빅데이터 분석 시각화

Homework#11

학번 : 2016039029
이름 : 이종영
마감일자 : 23/11/27

Contents

1. Data Analysis Processing
2. Result

1. Data Analysis Processing

1.1 데이터 분석에 사용될 라이브러리 및 데이터를 호출

```python
# 1. Import necessary libraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix
from sklearn import metrics
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.metrics import accuracy_score

# 2. Import the dataset (same directory)
df = pd.read_csv('melb_data.csv')
```

1.2 데이터 정보 확인 헤더정보, 크기, 데이터 형태

```python
# 3. Data Visualization
print(df.head())
print(df.shape)
print(df.info())

for col in df.columns:
    print(df[col].value_counts()/np.float64(len(df)))
```

✓ 0.0s

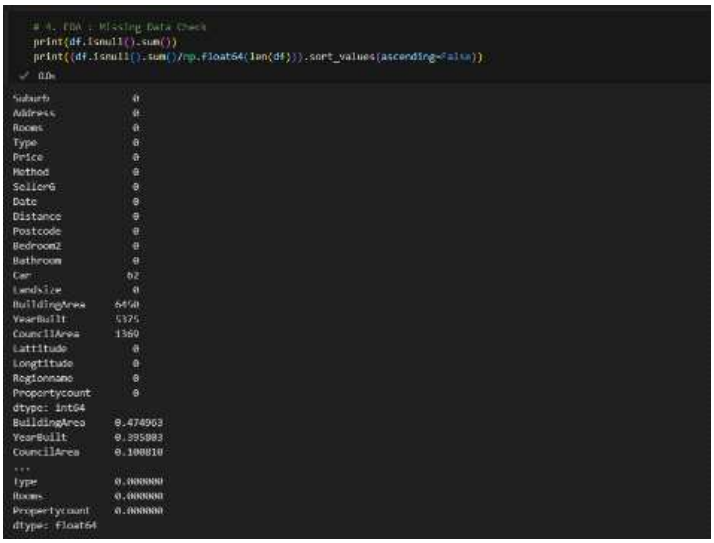
```
       Suburb           Address  Rooms Type      Price Method SellerG  \
0  Abbotsford      85 Turner St      2    h  1480000.0      S  Biggin
1  Abbotsford   25 Bloomburg St      2    h  1035000.0      S  Biggin
2  Abbotsford      5 Charles St      3    h  1465000.0     SP  Biggin
3  Abbotsford  40 Federation La      3    h   850000.0     PI  Biggin
4  Abbotsford       55a Park St      4    h  1600000.0     VB  Nelson

        Date  Distance  Postcode  ...  Bathroom  Car  Landsize  BuildingArea  \
0  3/12/2016       2.5    3067.0  ...       1.0  1.0     202.0           NaN
1  4/02/2016       2.5    3067.0  ...       1.0  0.0     156.0          79.0
2  4/03/2017       2.5    3067.0  ...       2.0  1.0     134.0         150.0
3  4/03/2017       2.5    3067.0  ...       2.0  1.0      94.0           NaN
4  4/06/2016       2.5    3067.0  ...       1.0  2.0     120.0         142.0

   YearBuilt  CouncilArea Lattitude  Longtitude             Regionname  \
0        NaN       Yarra   -37.7996    144.9984  Northern Metropolitan
1     1900.0       Yarra   -37.8079    144.9934  Northern Metropolitan
2     1900.0       Yarra   -37.8093    144.9944  Northern Metropolitan
3        NaN       Yarra   -37.7969    144.9969  Northern Metropolitan
4     2014.0       Yarra   -37.8072    144.9941  Northern Metropolitan

   Propertycount
0         4019.0
1         4019.0
2         4019.0
...
2891.0    0.000074
1863.0    0.000074
1624.0    0.000074
Name: count, Length: 311, dtype: float64
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

1.3 EDA 수행 종속변수 관계 확인 속성 선정

