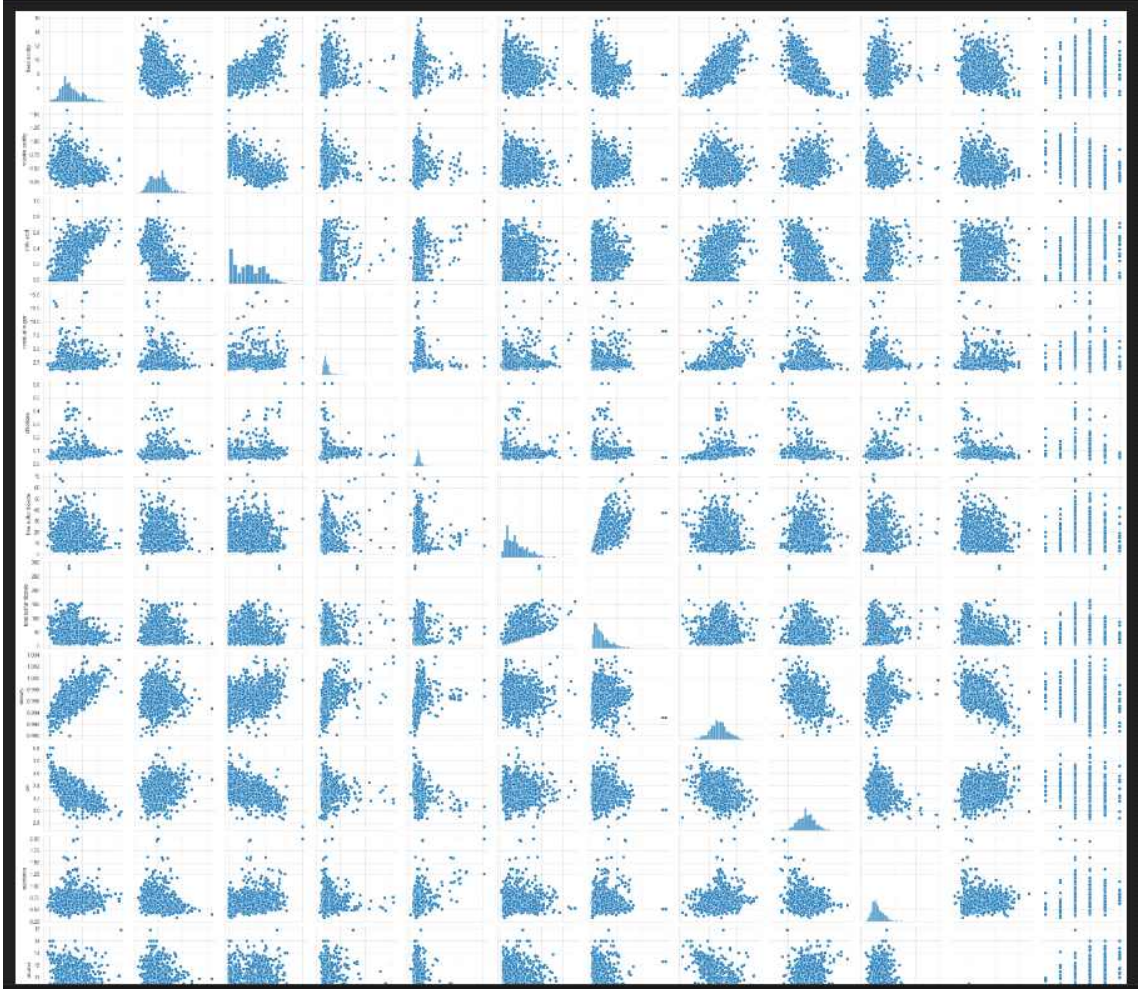
```
(1599, 12)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1599 entries, 0 to 1598
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   fixed acidity         1599 non-null   float64
 1   volatile acidity      1599 non-null   float64
 2   citric acid           1599 non-null   float64
 3   residual sugar        1599 non-null   float64
 4   chlorides             1599 non-null   float64
 5   free sulfur dioxide   1599 non-null   float64
 6   total sulfur dioxide  1599 non-null   float64
 7   density               1599 non-null   float64
 8   pH                    1599 non-null   float64
 9   sulphates             1599 non-null   float64
 10  alcohol               1599 non-null   float64
 11  quality               1599 non-null   int64
dtypes: float64(11), int64(1)
memory usage: 150.0 KB
None
       fixed acidity  volatile acidity  citric acid  residual sugar  \
count    1599.000000       1599.000000  1599.000000     1599.000000
mean        8.319637          0.527821     0.270976        2.538806
std         1.741096          0.179060     0.194801        1.409928
...
25%         3.210000          0.550000     9.500000        5.000000
50%         3.310000          0.620000    10.200000        6.000000
75%         3.400000          0.730000    11.100000        6.000000
max         4.010000          2.000000    14.900000        8.000000
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

1.3 EDA 수행 데이터 분포 관계 시각화

