기상청

5. 분석 알고리즘 맛보기

Python을 활용한 분석 기초 가이드 북

기상청

Ⅴ. 분석 알고리즘 예제

1. 알고리즘 선정
2. 예측: 단순선형회귀
3. 분류: k-최근접 이웃

1. 알고리즘 선정

분석을 위한 기계학습 알고리즘은 크게 지도학습(Supervised learning)과 비지도학습(Unsupervised learning)으로 구분됩니다. 본 장에서는 분석 알고리즘 맛보기라는 취지에 맞게 가장 기본이 되는 지도학습방법 중 회귀와 분류에 대해 알아봅니다. 하지만, 회귀와 분류만 하더라도 다양한 알고리즘이 존재하므로 분석 목적과 데이터에 맞게 알고리즘을 선택해야 합니다.

● 분석 목적, 데이터 고려

- 목적 값(Target value)을 예측하거나 예견하려고 한다면 지도학습방법을 살펴보고, 그렇지 않다면 비지도학습방법을 살펴봅니다.
- 지도학습방법에서는 목적 값이 수치(Number)로 나타나길 원한다면 회귀를 살펴보고, 'Y/N', '그룹1/그룹2/그룹3'등과 같이 이산적인 값이라면 분류를 살펴봅니다.
- 비지도학습방법에서는 가지고 있는 데이터가 각각의 무리에 알맞은 어느 정도의 수치인지로 평가하려 한다면 밀도 추정 알고리즘을 살펴보고, 어떤 이산적인 무리에 알맞은지 알아보고자 한다면 군집화 알고리즘을 살펴봅니다.
- 여기에서는 아주 보편적인 알고리즘 선정 예를 들었습니다. 하지만, 간혹 위와 같은 규칙 외에 다른 알고리즘을 사용하기도 합니다.
- 추가적으로, 성공적인 분석과 알고리즘 선택을 위해서는 보유한 데이터의 특성을 파악하는 일이 반드시 필요합니다. 예를 들어, 속성이 명목형/연속형인지, Null값이 존재하는지, Outlier가 있는지 등에 따라서 적용 가능한 알고리즘의 폭을 줄일 수 있습니다.

2. 예측: 단순선형회귀(1/3)

우리가 회귀분석을 하는 이유는 수치형 목적 값을 예측하기 위해서입니다. 선형 회귀의 장점은 결과를 해석하기가 쉽고 계산 비용이 적다는 점이며, 단점으로는 비선형 데이터를 모델링하기에 적합하지 않다는 점이 있습니다. 본 장에서는 단순선형회귀에 대하여 알아보도록 하겠습니다.

● 회귀 방정식과 회귀 가중치

- 전운량에 따른 일조를 예측할 때 독립변수인 전운량을 $\boldsymbol{x_1}$ 로, 종속변수인 일조를 $\boldsymbol{y}$ 로 하였을 때, 아래와 같은 방정식을 생각해 볼 수 있습니다.

$$\boldsymbol{y = \beta_0 + \beta_1 * x_1 + \varepsilon}$$
$$\boldsymbol{\hat{y} = \beta_0 + \beta_1 * x_1}$$

$\boldsymbol{y}$: 종속변수 관측치
$\boldsymbol{\hat{y}}$: 종속변수 추정치
$\boldsymbol{x}$: 독립변수 관측치
$\boldsymbol{\varepsilon}$: 오차

- 위와 같은 식을 회귀 방정식이라 부르고, 여기에서 $\boldsymbol{\beta_0}$ 를 절편, $\boldsymbol{\beta_1}$를 회귀 가중치라 부릅니다.
- 이렇게 회귀 가중치를 찾는 과정을 회귀분석이라고 하는데, 일단 모수(절편, 회귀 가중치)를 찾게 되면 새로운 독립변수($\boldsymbol{x_1}$)가 주어졌을 때 종속변수 값($\boldsymbol{\hat{y}}$)을 예측해내는 일은 쉽습니다.

● 전운량에 따른 일조(시간) 예측

- 전운량이란 대기 중 구름양을 측정해 0~10 사이의 실수 값으로 표현된 수치를 말하며, 일조란 하루 중 햇볕이 구름이나 안개 따위에 가려지지 아니하고 실제로 내리쬐는 시간을 말합니다.
- 우리는 지금부터 기상청에서 제공하는 종관기상관측장비(ASOS) 데이터를 활용해 일평균전운량(CA_TOT)에 따른 일조(SS_DAY)의 변화를 분석해볼 것입니다.