```
# 4. EDA : target attribute - Type
# h - house,cottage,villa, semi,terrace; u - unit, duplex; t - townhouse;
print(df['Type'].value_counts())
print()
print(df['Type'].value_counts()/np.float64(len(df)))
```

1.4 누락된 필드 체크



```
# 4. EDA : Missing Data Check
print(df.isnull().sum())
print((df.isnull().sum()/np.float64(len(df))).sort_values(ascending=False))
```

```
Suburb           0
Address          0
Rooms            0
Type             0
Price            0
Method           0
SellerG          0
Date             0
Distance         0
Postcode         0
Bedroom2         0
Bathroom         0
Car             62
Landsize         0
BuildingArea    6450
YearBuilt       5375
CouncilArea     1369
Lattitude        0
Longtitude       0
Regionname       0
Propertycount    0
dtype: int64
BuildingArea    0.474963
YearBuilt       0.395883
CouncilArea     0.100810
...
Type            0.000000
Rooms           0.000000
Propertycount   0.000000
dtype: float64
```

'BuildingArea','YearBuilt','CouncilArea''Car'

4가지 속성에 대해 전처리 필요

이때, 10% 초과인 3개 속성에 대해서는 CDA 진행 그 외

속성에 대해서는 결측값에 대해 대체값을 삽입한다.

각 튜플의 값이 다수건이 확인 되므로 단순대체를 이용한다.

```
# 4. EDA : CDA 10% OVER
df = df.drop(columns=['BuildingArea','YearBuilt','CouncilArea'])

# 4. EDA : SimpleImputer 10% UNDER
imputer = SimpleImputer(strategy='mean')
Car = df['Car'].values.reshape(-1, 1)
imputed_Car = imputer.fit_transform(Car)
df['Car'] = imputed_Car

print(df.isnull().sum())
```

```
Suburb          0
Address         0
Rooms           0
Type            0
Price           0
Method          0
SellerG         0
Date            0
Distance        0
Postcode        0
Bedroom2        0
Bathroom        0
Car             0
Landsize        0
Lattitude       0
Longtitude      0
Regionname      0
Propertycount   0
dtype: int64
```

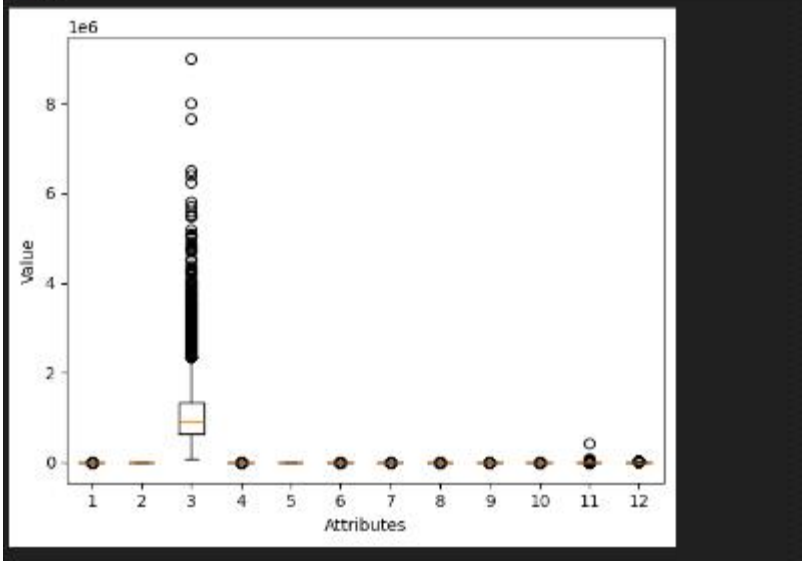
1.5 이상값 확인

이전에 확인한 데이터 중 범주형 데이터에 대한 작업을 진행하는데 방해되는 속성을 제거하며 그 외에는 라벨인코딩으로 동일한 데이터형으로 바꾸어 진행한다