```
# 3. Exploratory data analysis: Data Visualization
sb.set_style('whitegrid')
plt.rcParams['figure.figsize'] = (20,10)

sb.pairplot(df)
plt.savefig('pairplor.png')
```

✓ 31.4s



선형 관계에 가까운 속성을 선정 한다.

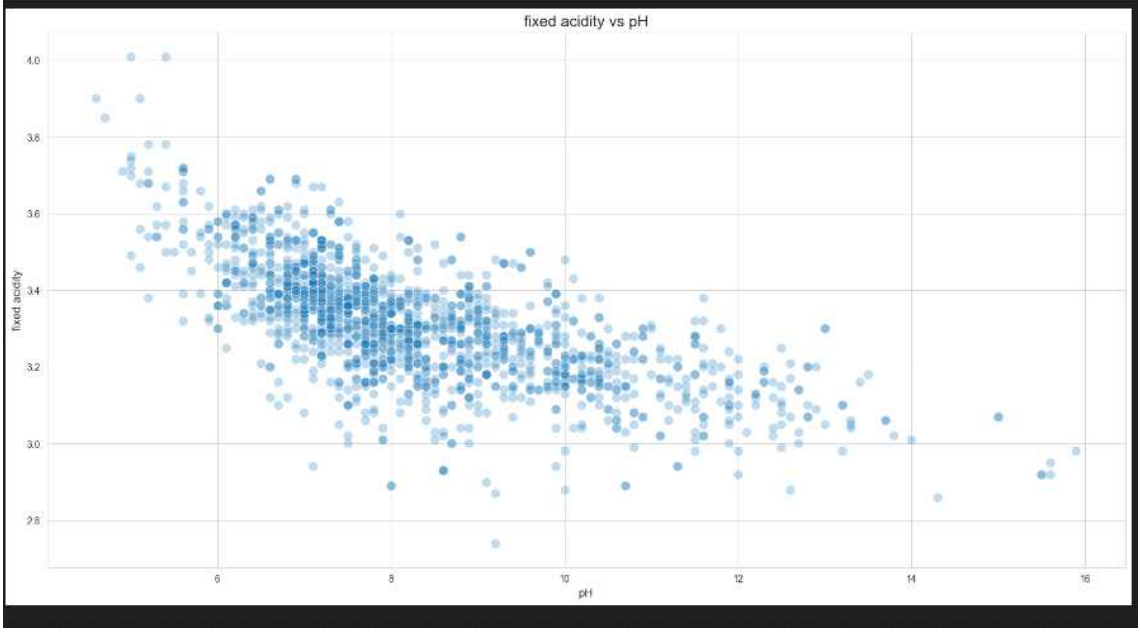
1.4 선형관계 형태의 분포 그래프를 가진 fixed acidity와 pH의 분포를 확인