2. 예측: 단순선형회귀(2/3)

● Python을 활용한 분석

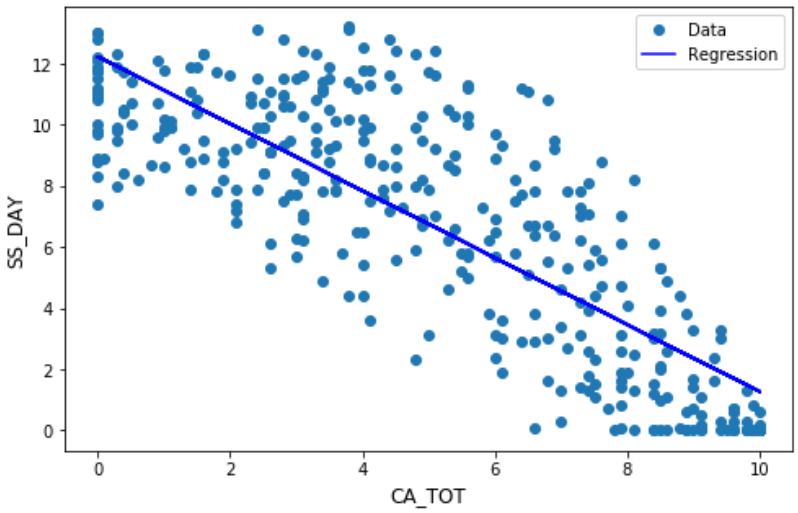
- 청주지역에서 2015년에 일 단위로 측정된 ASOS 예제 데이터를 활용해 일평균전운량에 따른 하루의 일조 합계 시간을 예측해 봅시다.
- 설치된 Spyder를 실행해 스크립트 편집 창에 아래의 코드를 붙여넣고 전체 실행시킵니다.

```
### 패키지 불러오기
import pandas as pd
import statsmodels.formula.api as sm
import matplotlib.pyplot as plt

### 예제 데이터 로드
mydata = pd.read_csv('E:/data_regression.csv')    # 데이터 위치 지정
print(mydata.head())                              # 로드 데이터 확인

### 데이터 정제
mydata.loc[mydata['SS_DAY'] == -9]                # 20150520 일조합 결측치(-9) 확인
mydata.loc[(mydata['SS_DAY'] != -9) & (mydata['CA_TOT'] == 0.5)]  # 20150520을 제외한
일평균전운량이 0.5였던 날들 확인
mydata.loc[mydata['SS_DAY'] == -9, 'SS_DAY'] = 10.7  # 일조합 결측치에 평균값 10.7 입력

### 단순선형회귀 모델링
result = sm.ols(formula='SS_DAY ~ CA_TOT', data=mydata).fit()

### 회귀분석 결과 요약
print(result.summary())

### 세부 분석 결과 확인
print('< Parameters > ₩n', result.params)              # 회귀계수 출력
print('< Prob (Parameters) > ₩n', result.pvalues)      # 회귀계수에 대한 P-value 출력
print('< Adj. R-squaured > ₩n', result.rsquared_adj) # 조정된 R-squared 출력
print('< Prob (F-statistic) > ₩n',  result.f_pvalue)      # 모형의 적합도 출력

### 그래프 그리기
fig, ax = plt.subplots(figsize=(8, 5))
plt.ylabel('SS_DAY', size=12)
plt.xlabel('CA_TOT', size=12)
ax.plot(mydata.CA_TOT.values, mydata.SS_DAY.values, 'o', label='Data')
ax.plot(mydata.CA_TOT.values, result.fittedvalues, 'b-', label='Regression')
ax.legend(loc='best')
```

2. 예측: 단순선형회귀(3/3)

● 분석결과

```
< Parameters >
 Intercept    12.221711
CA_TOT        -1.095343
dtype: float64

< Prob (Parameters) >
 Intercept    1.545615e-168
CA_TOT        2.877431e-90
dtype: float64

< Adj. R-squaured >
 0.672674117436

< Prob (F-statistic) >
 2.87743055453e-90
```



● 분석결과 해석

- 〈 Parameters 〉에는 회귀모형의 절편과 기울기가 제시되어 있습니다. 위 결과를 식으로 표현하면 다음과 같습니다.