```
# 4. EDA : Not Categorical data
df = df.drop(columns=['Suburb','Address','Date','Lattitude','Longtitude','Regionname'])


# 4. EDA : Categorical data Labelencoding
encoder = LabelEncoder()
df['Method'] = encoder.fit_transform(np.array(df['Method'].values.reshape(-1,1)))
df['SellerG'] = encoder.fit_transform(np.array(df['SellerG'].values.reshape(-1,1)))
df['Type'] = encoder.fit_transform(np.array(df['Type'].values.reshape(-1,1)))
```

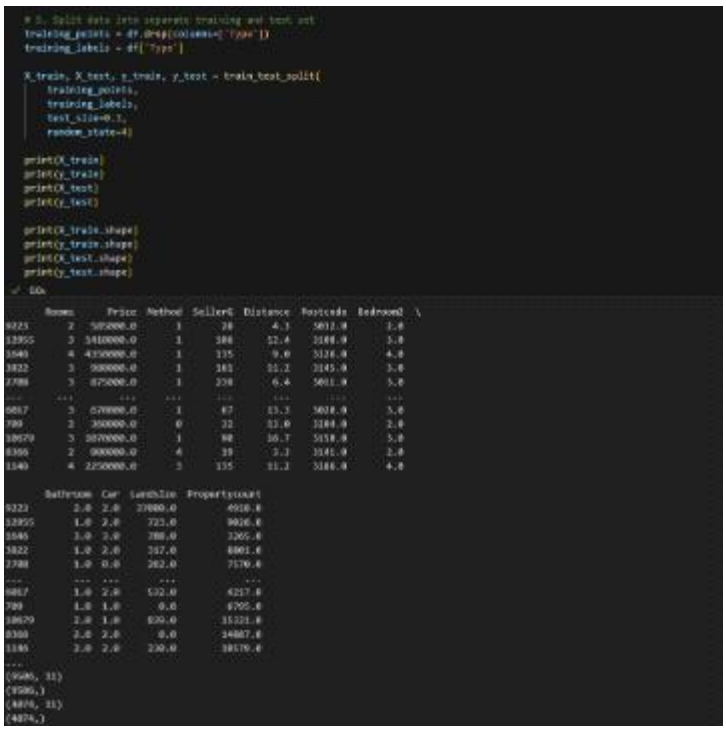
이후 이상값을 다시 확인한다.

```
# 4. EDA: Outliers
plt.boxplot(df)
plt.xlabel('Attributes')
plt.ylabel('Value')
plt.show()
✓ 0.0s
```

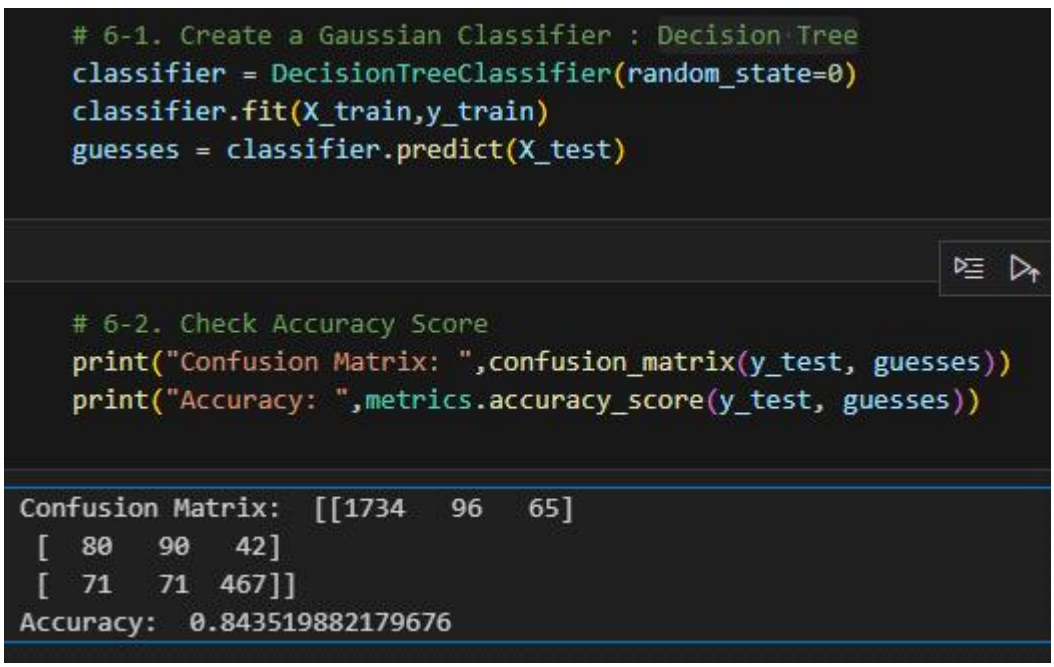


다음의 작업에서 3번 컬럼에 대해 조치 되는지 확인하기로 한 다음 다음 절차를 진행한다.

1.6 트레이닝셋을 8의 비율로 세트 및 확인 시드값은 5



1.7 Gaussian Classifier-Decision Tree 정확도 확인



정확도 : 0.843519882179676...

1.7.1 Gaussian Classifier-Random Forest 정확도 확인