```
# 3. Exploratory data analysis: Find linear relationships
plt.scatter(x = 'fixed acidity', y = 'pH',
        data = df, s = 100, alpha = 0.3, edgecolor = 'white')

plt.title('fixed acidity vs pH', fontsize = 16)
plt.ylabel('fixed acidity', fontsize = 12)
plt.xlabel('pH', fontsize = 12)

plt.savefig('fixed acidity2.png')
```

✓ 0.5s



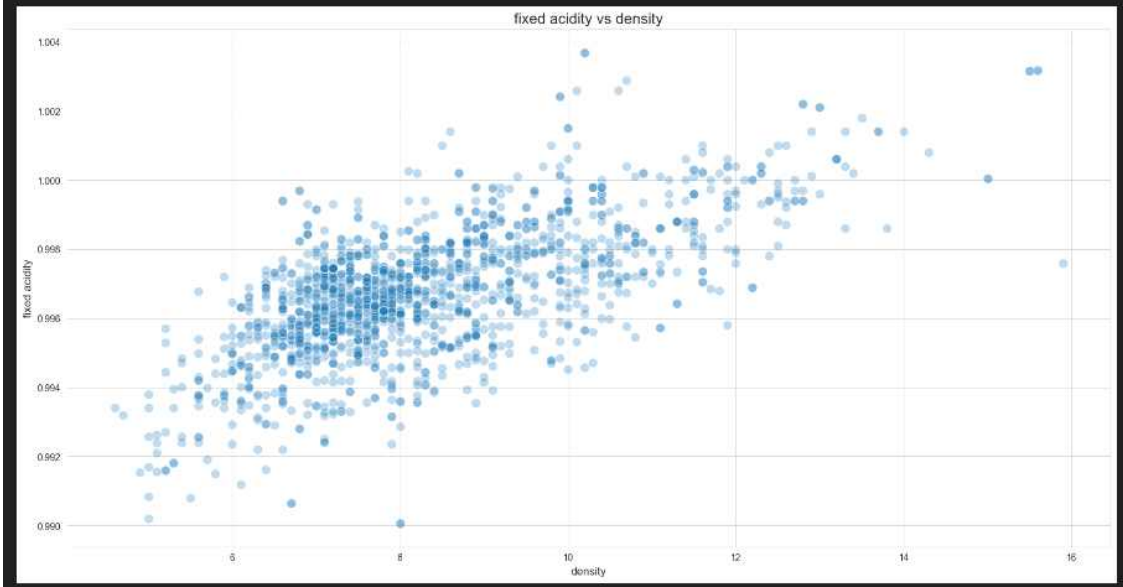
1.4 선형관계 형태의 fixedacidity와 density의 분포를 확인

```
# 3. Exploratory data analysis: Find linear relationships
plt.scatter(x = 'fixed acidity', y = 'density',
        data = df, s = 100, alpha = 0.3, edgecolor = 'white')

plt.title('fixed acidity vs density', fontsize = 16)
plt.ylabel('fixed acidity', fontsize = 12)
plt.xlabel('density', fontsize = 12)

plt.savefig('fixed acidity3.png')
```

✓ 0.4s



1.5 싱글LR 진행 독립속성의 pH에 대해 종속속성의 fixed acidity를 학습 테스트셋은 3:7로 정함.

```
# 4. SLR : id-value (pH), d-value (fixed acidity)
X_var = df[['pH']]
y_var = df['fixed acidity']

X_train, X_test, y_train, y_test = train_test_split(
    X_var, y_var, test_size = 0.3, random_state = 0)
```

✓ 0.0s

1.6 모델 생성

```
# 5. SLR : Training model
lr = LinearRegression()
lr.fit(X_train, y_train)
yhat = lr.predict(X_test)
```

✓ 0.0s

1.7.1 RMSE 확인

```
# 5. SLR : Check Accuracy Score - RMSE
MSE = mean_squared_error(y_test, yhat)
np.sqrt(MSE)
```

✓ 0.0s