$$\hat{Y} = 12.22 - 1.90X$$

$\hat{Y}$: 일조합(시간), X: 일평균전운량

- 〈 Prob (Parameters) 〉에는 각 모수(절편, 기울기)의 통계적 유의확률이 제시되어 있으며, 일반적으로 0.05 이하일 경우, 모수가 유의미한 것으로 해석합니다.
- 〈 Adj. R-squared 〉에는 회귀모형의 결정계수가 제시되어 있습니다. 결정계수는 모형의 설명력을 뜻하며, 위 모형의 설명력은 67.27%입니다.
- 〈 Prob (F-statistic) 〉에는 모형의 F검정의 유의확률이 제시되어 있습니다. 이 유의확률 역시 0.05 이하일 경우 모형이 유의미한 것을 의미합니다.

3. 분류: k-최근접 이웃(1/4)

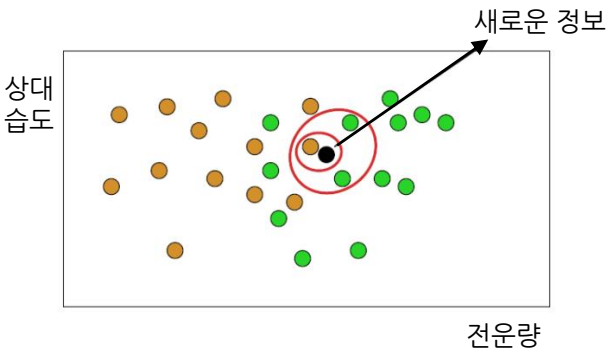
k-최근접 이웃(k-NN; k-nearest Neighbors) 분류기는 목적 대상(종속변수)의 범주를 활용 변수(독립변수)의 유사성에 기반해 범주를 지정하는 분류 알고리즘입니다. 이 알고리즘의 장점은 이상치(outlier)에 대해 둔감하고 데이터에 대한 가정이 없으며, 높은 정확도를 가진다는 점입니다. 단점으로는 계산 비용이 크고 많은 메모리 공간을 요구한다는 점입니다. 예제를 통해 k-최근접 이웃 알고리즘을 적용시켜 분석해봅시다.

● 전운량과 습도를 이용해 비(혹은 눈)이 온 날 예측

- 전운량이란 대기에 구름이 형성된 양을 측정해 0~10 사이의 실수 값으로 표현된 수치를 말하며, 습도란 평균 상대습도(%)로 0~100 사이의 실수 값으로 표현된 수치를 말합니다.
- 본 예제에서 비(혹은 눈)이 온 날은 일 강수량(mm)이 0보다 큰 모든 날로 정의합니다.
- 우리는 지금부터 기상청에서 제공하는 종관기상관측장비(ASOS) 데이터를 활용해 일평균전운량(CA_TOT)과 일평균상대습도(HM_AVG)를 이용해 비(혹은 눈)이 온 날(일 강수량(RN_DAY) > 0 mm)을 예측해볼 것입니다.

● k-최근접 이웃

- k-최근접 이웃은 새로운 데이터가 주어졌을 때 학습한 데이터 가운데 가장 가까운 k개 이웃 정보를 이용하여 새로운 종속변수가 어떤 결과 값을 갖을지 예측하는 것입니다.
- 선형 회귀분석과 같은 예측모형처럼 종속변수의 예측값을 추정하는 것이 아니라 분류하는 정보에 기반에 종속변수가 타겟에 속할 확률을 산출합니다.
- 전운량과 상대습도에 따른 비 온 날을 분류하는 것을 예로 들어보면,



새로운 상대습도와 전운량 정보가 주어졌을 때, 기존의 분류 학습 데이터와 이웃한 1개의 정보만 반영하면 주황색으로 예측할 수 있습니다. 그러나 이웃한 3개의 정보를 반영하게 되면 주황색일 확률은 33.3%, 녹색일 확률은 66.6% 입니다.