```
# 6-3 Create a Gaussian Classifier : Random Forest
classifier = RandomForestClassifier(random_state=0, n_estimators=3)
classifier.fit(X_train,y_train)
guesses = classifier.predict(X_test)
```

```
# 6-4. Check Accuracy Score
print("Confusion Matrix: ",confusion_matrix(y_test, guesses))
print("Accuracy: ",metrics.accuracy_score(y_test, guesses))
```

```
Confusion Matrix:  [[1789   50   56]
 [  99   79   34]
 [  84   48  477]]
Accuracy:  0.8634020618556701
```

정확도 : 0.8634020618556701...

1.7.2 Gaussian Classifier-Random Forest 트리 수 변경 n=10

```
# 6-5. Create a Gaussian Classifier : Random Forest increasing trees
classifier = RandomForestClassifier(random_state=0, n_estimators=10)
classifier.fit(X_train,y_train)
guesses = classifier.predict(X_test)
```

```
# 6-6. Check Accuracy Score
print("Confusion Matrix: ",confusion_matrix(y_test, guesses))
print("Accuracy: ",metrics.accuracy_score(y_test, guesses))
```

```
Confusion Matrix:  [[1805   43   47]
 [  90   88   34]
 [  74   44  491]]
Accuracy:  0.8777614138438881
```

정확도 : 0.8777614138438881...

1.7.3 Gaussian Classifier-Random Forest 트리 수 변경 n=100

```
# 6-5. Create a Gaussian Classifier : Random Forest increasing trees
classifier = RandomForestClassifier(random_state=0, n_estimators=100)
classifier.fit(X_train,y_train)
guesses = classifier.predict(X_test)
```

```
# 6-6. Check Accuracy Score
print("Confusion Matrix: ",confusion_matrix(y_test, guesses))
print("Accuracy: ",metrics.accuracy_score(y_test, guesses))
```