```
1.2216744330592366
```

1.7.2 R-squared 확인

```
# 6. SLR : Check Accuracy Score - R squared
print(r2_score(y_test, yhat))
```

✓ 0.0s

```
0.5059879527342228
```

별로 좋지 못한 성능을 확인할 수 있다.

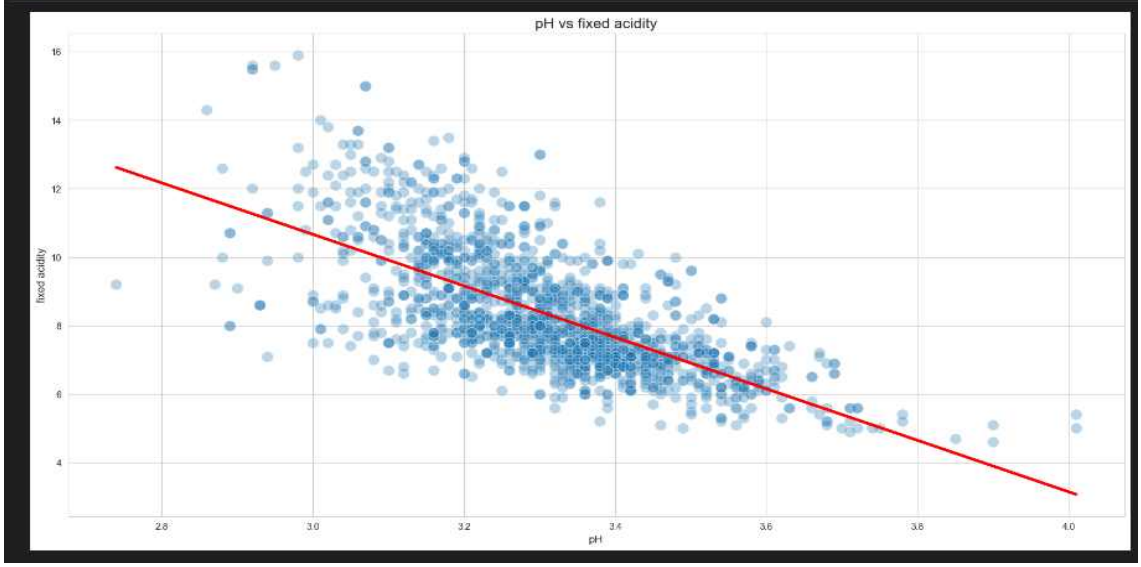
1.7.3 분포 및 정확도에 대한 시각화를 진행한다.

```
# 7. SLR : Check Accuracy Score Visualization
slr_slope = lr.coef_
slr_intercept = lr.intercept_

sb.scatterplot(x = 'pH', y = 'fixed acidity',
        data = df, s = 150, alpha = 0.3, edgecolor = 'white')
plt.plot(df['pH'], slr_slope*df['pH'] + slr_intercept,
        color = 'r', linewidth = 3)
plt.title('pH vs fixed acidity', fontsize = 16)
plt.ylabel('fixed acidity', fontsize = 12)
plt.xlabel('pH', fontsize = 12)

plt.savefig('enginesize_co2_fit.png')
```

✓ 0.4s



1.8 멀티LR 진행 density,ph,cirtric acid에 대해 종속속성의 fixed acidity를 학습 테스트셋은 3:7로 정함.

```
# 7. MLR : id-value (pH,density,citric acid), d-value (fixed acidity)
X1_var = df[['pH',
             'density',
             'citric acid']]
y_var = df['fixed acidity']

X_train, X_test, y_train, y_test = train_test_split(
             X1_var,
             y_var,
             test_size = 0.3,
             random_state = 0)
```

✓ 0.0s

1.9 모델 생성

```
# 7. MLR : Training model
lr = LinearRegression()
lr.fit(X_train, y_train)
yhat = lr.predict(X_test)
```

✓ 0.0s

1.10.1 RMSE 확인

