2021학년도 **1**학기 정보과학 프로젝트 제출 양식

간단한, 수업 중 프로젝트 수행평가입니다.

1. 머신러닝,딥러닝을 활용해서 자기 관심 분야와 관련된 모델 구현하여 발표하기

평가 관점은 다음과 같습니다**.**

① 프로젝트에 대한 설명이 구체적으로 설명되어 있다.

② 프로젝트 작품이 충실하게 구현되어있다.

③ 과학 및 수학,정보 분야와 연관성을 가진 주제를 다루고 있다.

④ 과학 및 수학 분야에 대한 흥미와 호기심을 불러일으킬 수 있다.

⑤ 창의적 아이디어가 제시되어 있다.

⑥ 구체적 적용 상황이 제시되어 있다.

⑦ 과학 및 수학,정보 분야의 연구 활동에서의 활용 가능성이 높다.

어떻게 만들어서 제출해야 하는가**?**

- 다음 페이지의 양식에 맞추어 내용을 작성해 제출하면 됩니다.

- 수업시간을 충분히 활용해서 과제를 수행하세요. 미루고 있다가 집에 가서**?** 하는 것 금지**!**

- 매시간 끝날 때, 클래스룸에 수업시간마다 제출합니다**.**

- 매시간 프로젝트 완성도를 체크하여 세특 및 수행평가에 반영합니다.

1. 프로젝트 제출

- 파일 이름 : (1차시)2021정보과학프로젝트(학번,이름)

- 기본 제출

-(1차시)2021정보과학프로젝트(학번,이름)

- 머신러닝, 딥러닝 소스 파일

이 양식 자료를 이쁘게**?(**글꼴**,** 배치 등**)** 꾸미려고 시간 버리지 말 것**!**

핵심으로 정확하게**!** 콕**! ,** 분량은 중요하지 않지만**,** 노력은 중요함**!**

대표 팀**:** 삼삼화재

학번 : 30308 30309 30310

이름 : 신영빈 윤태경 이준승

나이, 성별, BMI, 거주 지역 등의 정보로 건강보험료 예측하기(최종)

제목 : Prediction of Health Insurance Cost by Machine learning

만든 것에 대한 설명

- 관련 수학/과학 교과명 : 생명과학

- 관련 내용 또는 주제 : 의료데이터 사이언스, 빅데이터 AI, 바이오메디컬

1. 연구 주제 결정하기

현재 공공과 민간의 보건의료 빅데이터는 쌓여가고 있으나, 아직 이에 대한 활용은 초기 단계에 지나지 않고 있다. (데이터 표준화, 개인정보 문제 등 때문에)

현재 미국의 건강보험은 민영국민의료보험제도를 채택하고 있다.

이는 보험에 가입해야하는 '국민'과 보험 상품을 파는 '보험사' 간에는 정보의 비대칭성을 야기한다. 이에 민영기업은 건강한 가입자들을 선별하기 위해 노력한다.

본 연구에서는 공공 보건 의료기관 보유 데이터 중 의료보험 데이터를 사용하여 가입자의 관련 조건에 근거한 건강 보험료 예측 프로그램을 만들고자 한다.

개인 맞춤형 헬스케어 시대의 흐름에 맞춘 의료 서비스를 개발에 직접 참여하고자 한다

2. 문제 분석하기

건강 보험사의 의료보험데이터를 기반으로 새로운 가입자의 건강 보험료를 예측한다.

기존 가입자의 정보와 보험료를 분석하여 머신러닝 회귀과 딥러닝 회귀를 통해 테스트 데이터의 건강 보험료를 예측한다.

3. 관련 내용 학습하기

회귀: 여러 개의 독립변수(x)와 한 개의 종속변수(y) 간의 상관관계를 모델링

IQR 방식은 사분위(Quantile) 개념으로부터 출발합니다. 전체 데이터들을 오름차순으로 정렬하고, 정확히 4등분(25%, 50%, 75%, 100%)으로 나눕니다. 여기서 75% 지점의 값과 25% 지점의 값의 차이를 IQR이라고 합니다.

Linear Regression: 회귀 알고리즘으로 데이터를 가장 잘 대표하는 하나의 직선을 찾는 것이다.

Random Forest: 다수의 결정 트리들을 학습하여 최적화된 성능을 제시하는 방법

Gradient Boosting Algorithm (GBM): 앙상블 알고리즘 중 boosting 계열의 알고리즘으로 서로 다른 정확도를 가진 분류 방법들을 결합해 더 높은 정확도를 도출하는 것이다.

교차 검증(Cross Validation)은 머신러닝/딥러닝 평가에 필수적으로 사용되는 방법으로 데이터를 통한 모델을 설계한 후 모델을 검증하는 단계라고 볼 수 있다. 다시 말해 모델을 추정하는 데 사용되지 않았던 새로운 데이터를 예측하는 일반화 능력을 테스트하는 방법이다.

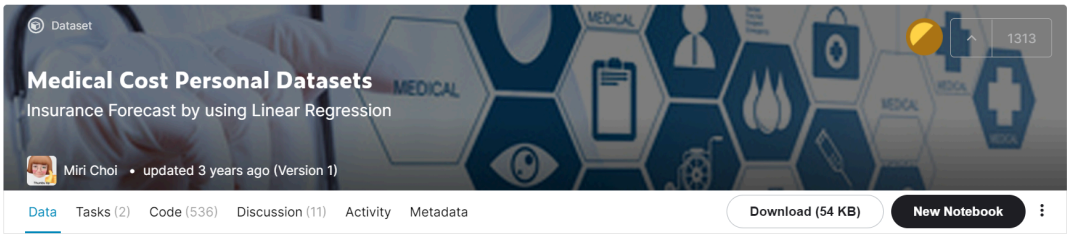
MAE: Mean Absolute Error의 약자이며, 오차(예측값-실제값)의 절대값의 평균이다.

MSE: Mean Square Error의 약자로 오차(실제값-예측값)를 제곱하여 평균을 계산한 지표이다.

RMSE: Root Mean Square Error의 약자로 MSE에 루트를 씌워 오류의 제곱으로 인해 실제 오류 평균보다 커지는 특성을 보완한 방법이다.

4. 데이터셋 획득과정과 분석

관련 데이터는 캐글 Medical Cost Personal Datasets
https://www.kaggle.com/mirichoi0218/insurance 를 사용하였다. 보험료 관련 표준화된 공공데이터셋이 없기 때문에 가장 관련도 높은 kaggle의 데이터 셋을 활용하였습니다.



```
# insurance.csv 파일을 pandas dataframe으로 읽기
data = pd.read_csv('/content/insurance.csv')
```

```
data.head()
```

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520

컬럼은 age(나이), sex(성별), bmi(체질량지수 ideally 18.5 to 24.9), children(건강보험으로 보장되는 자녀 수), smoker(흡연자), region(거주지역), charges(건강보험료 의료비) 입니다.

data.shape는 (1388,7)

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1338 entries, 0 to 1337
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       1338 non-null   int64  
 1   sex       1338 non-null   object 
 2   bmi       1338 non-null   float64
 3   children  1338 non-null   int64  
 4   smoker    1338 non-null   object 
 5   region    1338 non-null   object 
 6   charges   1338 non-null   float64
dtypes: float64(2), int64(2), object(3)
memory usage: 73.3+ KB
```