```
Confusion Matrix:  [[1821   34   40]
 [  82   91   39]
 [  61   43  505]]
Accuracy:  0.8899116347569955
```

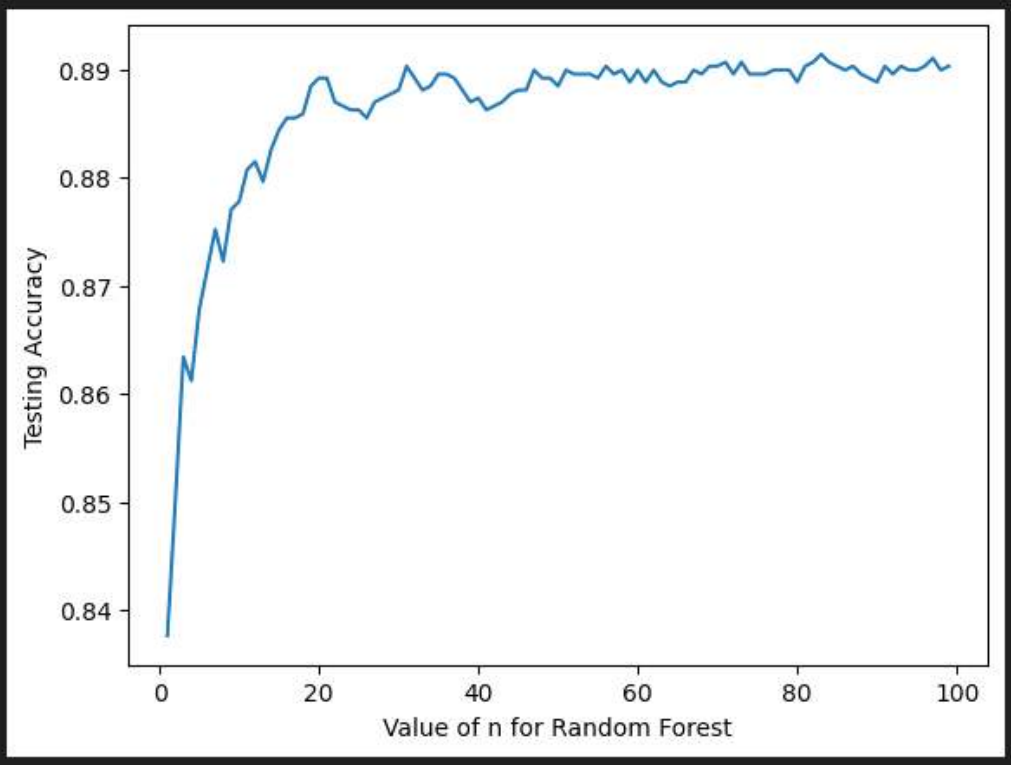
정확도 : 0.8899116347569955...

1.7.4 Gaussian Classifier-Random Forest 트리수 변경에 대한 시각화

```
# 7-1. Check Accuracy Score that Gaussian Naive Bayes : Numbers 1 to 100 Trees
n_range = range(1, 100)
accuracy_scores = []

for n in n_range:
    classifier = RandomForestClassifier(random_state=0, n_estimators=n)
    classifier.fit(X_train, y_train)
    guesses = classifier.predict(X_test)
    accuracy_scores.append(accuracy_score(y_test, guesses))
print(accuracy_scores)