```
# 7. MLR : Check Accuracy Score - RMSE
MSE = mean_squared_error(y_test, yhat)
np.sqrt(MSE)
```

✓ 0.0s

```
0.8340057828574466
```

1.10.2 R-squared 확인

```
# 7. MLR : Check Accuracy Score - R squared
print(r2_score(y_test, yhat))
```

✓ 0.0s

```
0.7697685261644683
```

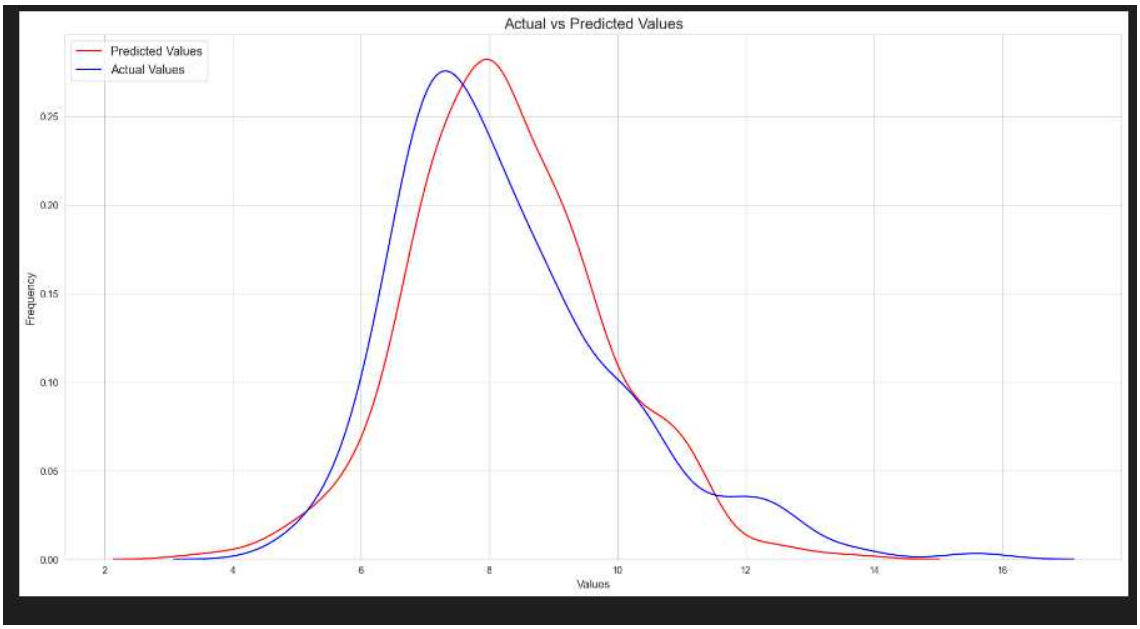
성능이 개선됐음을 확인할 수 있다.

1.10.3 분포 및 정확도에 대한 시각화를 진행한다.

```
# 7. MLR : Check Accuracy Score Visualization
sb.distplot(yhat, hist = False, color = 'r', label = 'Predicted Values')
sb.distplot(y_test, hist = False, color = 'b', label = 'Actual Values')
plt.title('Actual vs Predicted Values', fontsize = 16)
plt.xlabel('Values', fontsize = 12)
plt.ylabel('Frequency', fontsize = 12)
plt.legend(loc = 'upper left', fontsize = 13)

plt.savefig('ap.png')
```

✓ 0.4s



1.11 AdaBoost 이용하여 성능측정 모델 생성

```
# 8. Ada : Training model
ada = AdaBoostRegressor(n_estimators = 10)
ada.fit(X_train, y_train)
yhat = ada.predict(X_test)
```

✓ 0.0s

트리의 수는 10개로 정하였음.

1.12.1 RMSE 확인

```
# 8. Ada : Check Accuracy Score - RMSE
MSE = mean_squared_error(y_test, yhat)
np.sqrt(MSE)
```

✓ 0.0s