```
#결측값 확인
data.isnull().sum()
```

```
age         0
sex         0
bmi         0
children    0
smoker      0
region      0
charges     0
dtype: int64
```

Dtype이 object인 column은 sex,smoker,region 입니다. 결측 값은 없습니다.

자료 분석 및 군집화

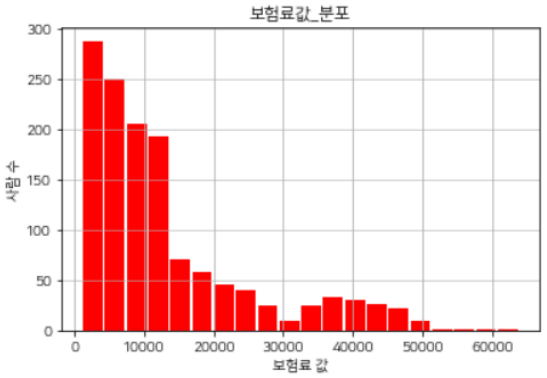
각 변수가 charges와 가지는 상관관계를 파악합니다.

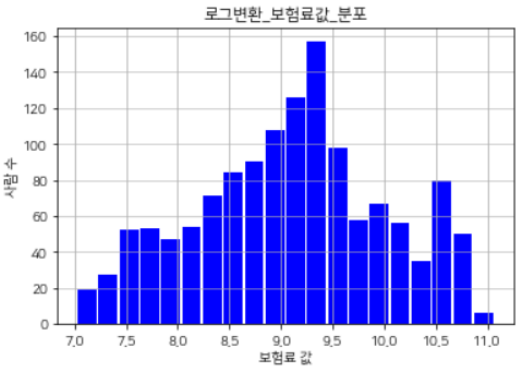
1) 각 컬럼별 분포유형 파악
2) 나이와 BMI지수, 요금 관계
3) 요금과 각 column별 유형 파악하기

4) 요금에 큰 영향을 미치는 체중, BMI, 흡연관계

5) 전체 데이터 방향성 확인 (흡연유무 중심)

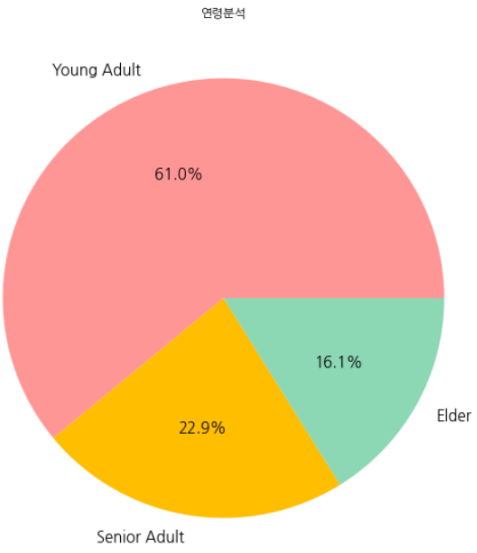
1. 보험료값 분포

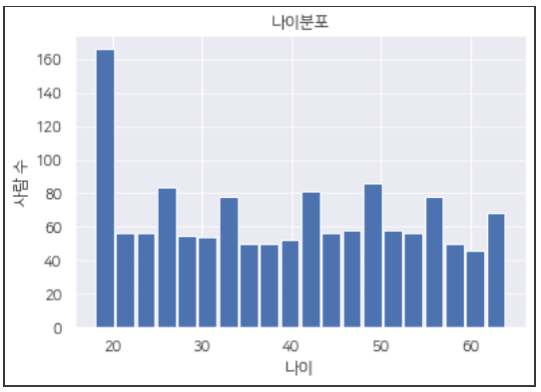




대부분의 환자가 2000달러 ~12000달러 사이에 청구되는 왼쪽으로 치우친 분포를 가지고 있다. Logarithms를 사용하여 보험료값 분포를 정규분포로 변환한다.

2. 연령대별 분포





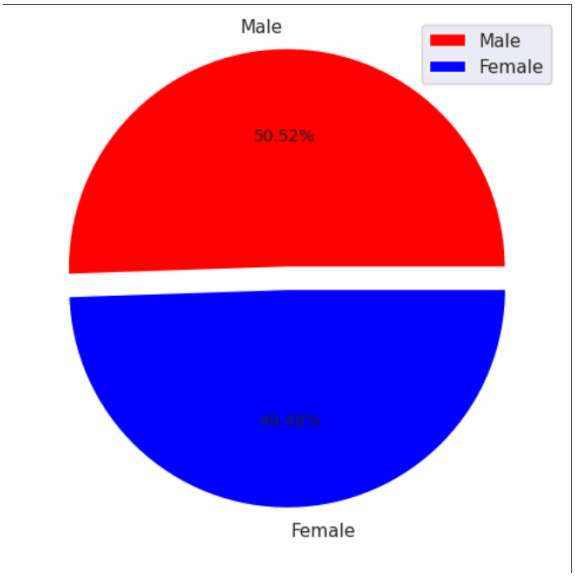
연령의 분포는 20대 이하의 청소년 분포가 가장 높으며, 이외 고른 분포를 가진다.
연령을 범주형 변수로 전환하여 청소년(61%), 성인(22.9%), 노인(16.1%)를 확인한다.

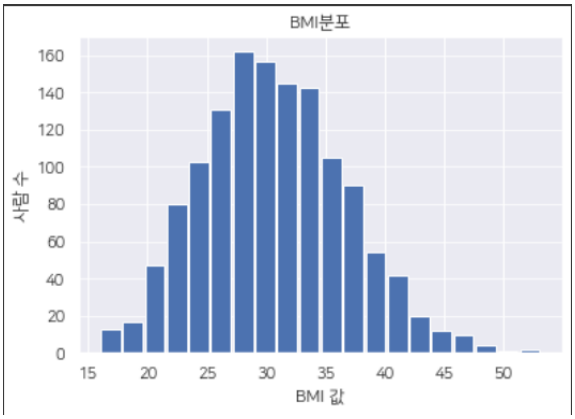
```
for col in lst:
    col.loc[(col['age'] >= 18) & (col['age'] <= 25), 'age_cat'] = 'Young Adult'
    col.loc[(col['age'] > 25) & (col['age'] <= 55), 'age_cat'] = 'Senior Adult'
    col.loc[col['age'] > 55, 'age_cat'] = 'Elder'
```

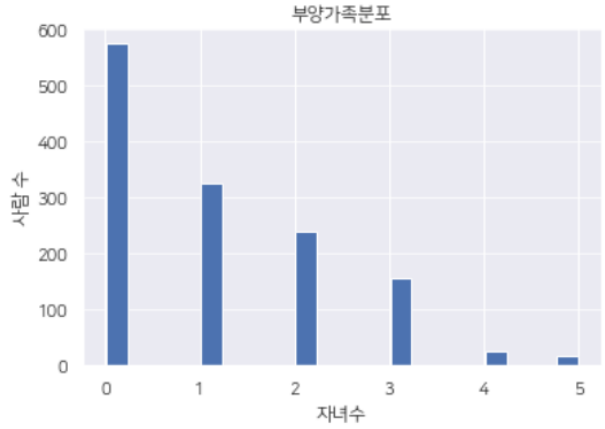
3. 성별**, BMI,** 부양가족 분포

성별의 분포는 1:1 양상을 보이며, BMI분포는 27~32 값에 가장 많이 분포한다.

부양가족수는 0명이 가장 많고, 최대 5명까지 있다.







4. BMI값은 범주형 변수로 변환하여 분석한다**.**

```
data["weight_condition"] = np.nan
lst = [data]