plt.plot(n_range, accuracy_scores)
plt.xlabel('Value of n for Random Forest')
plt.ylabel('Testing Accuracy')
plt.show()
```

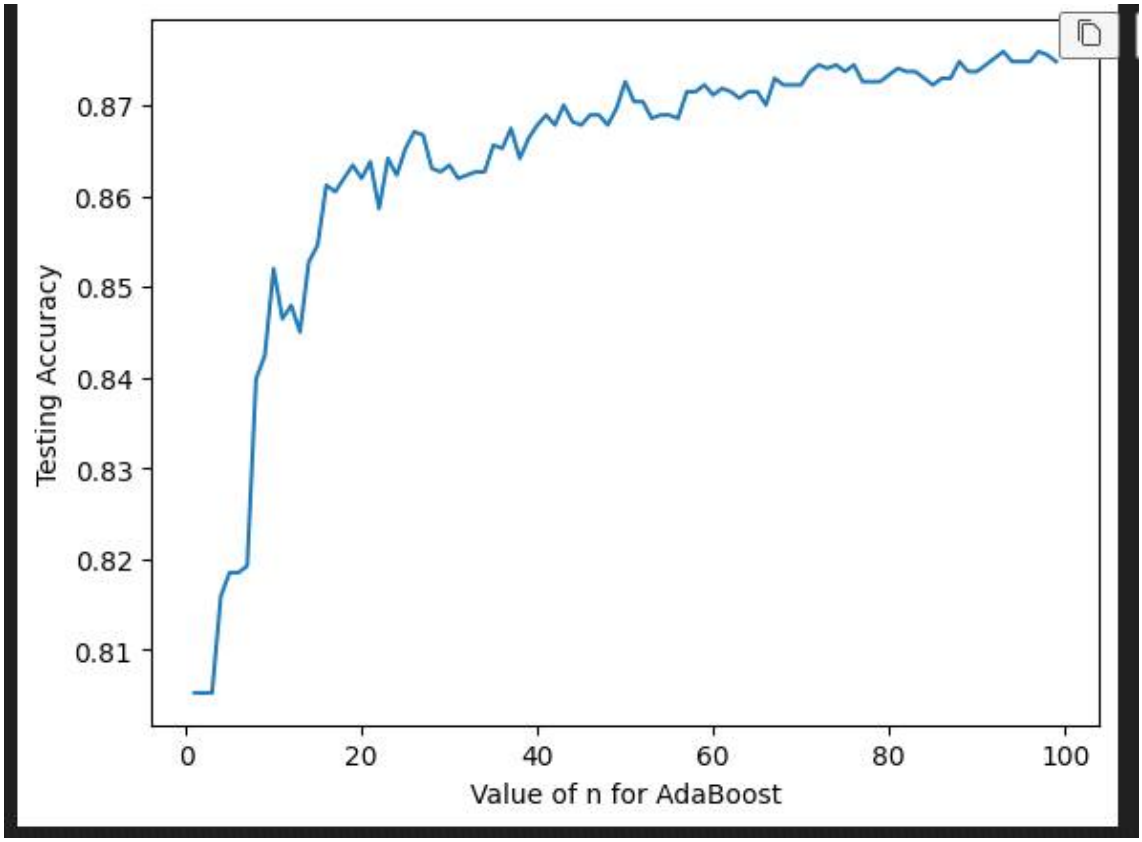
Python

```
[0.8376288659793815, 0.8501472754050073, 0.8634020618556701, 0.8611929307805597, 0.
```



1.8 Adaboost vs. Gaussian Naive Bayes

```
# 7-2. Check Accuracy Score that AdaBoost : Numbers 1 to 100 Trees
n_range = range(1, 100)
accuracy_scores = []

for n in n_range:
    classifier = AdaBoostClassifier(random_state=0, n_estimators=n)
    classifier.fit(X_train, y_train)
    guesses = classifier.predict(X_test)
    accuracy_scores.append(accuracy_score(y_test, guesses))
print(accuracy_scores)

plt.plot(n_range, accuracy_scores)
plt.xlabel('Value of n for AdaBoost')
plt.ylabel('Testing Accuracy')
plt.show()
```



2. Result

데이터 분석 후 정확도를 상승 시키는 전처리 과정을 진행하였습니다.
오브젝트 데이터를 지우고 기본키를 제거하여 전처리를 진행한 뒤 기준이 될 속성값을 선택하였습니다.
선택된 속성값에 대해 트레이닝셋과 테스트 셋의 비율은 8:2으로 선정하였습니다.
Gaussian Naive Bayes 알고리즘을 적용해 정확도를 파악하며 이때 의사정 트리와 랜덤포레스트를 선정하였고, 랜덤포레스트의 트리수를 변화시키며 정확도를 파악한 뒤 이를 시각화 하였습니다.
마지막으로 Adaboost를 이용해 정확도를 파악하였고 랜덤 포레스트와 비교할 시 램덤포레스트 쪽의 정확도가 더 높게 나오는 것을 확인 할 수 있었습니다.