```
0.8407859748268565
```

1.12.2 R-squared 확인

```
# 8. Ada : Check Accuracy Score - R squared
print(r2_score(y_test, yhat))
```

✓ 0.0s

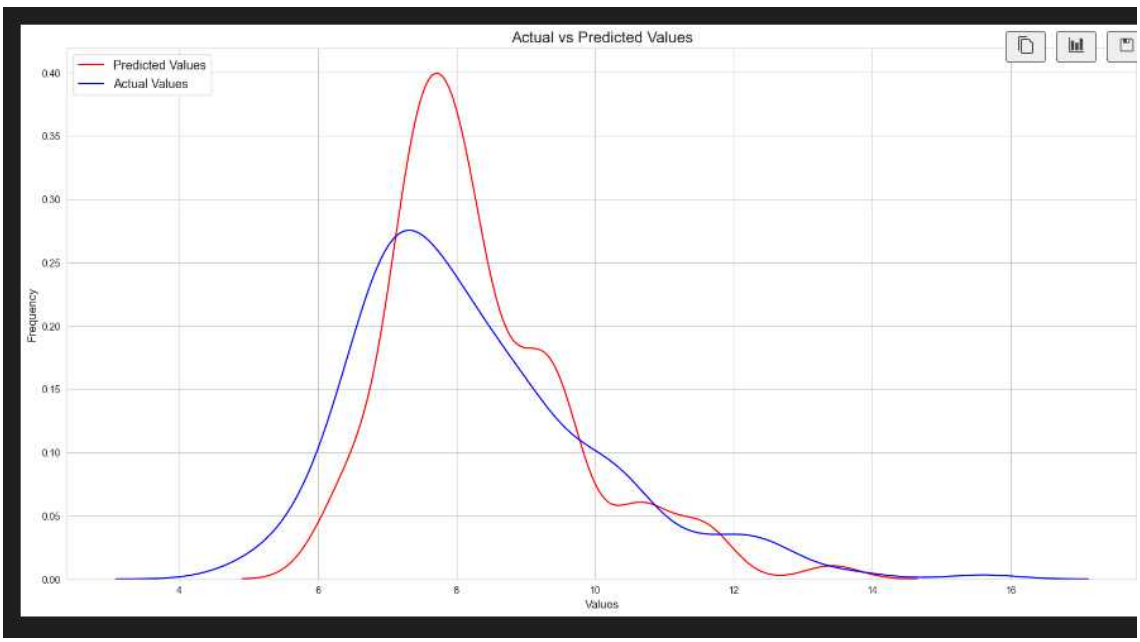
```
0.7660098979029796
```

1.12.3 분포 및 정확도에 대한 시각화를 진행한다.

```
# 8. Ada : Check Accuracy Score Visualization
sb.distplot(yhat, hist = False, color = 'r', label = 'Predicted Values')
sb.distplot(y_test, hist = False, color = 'b', label = 'Actual Values')
plt.title('Actual vs Predicted Values', fontsize = 16)
plt.xlabel('Values', fontsize = 12)
plt.ylabel('Frequency', fontsize = 12)
plt.legend(loc = 'upper left', fontsize = 13)

plt.savefig('ap.png')
```

✓ 0.5s



1.13 RandomForest 이용하여 성능측정 모델 생성

```
# 9. RandomForest : Training model
rf = RandomForestRegressor(n_estimators = 10)
rf.fit(X_train, y_train)
yhat = rf.predict(X_test)
```

✓ 0.0s

트리의 수는 10개로 정하였음.

1.14.1 RMSE 확인

```
# 9. RandomForest : Check Accuracy Score - RMSE
MSE = mean_squared_error(y_test, yhat)
np.sqrt(MSE)
```

✓ 0.0s