for col in lst:
    col.loc[col["bmi"] < 18.5, "weight_condition"] = "Underweight"
    col.loc[(col["bmi"] >= 18.5) & (col["bmi"] < 24.986), "weight_condition"] = "Normal Weight"
    col.loc[(col["bmi"] >= 25) & (col["bmi"] < 29.926), "weight_condition"] = "Overweight"
    col.loc[col["bmi"] >= 30, "weight_condition"] = "Obese"
```

저체중 : 체질량 지수 (BMI) <18.5

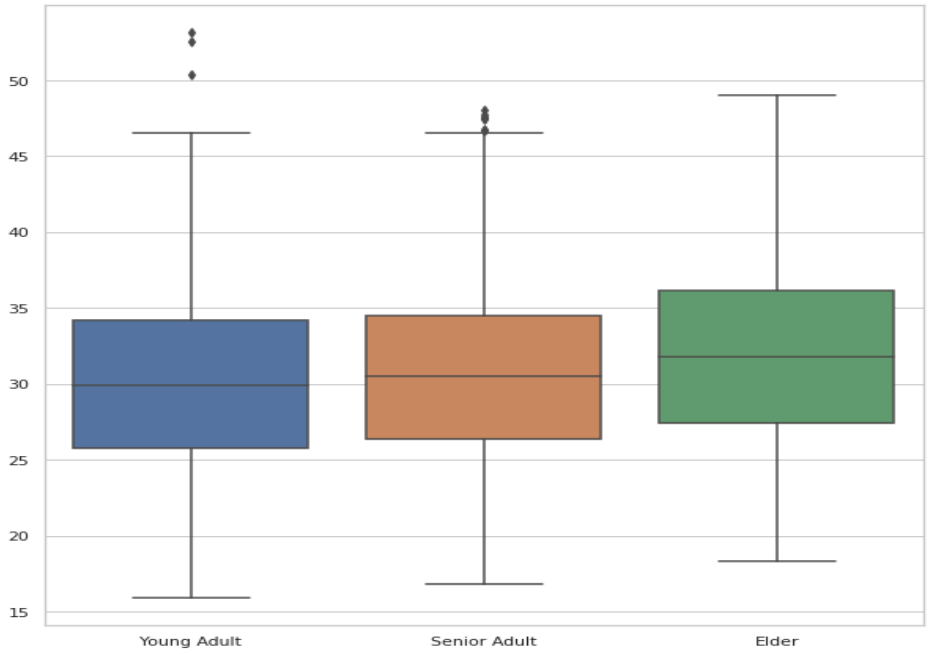
정상 체중 : 체질량 지수 (BMI) ≥ 18.5 및 체질량 지수 (BMI) <24.9

과체중 : 체질량 지수 (BMI) ≥ 25 및 체질량 지수 (BMI) <29.9

비만 : 체질량 지수 (BMI)> 30

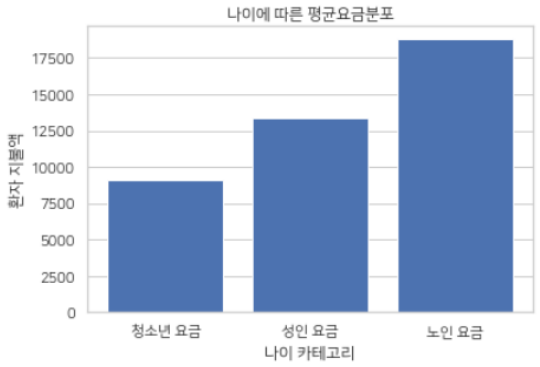
5. 나이와 **BMI**값 관계를 파악합니다**.**

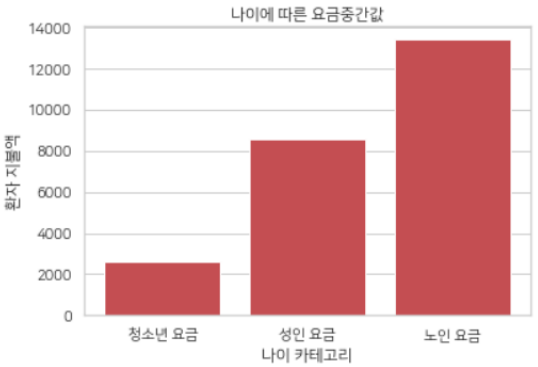
```
import sklearn
young_adults = df["bmi"].loc[df["age_cat"] == "Young Adult"].values
senior_adult = df["bmi"].loc[df["age_cat"] == "Senior Adult"].values
elders = df["bmi"].loc[df["age_cat"] == "Elder"].values
```



나이와 BMI 두 변수의 상관관계는 0.1 정도로 미미하고 세 연령 범주가 BMI 지수에 큰 영향을 미치지 않는다는 것을 알 수 있다.

6. 나이와 평균 요금분포의 상관관계를 파악한다**.**

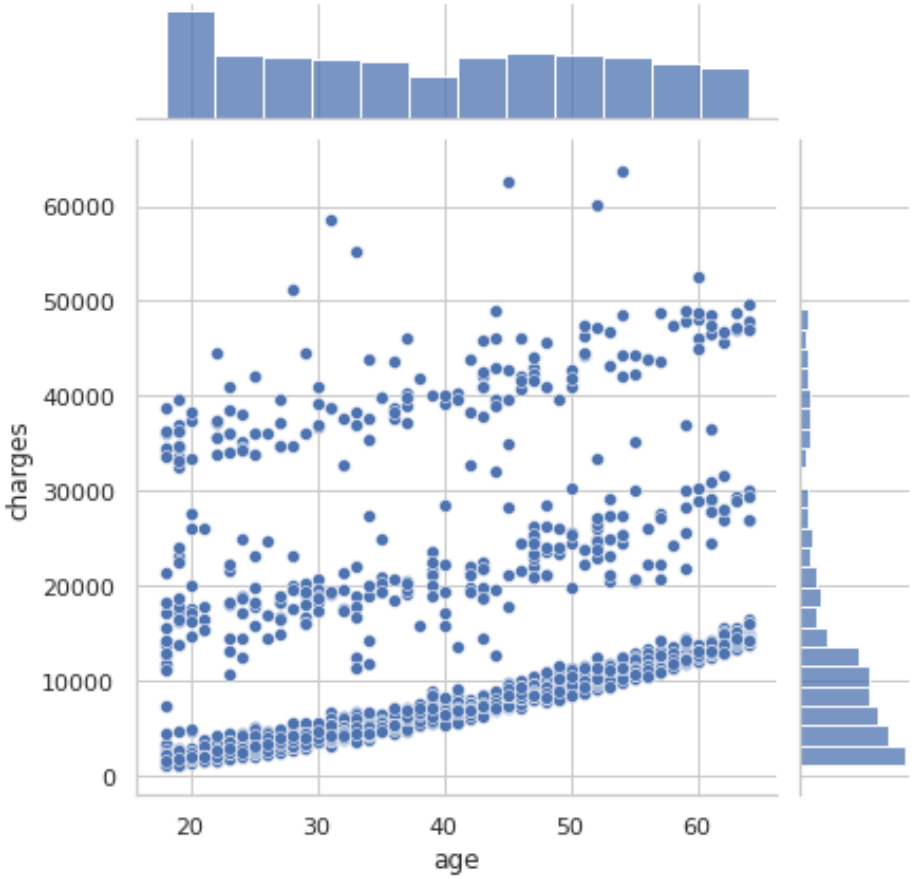




환자 부담 평균을 살펴보면 청소년 : 7,944, 성인 : 14,785, 노인 : 18,795 이다.

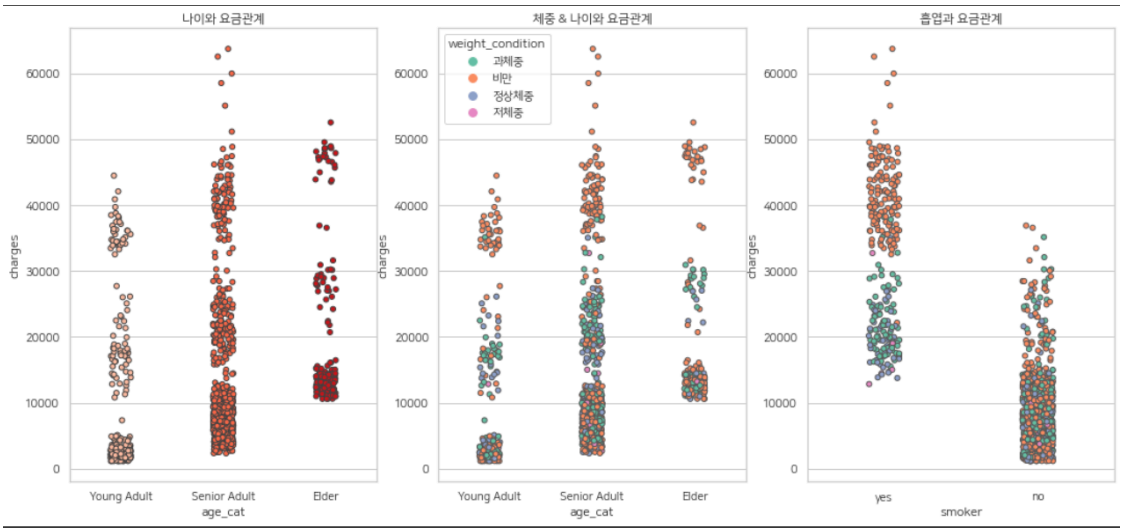
환자 요금의 중앙값은 청소년은 4,252 명, 성인은 9,565 명, 노인은 13,429 명이다.

이때, 평균 및 중앙값이 특이 치의 영향을 받기 쉽기 때문에 (여기서는 크게 영향을 주는 요인 없을 것으로 예상) 주의가 필요하다.



6. 요금과 각 칼럼별 유형 파악하기

```
f, (ax1, ax2, ax3) = plt.subplots(ncols=3, figsize=(18,8))

