보험료 예측하기 - 선형회귀

선형회귀는 가장 기초적인 머신러닝 모델이다. 여러가지 데이터를 활용하여 연속형 변수인 목표변수를 예측해 내는 것이 목적이다. 예를 들어 몸무게, 나이, 성별등을 데이터로 활용하여 키와 같은 연속형 변수를 예측하는 것이다.

장점

- 모델이 간단하기 때문에 구현과 해석이 쉽다
- 같은 이유로 모델링하는 데 오랜 시간이 걸리지 않는다.

단점

- 최신 알고리즘에 비해 예측력이 떨어진다.
- 독립변수와 예측변수의 선형관계를 전제로 하기 때문에, 이런 전제에서 벗어나는 데이터에서는 좋은 예측을 보여주기 어렵다.

4.2 라이브러리 및 데이터 불러오기

```
import pandas as pd

file_url ='http://media.githubusercontent.com/media/musthave-ML10/data_source
          /main/insurance.csv'
data = pd.read_csv(file_url) #데이터셋 읽기
```

4.3 데이터 확인하기

```
data.head() #상위 5줄 출력
```

- 연속형 변수

연속형 변수는 나이, 키와 깉이 연속적으로 이어지는 변수이다. (사칙연산가능, 평균가능)

- 범주형 변수

범주형 변수는 숫자가 아닌 범주로 구성된 변수이다. (계절, 성별)

```
data.info() # 컬럼 정보를 출력한다.
```

```
round(data, describe(), 2) #소수점 2째자리까지만 표시해 통계 정보 출력
```

	age	sex	bmi	children	smoker	charges
count	1338.00	1338.00	1338.00	1338.00	1338.0	1338.00
mean	39.21	0.51	30.66	1.09	0.2	13270.42
std	14.05	0.50	6.10	1.21	0.4	12110.01
min	18.00	0.00	15.96	0.00	0.0	1121.87
25%	27.00	0.00	26.30	0.00	0.0	4740.29
50%	39.00	1.00	30.40	1.00	0.0	9382.03
75%	51.00	1.00	34.69	2.00	0.0	16639.91
max	64.00	1.00	53.13	5.00	1.0	63770.43

4.4 전처리 : 학습셋과 시험셋 나누기

데이터를 나누는 작업은 크게 2가지 차원으로 진행된다. 첫째는 종속변수와 독립변수 분리이다. 둘째는 학습용 데이터셋인 학습셋과 평가용 시험셋을 나누는 것이다. 총 4개의 데이터 셋으로 나눈다.

4.4.1 변수와 데이터셋을 나누는 이유

지도학습에 속하는 머신러닝 모델은 독립변수를 통하여 종속변수를 예측하는 것이므로, 모델링할 때 어떤 변수가 종속변수인지 명확히 알려주어야 한다.

4.4.2 데이터셋 나누기

우선 종속변수와 독립변수를 나눈 후에 학습 셋과 시험 셋으로 나누어야한다. 우선 독립변수를 X, 종속변수를 y로 나눈다.

```
X = data[['age', 'sex', 'bmi', 'children', 'smoker']] # 독립변수 : 데이터 프레임이기 때문에 대문자로 쓴다.
y = data['charges'] # 종속변수 : 변수가 하나인 '시리즈'이기 때문에 소문자로 쓴다.
```

```
from sklearn.model_selection import train_test_split #사이킷런 임포트
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state=100) #데이터셋 분할
#train_test_split 함수가 데이터셋 4개를 결과물로 내보낸다. 이를 받아줄 변수 4개가 필요하다. 시험 셋의 크기는 20%,8:2 random_state를 100으로 설정 랜덤 샘플링은
#그래서 매번 실행할 때마다 train_set과 test_set에 있는 데이터가 달라진다.
```

독립변수/종속변수, 그리고 학습셋/시험셋 조합으로 총 4개 데이터셋(X_train, X_test, y_train, y_test)이 나옴.

4.5 모델링

모델링은 머신러닝 알고리즘으로 모델을 학습시키는 과정이다. 결과물이 머신러닝 모델이 된다. 독립변수와 종속변수를 fit()함수에 인수로 주어 학습한다.이번 데이터셋에는 선형 회귀 알고리즘을 사용한다.

```
from sklearn.linear_model import LinearRegression
```

이제 모델을 만들어주어야 한다. model이라는 이름의 객체에 선형 회귀 속성을 부여한다.

```
model = LinearRegression()
```

객체를 생성했으니 객체를 사용해 학습한다.

```
model.fit(X_train, y_train)# (독립변수, 종속변수)
```

'학습시킨다'함은 데이터를 모델 안에 넣어서 독립변수와 종속변수 간의 관계들을 분석해 새로운 데이터를 예측할 수 있는 상태로 만드는 것이다.

