# 빅데이터 분석기사 실기

# 2021년 제2회 기출문제 해설

# 01. 작업형 2

## 작업형 2-1

다음은 Boston Housing 데이터셋이다. crim 항목의 상위에서 10번째 값 (즉, 상위 10번째 값 중에서 가장 작은 값)으로 상위 10개의 값을 변환하고 age 80 이상인 값에 대하여 crim 평균을 구하시오

**< housing.csv >**

| 변수 | 설명 |
|---|---|
| crim | 도시별 1인당 범죄율 |
| zn | 25,000 평방피트 이상의 부지로 구획된 주거용 토지의 비율 |
| Indus | 도시 당 비소매업 에이커 비율 |
| chas | Charles River에 대한 더미 변수<br>(강의 경계에 위치한 경우는 1, 아니면 0) |
| nox | 산화질소 농도 (1천만분의 1) |
| rm | 가구당 평균 방의 개수 |
| age | 1940년 이전에 건축된 소유주택의 비율 |

| 변수 | 설명 |
|---|---|
| dis | 5개의 보스턴 고용센터까지의 가중 거리 |
| rad | 방사형 고속도로 접근성 지수 |
| tax | USD10,000 당 재산세율 |
| pratio | 도시별 학생 – 교사 비율 |
| b | $1000(B - 0.63)^2$ 여기서 B는 도시별 흑인 비율 |
| lstat | 모집단의 하위계층의 비율(%) |
| medv | 소유자가 거주하는 주택의 중앙값(단위:USD1,000) |

# 01. 작업형 2

## 작업형 2-1

```
import pandas as pd
data=pd.read_csv('housing.csv')
# print(data.sort_values(by='crim', ascending=False))
data_sort=data.sort_values(by='crim', ascending=False)
# print(data_sort.head(12))
def recode(series):
    if series>=25.9406:
        return 25.9406
    else:
        return series
data_sort['re_crim']=data_sort['crim'].apply(recode)
# print(data_sort.head(20))
data_80 = data_sort[data_sort['age'] >=80]
# print(data_80.head(50))
print(data_80['crim'].mean())
```

# 01. 작업형 2

## 작업형 2-2

주어진 데이터의 첫 번째 행부터 순서대로 80%까지의 데이터를 훈련 데이터로 추출 후 'total_bedrooms' 변수의 결측값(NA)을 'total_bedrooms' 변수의 중앙값으로 대체하고 대체 전의 'total_bedrooms'변수 표준편차 값의 차이의 절댓값을 구하시오

### <California Housing Prices 데이터셋>

| 변수 | 설명 |
|---|---|
| longitude | 경도 |
| latitude | 위도 |
| housing_median_age | 주택 나이(중앙값) |
| total_rooms | 전체 방 개수 |
| total_bedrooms | 전체 침실 개수 |
| population | 인구 |
| households | 세대 |
| median_income | 소득(중앙값) |
| Median_house_value | 주택가치(중앙값) |
| Ocean_proximity | 바다근접도 |

# 01. 작업형 2

## 작업형 2-2

```
import pandas as pd
data=pd.read_csv('california_housing.csv')
# print(data.info())
data_80=data[:16512]
# print(len(data_80))
pre_std=data_80['total_bedrooms'].std()
print(pre_std)
data_80_fill=data_80.fillna(data_80.median())
post_std=data_80_fill['total_bedrooms'].std()
print(post_std)
print(abs(pre_std-post_std))
```

# 01. 작업형 2

## 작업형 2-3

다음은 Insurance 데이터셋이다. charges 항목의 이상값의 합을 구하시오
(이상값은 평균에서 1.5표준편차 이상인 값)

### < Insurance 데이터셋>

| 변수 | 설명 |
|---|---|
| age | 나이 |
| sex | 성별(male, female) |
| bmi | 체질량 지수 |
| children | 어린이(0,1) |
| smoker | 흡연(yes,no) |
| region | 지역 |
| charges | 요금 |

# 01. 작업형 2

## 작업형 2-3

```
import pandas as pd
data=pd.read_csv('insurance.csv')
# print(data.info())
# print(data['charges'].describe())
mean=data['charges'].mean()
std=data['charges'].std()
# print(mean)
# print(std)
re_data=data[(data['charges']>=mean+1.5*std)|(data['charges']<=mean-1.5*std)]
print(re_data['charges'].sum())
```

# 02. 작업형 3

## 작업형 3

아래 E-Commerce Shipping Data의 train set을 참조하여 고객이 주문한 물품의 정시 도착 여부를 예측하시오. (ID와 예측치를 csv 파일로 저장하여 제출하시오)

### < E-commerce Shipping Data train 데이터셋>

| 변수 | 설명 |
|---|---|
| ID | 고객의 ID 번호 |
| Warehouse_block | 창고의 블록 단위 구역 (A,B,C,D,F) |
| Mode_of_Shipment | 제품 배송 방법 |
| Customer_care_calls | 문의 전화 수 |
| Customer_rating | 고객의 등급 (1:가장 낮음 5:가장 높음) |
| Cost_of_the_Product | 제품의 비용(달러 기준) |
| Prior_purchases | 사전 구매 수량 |
| Product_importance | 제품의 중요도(high, medium, low) |
| Gender | 성별(F: 여성, M: 남성) |
| Discount_offered | 할인혜택 |
| Weight_in_gms | 그램 단위 무게 |
| Reached.on.Time_Y.N | 정시 도착 여부 (1: 정시에 도착하지 않음 0: 정시 도착) |

# 02. 작업형 3

## 작업형 3

```
import pandas as pd
test = pd.read_csv("X_test.csv")
X = pd.read_csv("X_train.csv")
y = pd.read_csv("y_train.csv")

X_num = X[['Customer_care_calls', 'Customer_rating', 'Cost_of_the_Product', 'Prior_purchases', 'Discount_offered', 'Weight_in_gms']]
X_cat = X[['Warehouse_block', 'Mode_of_Shipment', 'Product_importance', 'Gender']]
X_cat=pd.get_dummies(X_cat)

test_num = test[['Customer_care_calls', 'Customer_rating', 'Cost_of_the_Product', 'Prior_purchases', 'Discount_offered', 'Weight_in_gms']]
test_cat = test[['Warehouse_block', 'Mode_of_Shipment', 'Product_importance', 'Gender']]
test_cat = pd.get_dummies(test_cat)

X_cat, test_cat = X_cat.align(test_cat, join='inner', axis=1)

from sklearn.preprocessing import MinMaxScaler
minmax=MinMaxScaler()
minmax.fit(X_num)
X_scaled=minmax.transform(X_num)
test_scaled=minmax.transform(test_num)
```

# 02. 작업형 3

## 작업형 3

```
X_final = pd.concat([pd.DataFrame(X_scaled), X_cat], axis=1)
test_final = pd.concat([pd.DataFrame(test_scaled), test_cat], axis=1)

y = y['Reached.on.Time_Y.N']

from sklearn.linear_model import LogisticRegression
model=LogisticRegression()
model.fit(X_final, y)

pred_test=model.predict_proba(test_final)
pred_test_prob = pd.DataFrame(pred_test[:, 1], columns = ['predict_prob'])
final_predict = pd.concat([test['ID'], pred_test_prob], axis=1)
print(final_predict)
final_predict.to_csv("20211204.csv", index=False)
```