# I wonder if the cluster that is on the top is from obese people
sns.stripplot(x="age_cat", y="charges", data=df, ax=ax1, linewidth=1, palette="Reds")
ax1.set_title("나이와 요금관계")


sns.stripplot(x="age_cat", y="charges", hue="weight_condition", data=df, ax=ax2, linewidth=1, palette="Set2")
ax2.set_title("체중 & 나이와 요금관계")

sns.stripplot(x="smoker", y="charges", hue="weight_condition", data=df, ax=ax3, linewidth=1, palette="Set2")
ax3.legend_.remove()
ax3.set_title("흡연과 요금관계")

plt.show()
```



보험료에 높은 영향을 미치는 요인들과 그 요인들의 상관관계를 분석하기 위해

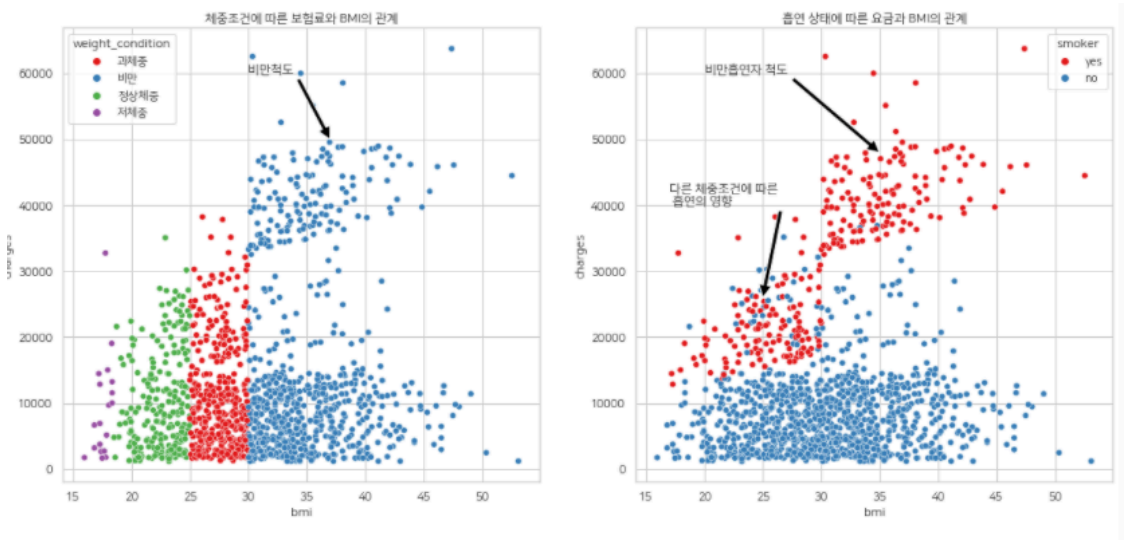
'나이', '체중', '흡연 유무' 와 보험료의 관계를 시각화해보았다. 각 요인별로 (청소년, 성인, 노인), (저체중, 정상, 과체중, 비만), (흡연자, 비흡연자)의 그룹

비만과 흡연의 영향이 돋보인다.

나이와 요금과 기타 상관관계가 떨어지므로, 체중조건&BMI, 체중조건&흡연 변수가 건강보험 요금에 미치는 영향을 scatter 플롯으로 시각화 한다.

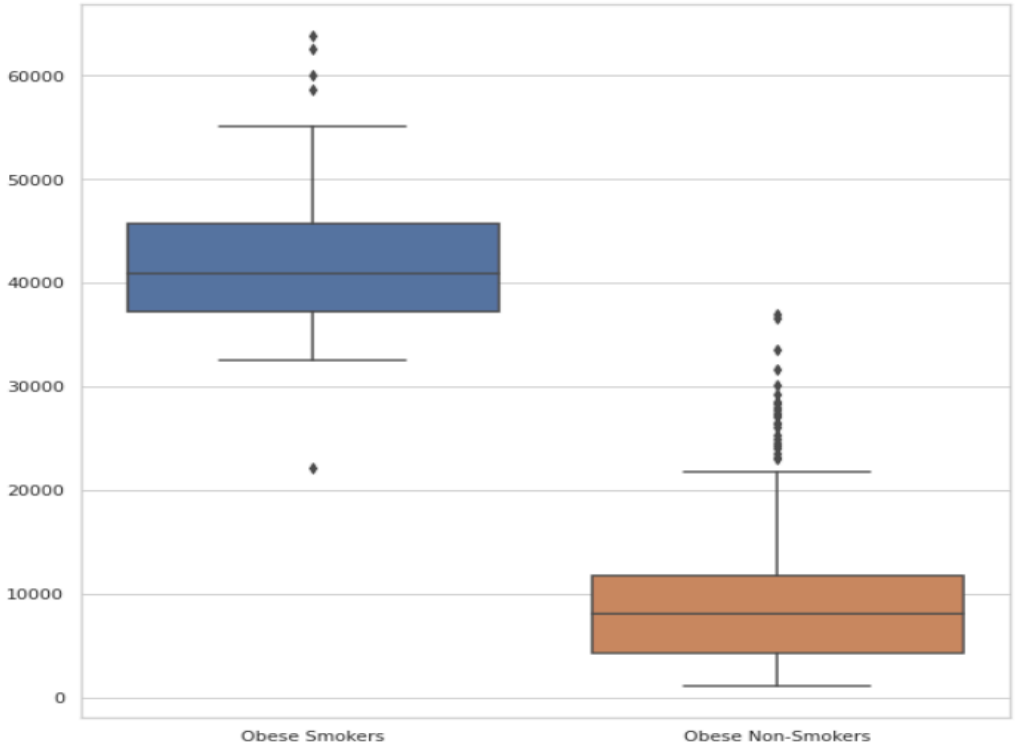
```
sns.scatterplot(x="bmi", y="charges", hue="weight_condition", data=df, palette="Set1", ax=ax1)
ax1.set_title("체중조건에 따른 보험료와 BMI의 관계")
ax1.annotate('비만척도 ', xy=(37, 50000), xytext=(30, 60000),
             arrowprops=dict(facecolor='black'),
             fontsize=12)
sns.scatterplot(x="bmi", y="charges", hue="smoker", data=df, palette="Set1", ax=ax2)