4.6 모델을 활용해 예측하기

예측하는 실습을 하자. 원래는 예측 및 평가에서 학습셋과 시험셋을 각각 사용해 오버피팅 문제를 확인하지만, 이번 장에서는 간단하게 시험셋만 가지고 예측/평가를 하자.

```
pred = model.predict(X_test) #predict로 예측을 할 수 있고, 괄호 안에는 예측 대상을 넣어주면 된다. 목표 변수가 예측 대상이다.
```

predict()함수로 예측을 할 수 있으며 괄호 안에는 예측 대상을 넣어주면 된다. 여기에 들어가는 데이터는 당연히 목표변수가 포함되어서는 안된다. 정답을 알려주는 꼴이 된다. 독립변수들을 가진 데이터를 넣어주어야 한다.

4.7 예측 모델 평가하기

'테이블로 평가하기, 그래프로 평가하기, 통계적인 방법으로 평가하기'가 있다.

4.7.1 테이블로 평가하기

```
#테이블로 평가하기
comparison = pd.DataFrame({'actual' : y_test, 'pred' : pred})
comparison
```

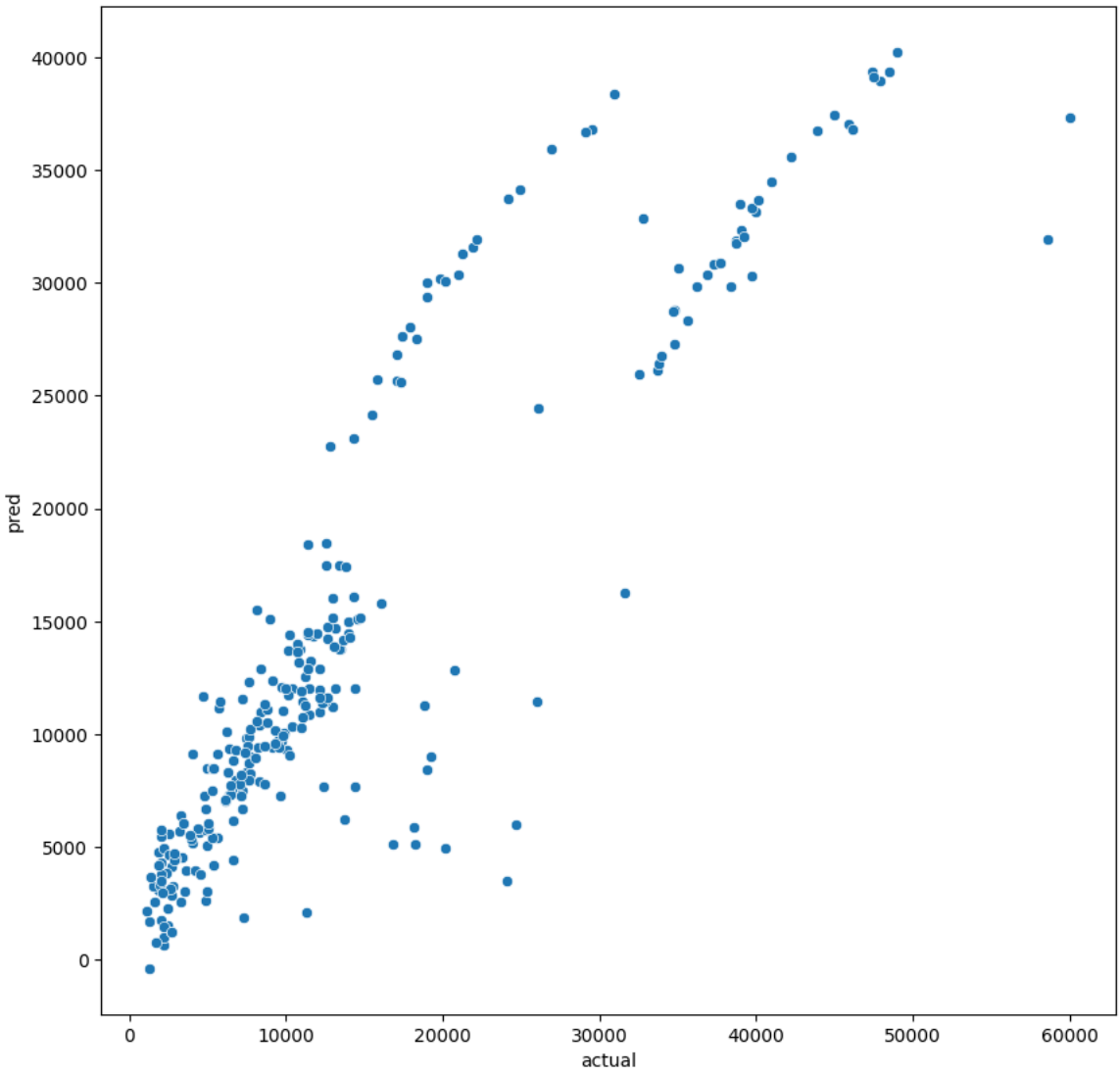
	actual	pred
12	1826.84300	4765.249466
306	20177.67113	4957.730865
318	7421.19455	8298.988153
815	1877.92940	3078.811868
157	15518.18025	24165.956542
...	...	...
713	1984.45330	5776.764928
1282	14283.45940	23102.847340
531	14043.47670	14280.732585
537	8825.08600	10527.417291
1015	12124.99240	11638.260006