```
0.795150944535003
```

1.14.2 R-squared 확인

```
# 9. RandomForest : Check Accuracy Score - R squared
print(r2_score(y_test, yhat))
```

✓ 0.0s

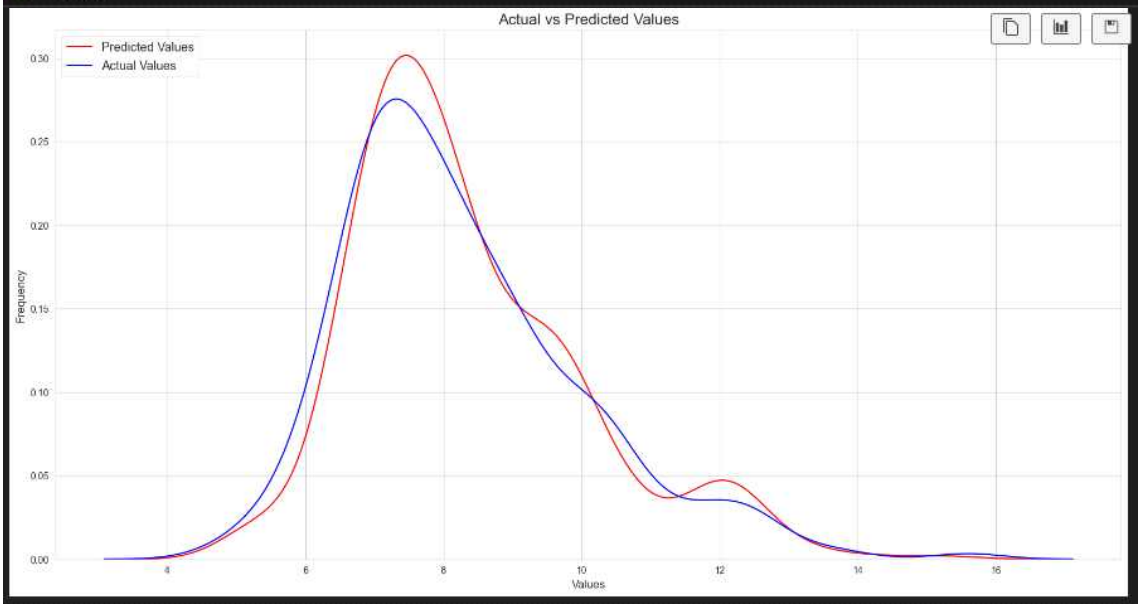
```
0.7907209630926777
```

1.14.3 분포 및 정확도에 대한 시각화를 진행한다.

```
# 9. RandomForest : Check Accuracy Score Visualization
sb.distplot(yhat, hist = False, color = 'r', label = 'Predicted Values')
sb.distplot(y_test, hist = False, color = 'b', label = 'Actual Values')
plt.title('Actual vs Predicted Values', fontsize = 16)
plt.xlabel('Values', fontsize = 12)
plt.ylabel('Frequency', fontsize = 12)
plt.legend(loc = 'upper left', fontsize = 13)

plt.savefig('ap.png')
```

✓ 0.4s



1.15 성능비교

진행한 성능측정 모델의 R-squared 결과는 다음과 같다.

SLR - 0.5059879527342228

MLR - 0.7697685261644683

MLR(Ada) - 0.7660098979029796

MLR(RandomForest) - 0.7907209630926777

SLR < MLR(Ada) < MLR < MLR(RandomForest)

2. Result

데이터 분석 후 분포에 대한 성능 측정 모델을 구축하였습니다.

측정에 사용하는 알고리즘 마다의 RMSE 및 R-squared를 측정하여 성능을 비교하였습니다.

선택된 속성값에 대해 트레이닝셋과 테스트 셋의 비율은 7:2으로 선정하였습니다.

먼저 속성에 대한 종속,독립 속성의 관계로 보이는 선형분포 관계를 가진 속성을 선정하였고 SLR,MLR 및 MLR에서 모델 생성에 대한 각 알고리즘을 이용하여 성능측정을 진행하였습니다.

R-squared 가 1에 가까운 가장 성능이 좋은 모델은 MLR(RandomForest)을 이용하였을 때 도출해낼 수 있었습니다.