ax2.set_title("흡연 상태에 따른 요금과 BMI의 관계")
ax2.annotate('비만흡연자 척도 ', xy=(35, 48000), xytext=(20, 60000),
             arrowprops=dict(facecolor='black'),
             fontsize=12)
ax2.annotate('다른 체중조건에 따른 ₩n 흡연의 영향 ', xy=(25, 26000), xytext=(17, 40000),
             arrowprops=dict(facecolor='black'),
             fontsize=12)
```

6. 요금에 큰 영향을 미치는 흡연 **&** 체중 상관관계 분석



왼쪽 그래프에서 마찬가지로 비만 그룹에서 요금이 높은 분포가 발생하였고 이는 '흡연' 그룹이었다. 오른쪽 그래프에서 흡연과 비만 그룹 중 요금이 높은 분포가 일치하는 것을 확인했다.

결과적으로 체중과 흡연 변수간 상관관계가 높으며 이것이 요금에 높은 비중을 차지한다.



비만 흡연자와 정상 체중의 흡연자 간의 보험료 분리를 한 이 차트에서 비만 흡연자와 정상 체중 흡연자를 서로 다른 그룹으로 분리하는 방법을 시각화하였다

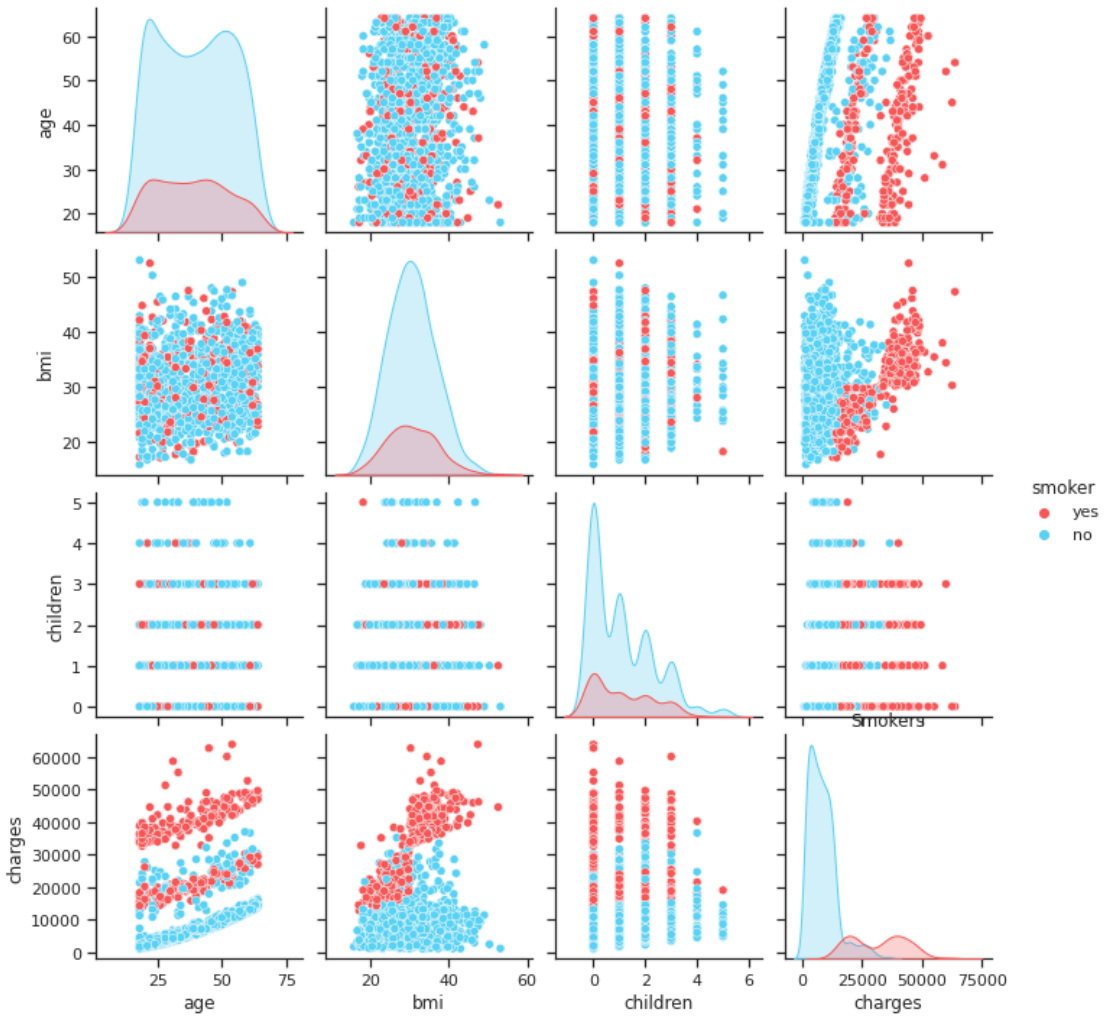
비만 + 흡연자 대부분의 비만 흡연자의 의료비 : 4 만 달러.

비만 + 비흡연자는 의료비 : 8천 달러

비만 환자일 경우 흡연의 유무는 5배 이상의 차이를 내는 중요한 요소이다.

(따라서 두 조건을 모두 만족하는 경우에는 가중치를 높여야 할 것이다.)

6. 전체 **column**과 흡연관계 유형요약



흡연이 charges와 age, bmi와 일정한 양상을 보임을 알수있다.

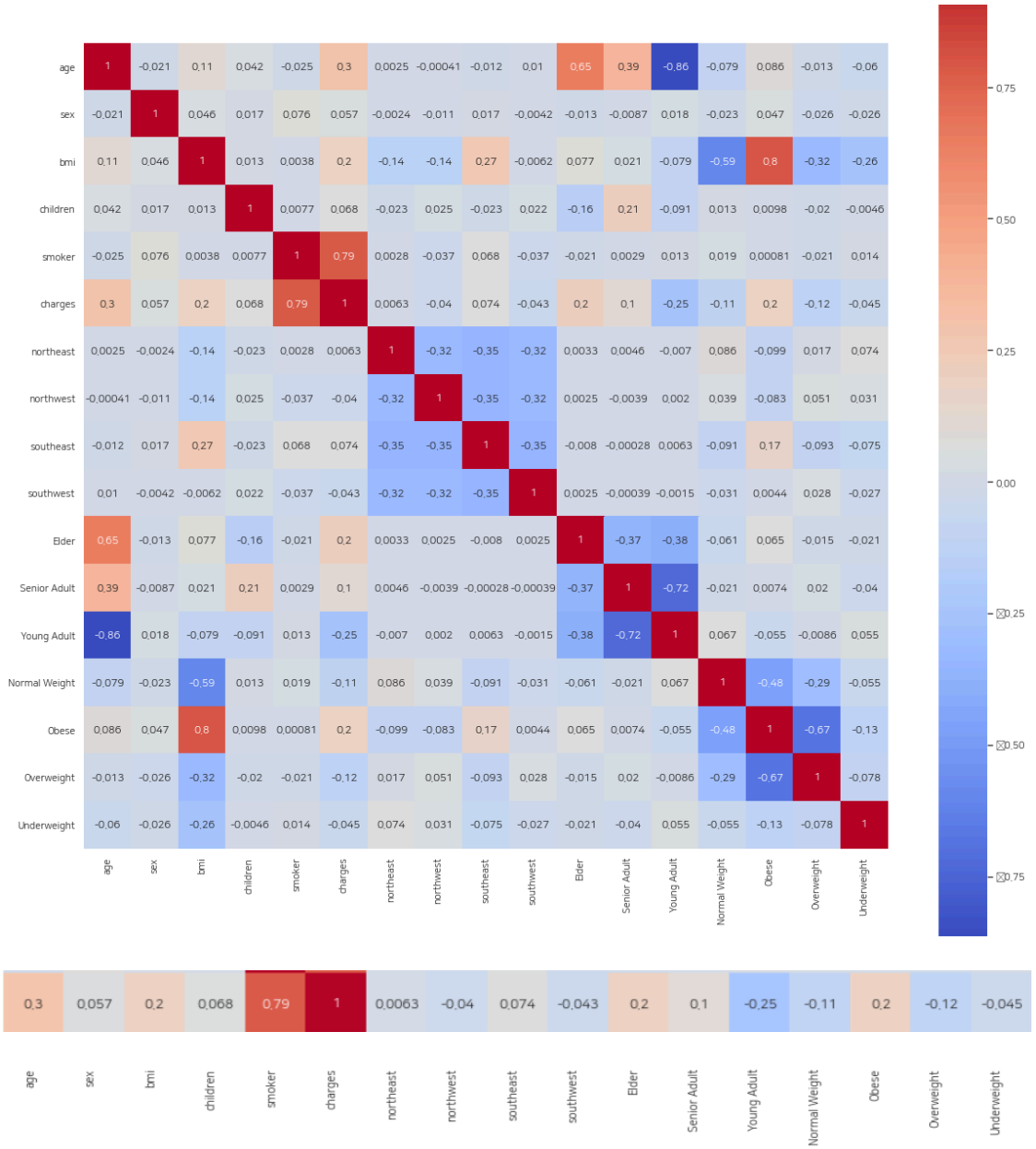
지금까지 수동으로 클러스터링을 수행하였다.

흡연 군집: 높은 요금의 가장 큰 비율을 차지한다.

비만 군집 : 각 연령 그룹에 대해 요금의 상단 부분에 비만 군집이 있음을 알 수 있습니다.

연령계수를 추가하여 변수를 비교하였음에도 흡연과 비만의 영향이 크다는 것을 알 수 있었다.

#heatmap 살펴보기



charges 와 나이 유형간 상관관계가 떨어짐을 확인 할 수있다.

기존 칼럼 5개보다 가중치 분포가 적절히 분배되었음,

더하여 가장 가중치가 컸던 비만과 흡연을 포함시킨 칼럼을 추가하였다. (관련도:0.64)