수많은 데이터를 눈으로 보기 어렵다. 산점도 그래프를 그려보자.

4.7.2 그래프로 평가하기

```
#그래프로 평가하기
import matplotlib.pyplot as plt #matplotlib를 plt로 줄여서 사용한다.
import seaborn as sns #seaborn를 sns로 줄여서 사용한다.
```

```
plt.figure(figsize=(10, 10)) #그래프의 크기를 정의한다.
sns.scatterplot(x = 'actual', y = 'pred', data = comparison) # scatter함수를 이용해서 산점도 그래프를 만들었다.
```



그래프로 그렸더니 테이블로 일일히 확인할 때보다 편한 것을 볼 수 있다. 어디까지나 직관적으로 예측력을 확인할 뿐이지 객관적인 기준이 되지는 않는다.

4.7.3 통계적인 방법으로 평가하기 : RMSE

이번엔 통계적인 방법으로 접근하자.

#통계적인 방법으로 평가하기 : RMSE.. MSE : 절댓값 차이를 이용하는 방법이다. (평균 절대 오차) 앞에 mean이 붙은 이유는 차이의 합을 총 개수로 나누어 평균을 내기 때문이다.
#MAE(평균 절대 오차) : 절대값 차이를 이용하는 방법. 실젯값과 예측값의 사이의 오차에 절댓값을 씌운 뒤 이에 대한 평균을 계산한다. 제곱X

#MSE(평균 제곱 오차) : 제곱 차이를 활용하는 방법. 실젯값과 예측값의 사이의 오차를 제곱한 뒤 이에 대한 평균을 계산한다.

#RMSE(루트 평균 제곱 오차) : MSE에 루트를 씌운 값으로 가장 일반적으로 사용된다. 값이 작을수록 좋은 지표이다. 이 지표가 연속형 변수를 예측할 때 가장 일반적으로 쓰이는 평가지표이다.

이제 RMSE로 구해보자. 우선 MSE를 구하는 함수를 쓰고 루트를 씌워주자.

```
from sklearn.metrics import mean_squared_error #MSE 라이브러리 import
mean_squared_error(y_test, pred) ** 0.5 #RMSE 계산 실행.
```

** 0.5는 루트이다. 파이썬에서는 **가 제곱이므로 0.5를 제곱하면 루트를 씌운 값을 계산한다.

```
mean_squared_error(y_test, pred, squared = False)
```

#RMSE는 근본적으로 에러에 대한 합을 계산한 것이기 때문에, 작을 수록 예측력이 좋은 것이다.
#앞으로 다양한 알고리즘과 활용법을 배우고, 같은 데이터에 여러가지 모델링을 해보고, 그 중 어떤 모델이 가장 뛰어난 예측력을 보이는지를 판단할 때 RMSE가 가장 낮은 모델을 선택하면 된다.

```
model.score(X_train, y_train)
>>> 0.7368280129839127
```

약 74% 정확도를 가지고 있다.

4.8 이해하기 : 선형회귀

선형회귀는 독립변수와 종속변수 간에 선형 관계가 있음을 가정하여 최적의 선을 그려서 예측하는 방법이다.

머신러닝에서는 손실함수를 최소화하는 선을 찾아서 모델을 만들어낸다. MSE, RMSE등이 손실함수이다.

새로운 함수와 라이브러리

1. round() : 반올림
2. pandas.DataFrame() : 데이터를 DataFrame 형태로 변환
3. pandas.DataFrame.head() : 데이터의 초반부 호출(5줄)
4. pandas.DataFrame.info() : 변수에 대한 결측치, 데이터 타입 정보 확인
5. pandas.DataFrame.describe() : 데이터프레임의 통계적 요약 확인
6. sklearn.model_selection.train_test_split() : 훈련셋과 시험셋 분류
7. sklearn.metrics.mean_squared_error(): 평균 제곱 오차 계산
8. 모델.fit() : 모델 학습
9. 모델.predict: 학습된 모델로 예측
10. matplotlib.pyplot.figure() : 새로운 도표를 만들거나 기존 도표를 활성화
11. seaboard.scatterplot() : 산점도 그래프 생성