```
data.corrwith(data['charges'])
```

```
age               0.299008
sex               0.057292
bmi               0.198341
children          0.067998
smoker            0.787251
charges           1.000000
northeast         0.006349
northwest        -0.039905
southeast         0.073982
southwest        -0.043210
Normal Weight    -0.106265
Obese             0.199532
Overweight       -0.120083
Underweight      -0.044960
smoker_obese      0.649832
dtype: float64
```

5. 데이터셋 전처리 과정을 구체적으로 제시하기

```
# insurance.csv 파일을 pandas dataframe으로 읽기
data = pd.read_csv('/content/insurance.csv')
```

#1. 연령대별 칼럼생성하기

```
for col in lst:
    col.loc[(col['age'] >= 18) & (col['age'] <= 25), 'age_cat'] = 'Young Adult'
    col.loc[(col['age'] > 25) & (col['age'] <= 55), 'age_cat'] = 'Senior Adult'
    col.loc[col['age'] > 55, 'age_cat'] = 'Elder'
```

#2. bmi에서 체중척도 생성하기

```
for col in lst:
    col.loc[col["bmi"] < 18.5, "weight_condition"] = "Underweight"
    col.loc[(col["bmi"] >= 18.5) & (col["bmi"] < 24.986), "weight_condition"] = "Normal Weight"
    col.loc[(col["bmi"] >= 25) & (col["bmi"] < 29.926), "weight_condition"] = "Overweight"
    col.loc[col["bmi"] >= 30, "weight_condition"] = "Obese"
```

#3. 카테고리형 데이터(Categorical Data)를 수치형 데이터(Numerical Data)로 변환

```
# sex column ['female', 'male']->[0, 1] 변환
data.sex.unique()
data['sex'] = data['sex'].replace(('female', 'male'), (0, 1))

# smoke column ['yes', 'no']->[2, 1] 변환
data['smoker'] = data['smoker'].replace(('yes', 'no'), (2, 1))
```

```
# region column을 원-핫 인코딩하여 수치형 데이터로 변환

region_dummies = pd.get_dummies(data['region'])
```

```
# age_cat 원-핫 인코딩

new_data['age_cat'].unique()
age_cat_dummies = pd.get_dummies(new_data['age_cat'])

new_data = pd.concat([new_data, age_cat_dummies], axis = 1)
new_data.drop(['age_cat'], axis = 1, inplace = True)
```

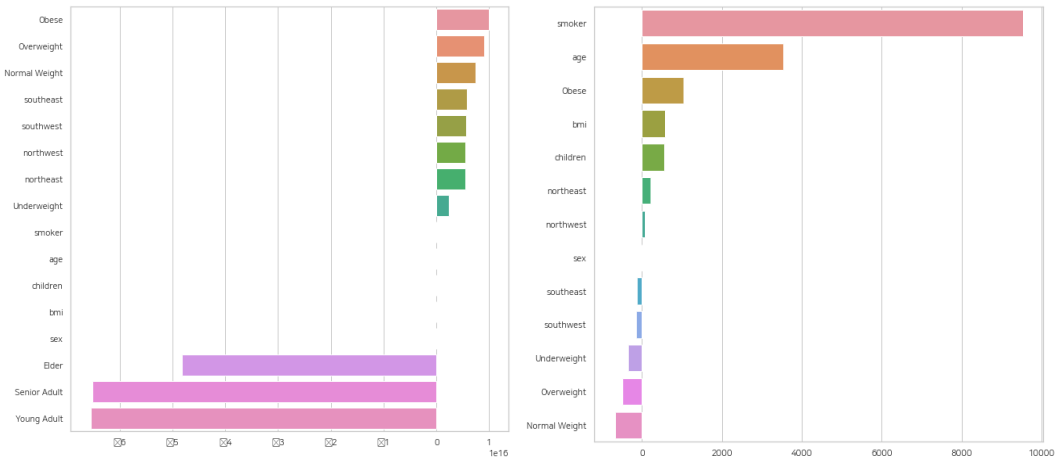
```
# weight_condition 원-핫 인코딩

new_data['weight_condition'].unique()
age_cat_dummies = pd.get_dummies(new_data['weight_condition'])
```

```
data.shape
```

```
(1338, 17)
```

#4. IQR 방식을 이용한 이상치 데이터(Outlier) 제거

```
def get_outlier(df=None, column=None, weight=1.5):
  # target 값과 상관관계가 높은 열을 우선적으로 진행
  quantile_25 = np.percentile(new_data[column].values, 25)
  quantile_75 = np.percentile(new_data[column].values, 75)

  IQR = quantile_75 - quantile_25
  IQR_weight = IQR*weight

  lowest = quantile_25 - IQR_weight
  highest = quantile_75 + IQR_weight

  outlier_idx = new_data[column][ (new_data[column] < lowest) | (new_data[column] > highest) ].index
  return outlier_idx

# 함수 사용해서 이상치 값 삭제
outlier_idx = get_outlier(df=new_data, column='charges', weight=1.5)
new_data.drop(outlier_idx, axis=0, inplace=True)
```




```
(1338, 13)
Index(['age', 'sex', 'bmi', 'children', 'smoker', 'northeast', 'northwest',
       'southeast', 'southwest', 'Normal Weight', 'Obese', 'Overweight',
       'Underweight'],
      dtype='object')
```

#5. 데이터 정규화

```
y = new_data['charges']
x = new_data.drop(['charges'], axis = 1)
```

```
from sklearn.preprocessing import StandardScaler

scaler_x = StandardScaler()
x_scale = scaler_x.fit_transform(x)

scaler_y = StandardScaler()
y_scale = scaler_y.fit_transform(y.reshape(-1,1))
```

5. 모델 설계하기(팀원들끼리 분담하여 작성해봅시다)

1. 머신러닝 모델

#Linear Regression (LR)

```
from sklearn.linear_model import LinearRegression

#LinearRegression
model = LinearRegression()
model.fit(x_train, y_train)

y_predict = model.predict(x_test)
```

#Random Forest regressor (RF)

```
#RandomForestRegressor
from sklearn.ensemble import RandomForestRegressor

rf_model = RandomForestRegressor()
rf_model.fit(x_train, y_train)

y_predict_rf = rf_model.predict(x_test)
```

Gradient Boosting Algorithm (GBM)

```
#Gradient Boosting Regressor Model
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error

gb_model = GradientBoostingRegressor(max_depth=2, n_estimators=100, learning_rate =.1)
gb_model.fit(x_train, y_train)

y_predict_gb = gb_model.predict(x_test)
```

그리드 서치 모델 추가적 도입 및 교차검증

```
# 파라미터 그리드 생성
param_grid = {
    'bootstrap': [True],              # 학습셋 중 무작위로 중복 허용하여 데이터 선택
    'max_depth': [4, 5, 6, 7],        # 트리 최대 깊이
    'max_features': [3, 4, 5],        # 트리 학습에 사용할 최대 피처 수
    'min_samples_leaf': [2, 3, 4],    # 최소 샘플 리프 수
    'min_samples_split': [3, 4, 5],   # 최소 분할 샘플 수
    'n_estimators': [10, 30, 50, 100] # 생성할 트리의 개수
}

# 베이스 모델 정의
rf = RandomForestRegressor(criterion='mse')

# 그리드 서치 모델 인스턴스화
grid_search_rf = GridSearchCV(estimator = rf, param_grid = param_grid,
                          cv = 5, n_jobs = -1, verbose = 4)
```

```
# 그리드 서치 모델 학습
grid_search_rf.fit(x_train,y_train)

# 최적 파라미터 출력
grid_search_rf.best_params_
```

```
Fitting 5 folds for each of 432 candidates, totalling 2160 fits
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=-1)]: Done  64 tasks      | elapsed:    3.8s
[Parallel(n_jobs=-1)]: Done 356 tasks      | elapsed:   20.7s
[Parallel(n_jobs=-1)]: Done 848 tasks      | elapsed:   50.2s
[Parallel(n_jobs=-1)]: Done 1532 tasks      | elapsed:  1.5min
[Parallel(n_jobs=-1)]: Done 2160 out of 2160 | elapsed:  2.2min finished
{'bootstrap': True,
 'max_depth': 7,
 'max_features': 5,
 'min_samples_leaf': 2,
 'min_samples_split': 4,
 'n_estimators': 50}
```

```
# 테스트 데이터를 모델에 대입하여 RMSE, MSE, MAE 출력
y_pred = grid_search_rf.predict(x_test)
```

2. 딥러닝 모델

```
from tensorflow.keras.layers import Dense, Activation, Dropout, BatchNormalization
from tensorflow.keras.models import Sequential
from tensorflow.keras.optimizers import Adam

# 모델 정의
model = Sequential()
model.add(Dense(50,input_dim=16))
model.add(Activation('relu'))

model.add(Dense(50))
model.add(Activation('relu'))
model.add(Dropout(0.3))

model.add(Dense(60))
model.add(Activation('relu'))
model.add(Dropout(0.5))

model.add(Dense(100))
model.add(Activation('sigmoid'))
model.add(Dropout(0.3))

model.add(Dense(100))
model.add(Activation('elu'))
model.add(Dropout(0.3))
model.add(Dropout(0.3))

model.add(Dense(324))
model.add(Activation('linear'))
model.add(Dropout(0.5))

model.add(Dense(120))
model.add(Activation('linear'))
model.add(Dropout(0.3))

model.add(Dense(1))
# 모델 컴파일
model.compile(loss='mse',optimizer='adam')

# 모델 요약
model.summary()
```

```
from sklearn.model_selection import cross_val_score
score = cross_val_score(model, x, y, cv=5)
print(score)
```

```
Model: "sequential_32"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_232 (Dense)            (None, 50)                850
_________________________________________________________________
activation_202 (Activation)  (None, 50)                0
_________________________________________________________________
dense_233 (Dense)            (None, 50)                2550
_________________________________________________________________
activation_203 (Activation)  (None, 50)                0
_________________________________________________________________
dropout_178 (Dropout)        (None, 50)                0
_________________________________________________________________
dense_234 (Dense)            (None, 60)                3060
_________________________________________________________________
activation_204 (Activation)  (None, 60)                0
_________________________________________________________________
dropout_179 (Dropout)        (None, 60)                0
_________________________________________________________________
dense_235 (Dense)            (None, 100)               6100
_________________________________________________________________
activation_205 (Activation)  (None, 100)               0
_________________________________________________________________
dropout_180 (Dropout)        (None, 100)               0
_________________________________________________________________
dense_236 (Dense)            (None, 100)               10100
_________________________________________________________________
activation_206 (Activation)  (None, 100)               0
_________________________________________________________________
dropout_181 (Dropout)        (None, 100)               0
_________________________________________________________________
dropout_182 (Dropout)        (None, 100)               0
_________________________________________________________________
dense_237 (Dense)            (None, 324)               32724
_________________________________________________________________
activation_207 (Activation)  (None, 324)               0
_________________________________________________________________
dropout_183 (Dropout)        (None, 324)               0
_________________________________________________________________
dense_238 (Dense)            (None, 120)               39000
_________________________________________________________________
activation_208 (Activation)  (None, 120)               0
_________________________________________________________________
dropout_184 (Dropout)        (None, 120)               0
_________________________________________________________________
dense_239 (Dense)            (None, 1)                 121
=================================================================
Total params: 94,505
Trainable params: 94,505
Non-trainable params: 0
_________________________________________________________________
```

6.모델 학습 후 성능 파악하기

```
# checking model accuracy
from sklearn.metrics import r2_score, mean_squared_error
import numpy as np

lr_r2_score = r2_score(y_test, y_predict)
print('R square Score = ', round(lr_r2_score, 3))

mse = mean_squared_error(y_test, y_predict)
rmse = np.sqrt(mse)
print('Root Mean Squared Error = ', round(rmse, 3))
```

```
R square Score =  0.786
Root Mean Squared Error =  0.424
```

```
# checking model accuracy
from sklearn.metrics import r2_score, mean_squared_error
import numpy as np
rf_r2_score = r2_score(y_test, y_predict_rf)
print('R square Score = ', round(rf_r2_score, 3))

rf_mse = mean_squared_error(y_test, y_predict_rf)
rf_rmse = np.sqrt(rf_mse)
print('Root Mean Squared Error = ', round(rf_rmse, 3))
```

```
R square Score =  0.856
Root Mean Squared Error =  0.347
```

```
# checking model accuracy
from sklearn.metrics import r2_score, mean_squared_error
import numpy as np
rf_r2_score = r2_score(y_test, y_predict_rf)
print('R square Score = ', round(rf_r2_score, 3))

rf_mse = mean_squared_error(y_test, y_predict_rf)
rf_rmse = np.sqrt(rf_mse)
print('Root Mean Squared Error = ', round(rf_rmse, 3))
```

```
R square Score =  0.864
Root Mean Squared Error =  0.338
```

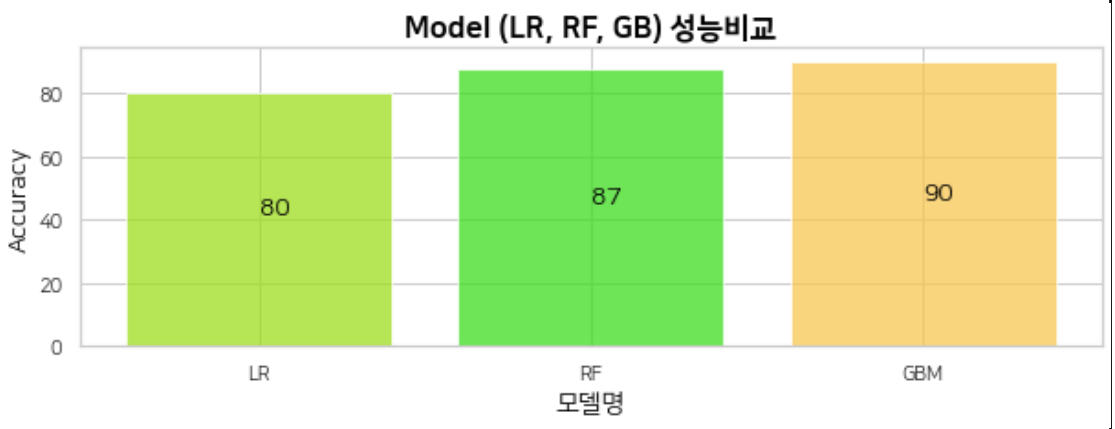
```
# checking model accuracy
from sklearn.metrics import r2_score, mean_squared_error
import numpy as np
gb_r2_score = r2_score(y_test, y_predict_gb)
print('R square Score = ', round(gb_r2_score, 3))

gb_mse = mean_squared_error(y_test, y_predict_gb)
gb_rmse = np.sqrt(gb_mse)
print('Root Mean Squared Error = ', round(gb_rmse, 3))

RMSE = float(format(np.sqrt(mean_squared_error(y_test, y_predict_gb)),'.3f'))
MSE = mean_squared_error(y_test, y_predict_gb)
MAE = mean_absolute_error(y_test, y_predict_gb)

print('RMSE =',RMSE,
      '₩nMSE =',MSE,
      '₩nMAE =',MAE)
```

```
R square Score =  0.866
Root Mean Squared Error =  0.356
RMSE = 0.356
MSE = 0.1263897284438338
MAE = 0.19488172138306092
```

그리드 서치모델 추가

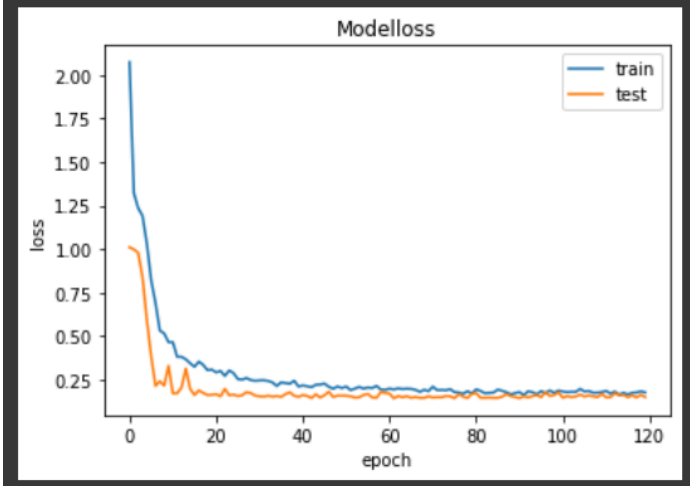
```
#교차검증
from sklearn.model_selection import cross_val_score
score = cross_val_score(gb_model, x, y, cv=5)
print(score)
```

```
[0.88508309 0.80678617 0.89000494 0.85124971 0.86235438]
```



딥러닝 모델 성능

```
# 학습 데이터 손실, 테스트 데이터 손실 출력
modelplot(hist, 'loss', 'val_loss')
```

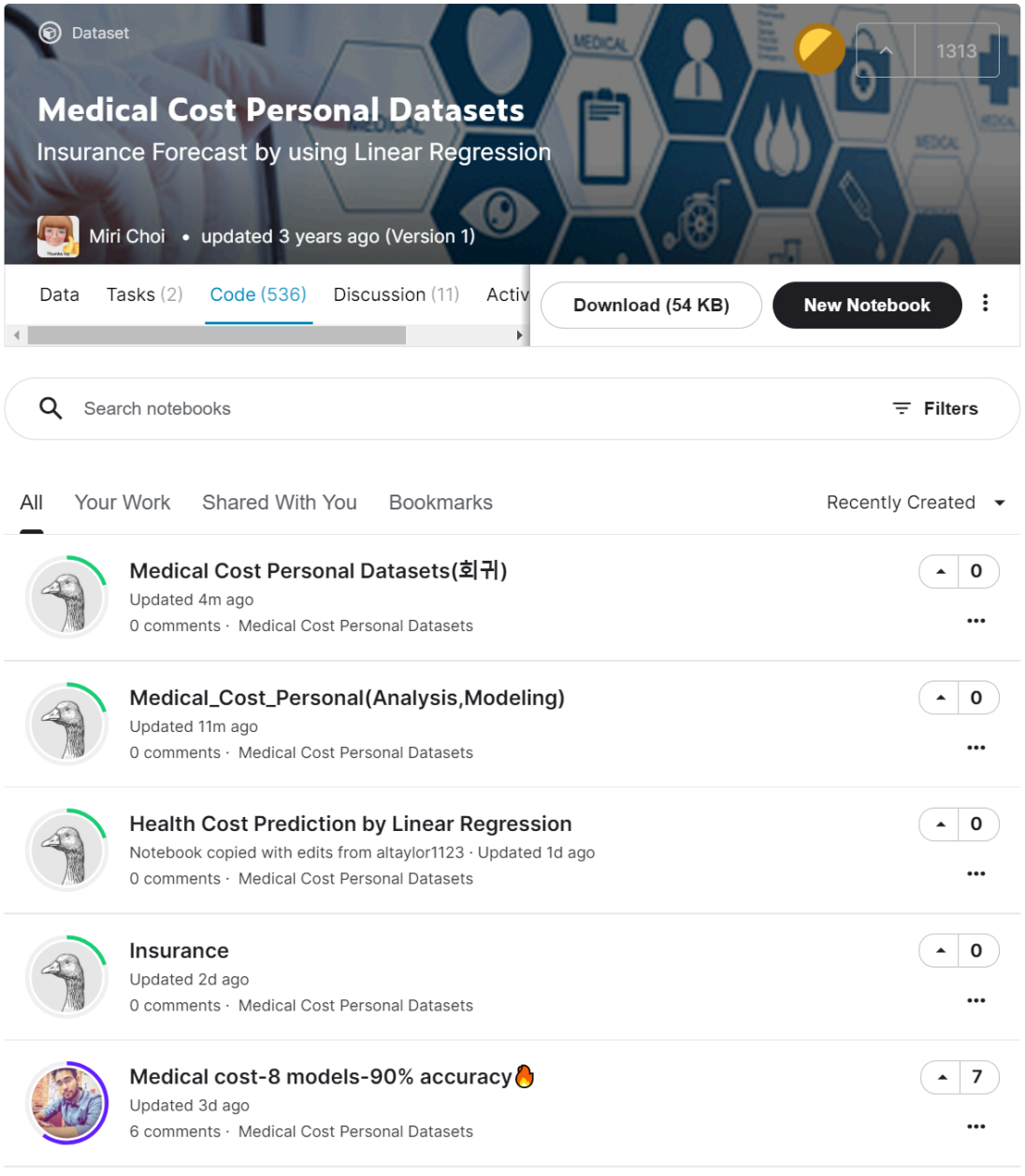



```
<Figure size 720x720 with 0 Axes>
```

```
# 테스트 데이터를 모델에 대입하여 RMSE, MSE, MAE 출력
y_pred = model.predict(x_test)

y_pred_orig = scaler_y.inverse_transform(y_pred)
y_test_orig = scaler_y.inverse_transform(y_test)

from sklearn.metrics import mean_squared_error, mean_absolute_error

RMSE = float(format(np.sqrt(mean_squared_error(y_test_orig, y_pred_orig)),'.3f'))
MSE = mean_squared_error(y_test_orig, y_pred_orig)
MAE = mean_absolute_error(y_test_orig, y_pred_orig)

from sklearn.metrics import r2_score, mean_squared_error
import numpy as np

score = r2_score(y_test_orig, y_pred_orig)
print('R square Score = ', round(score, 3))

print('RMSE =',RMSE,
      '₩nMSE =',MSE,
      '₩nMAE =',MAE)
```

```
R square Score =  0.846
RMSE = 4615.529
MSE = 21303112.0
MAE = 2480.503
```

7.모델 공유 및 활용 방법 구상하기



Kaggle dataset의 code를 공유하여 사람들의 코멘트를 받을 수 있도록 하였습니다.

이후 국가 수준의 빅 데이터 셋을 활용하여 보험료 예측 및 관련 병원을 추천해주는 시스템으로 발전 시키고 싶다. 개인의 건강정보와 보험료를 함께 매치시켜놓은 데이터셋이 없기 때문에 관련 데이터를 찾고 표준화 하는데 많은 시간이 걸릴 것으로 보인다.

활용방안

건강 보험료 예측은 미래 개인형 맞춤 의료서비스 시대에 필요한 기술이며, AI가 질병 진단 및 예측 뿐만 아니라, 질병에 적합한 병원 추천과 식단 및 생활습관 관리등 환자의 일상생활에 밀접하게 접근하여, “더 높은 수준의 헬스케어 개발 및 상용화” 에 도입되어야 한다고 생각한다. 이에 의료보험료 예측 프로그램을 발전시켜, 건강 수치를 기반으로 통계적 수치에 따른 예상 질병 제시와 관련 건강진단 추천등의 기능을 추가 할 수있을 것으로 보인다.

현재 미국의 민간 의료보험은 가격이 높으며, 가입자 또한 기타 선진국에 비해 적은 편이다. 이를 해결하기 위해서는 가입자의 예상 질병및 기타 가능성을 꼼꼼히 조사하여, 본 프로젝트와 같은 보험료 예측 프로그램을 통해 합리적인 가격의 의료보험 시스템을 도입할 필요가 있다고 생각 된다.

한국은 현재 의료보험이 잘 상용화 되었지만, 전국민적 가입유도를 위해 책정한 적은 의료보험료로 적자가 나고 있는 상황이다. 앞으로는 본 프로젝트와 같은 건강 보험료 예측 및 책정 프로그램을 방대한 기존 데이터에 적용시켜서 점진적으로 가입자의 건강상태에 따라 더 합리적인 의료보험료를 책정해야 할 것이다.

8.(중요)프로그램을 제작하며 새로 알게 된점, 추가로 넣고 싶은 기능등을 작성해보자

신영빈

머신러닝을 설계하고 구현하는 과정에서 데이터의 전처리 과정을 통해 모델의 정확도를 높이는 부분에서 교과시간에 배웠던 수학적 통계 지식이 pands plot을 활용하는데 많은 도움이 되었다. Gradient Boosting Algorithm (GBM)에 그리드 서치 기법을 도입하여, 하이퍼 파라미터 튜닝을 통해 RMSE 값을 효과적으로 낮출 수 있음을 알 수 있었다. 또 환자의 건강상태의 지표가 보험료에 미치는 영향을 수동적 클러스터링 결과와 머신러닝 feature의 가중치를 통해 직관적으로 알 수 있었다. 실 보험료 책정에는 훨씬 많은 변수가 존재한다. (콜레스테롤 수치, 과거 의료기록, 당뇨, 빈혈, 백혈구수 등등) 따라서 기초 건강검진의 데이터를 활용한 의료보험료 책정에 대한 프로젝트를 진행해보고 싶다.

윤태경

보건복지부에서 이런 프로그램을 만들고 운영하기 위해 노력하고 있다는 것을 알게 되었다. 그 프로그램과 같으면 의미가 없기 때문에 차별점을 둬야한다. 그러기 위해서는 BMI지수 같은 것 뿐만 아니라 다른 여러가지 질병의 유무, 가족력 등을 추가하여 정확도를 증가시키고 싶다.

이준승

RMSE ,Random Forest, GBM을 사용함에 따라 정확도가 확연히 증가하는 모습을 통해 학습 모델의 중요성에 대해 깨달았고, 상황에 따라 적절한 모델의 사용이 중요하다는 점을 느꼈다. 새로운 모델을 구상해내기 위해서 많은 수학적 지식이 필요할 것 같다는 생각이 들었다.

딥러닝이나 머신러닝을 통해 나타나는 결과도 사실 어느정도 예측 가능한 사실이라는 것이 아쉬웠다. 아직까지는 더 큰 자료를 다루어보지 못해서 그런 것 같다.

체내 건강 상태를 알 수 있는 다양한 화학 반응들이 있다. (예를 들면, 당뇨 진단을 위한 체액 내 포도당 농도에 따라 색변화가 나타나는 반응 / 흡연자의 폐 상태 변화 ...) 이러한 시각적으로 관찰할 수 있는 반응들을 통해 건강 상태를 진단할 수 있는 머신러닝을 구현해 넣어보고 싶다.