3. 분류: k-최근접 이웃(2/4)

● Python을 활용한 분석

- 서울지역에서 2015년에 일 단위로 측정된 ASOS 예제 데이터를 활용해 일평균전운량과 평균 상대습도에 따른 비(또는 눈)이 온 날을 예측해 봅시다.
- 설치된 Spyder를 실행해 스크립트 편집 창에 아래의 코드를 붙여넣고 전체 실행시킵니다.

```
### 패키지 불러오기
import numpy as np
from sklearn import neighbors # sklearn 은 scikit-learn 의 줄임말 입니다. 아나콘다에서 패키지 설치 시 scikit-learn 으로 검색합니다.
import pandas as pd
from matplotlib import colors as c
from sklearn.metrics import classification_report


### 예제 데이터 로드
mydata = pd.read_csv('E:/data_kNN.csv')    # 데이터 위치에서 데이터 불러오기
print(mydata.head())                                    # 로드 데이터 확인

### 데이터 정제
mydata.loc[mydata['RN_DAY'] <= 0, 'RN_DAY'] = 0    # 강수량 0mm 이하(결측 포함)에 0 입력
mydata.loc[mydata['RN_DAY'] > 0, 'RN_DAY'] = 1     # 강수량 0mm 초과에 1 입력
mydata['RN_DAY'] = mydata['RN_DAY'].astype('category') # 강수량 변수 범주형 변환
mydata.loc[mydata['CA_TOT'] == -9, 'CA_TOT'] = 0     # 전운량 결측치(-9)에 0 입력
mydata.loc[mydata['HM_AVG'] == -9, 'HM_AVG'] = 0     # 상대습도 결측치(-9)에 0 입력

### 종속변수, 독립변수 설정
X  =  mydata.iloc[:,2:4]    # 세번째 컬럼(CA_TOT)와 네번째 컬럼(HM_AVG)을 독립변수로 설정
y  =  data['RN_DAY']   # 이분형으로 변환한 강수량 변수를 종속변수로 설정

### k-최근접 이웃 모델링
kNN = neighbors.KNeighborsClassifier()
kNN.fit(X, y)

Z = kNN.predict(X) # 예측값 산출

### k-최근접 결과
target_names  = ['0', '1']
print(classification_report(y, Z, target_names=target_names))
```

3. 분류: k-최근접 이웃(3/4)

● Python을 활용한 분석

- 서울지역에서 2015년에 일 단위로 측정된 ASOS 예제 데이터를 활용해 일평균전운량과 평균 상대습도에 따른 비(또는 눈)이 온 날을 예측해 봅시다.
- 설치된 Spyder를 실행해 스크립트 편집 창에 아래의 코드를 붙여넣고 전체 실행시킵니다.

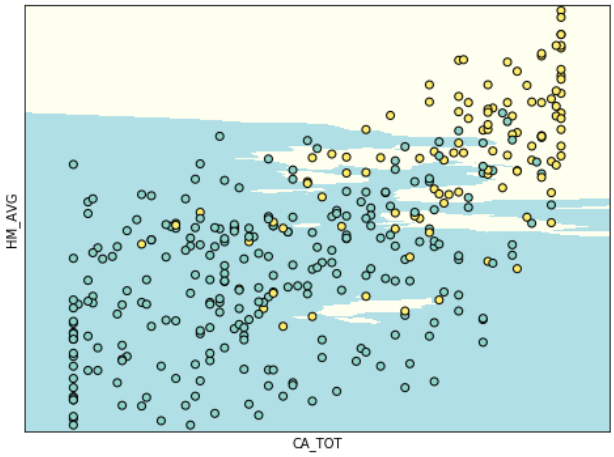
```
# 도표 눈금(의사결정 경계를 얼마나 촘촘하게 할 것인지) 설정
plot_step = 0.02

# 경계 그리기
x_min, x_max = X['CA_TOT'].min() - 1, X['CA_TOT'].max() + 1
y_min, y_max = X['HM_AVG'].min() - 1, X['HM_AVG'].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, plot_step),np.arange(y_min, y_max, plot_step))
Z = kNN.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)

# kNN의 결과를 반영해 색상 입히기
pl.figure(1, figsize=(8, 6))
pl.set_cmap(pl.cm.Paired)
cMap = c.ListedColormap([ ' powderblue ' , ' ivory ' ])
pl.pcolormesh(xx, yy, Z,cmap=cMap)

# 독립변수 시각화
plt.scatter(X['CA_TOT'], X['HM_AVG'], c=y, cmap=plt.cm.Set3, edgecolor='k')
pl.xlabel('CA_TOT')
pl.ylabel('HM_AVG')

# 축이름 및 도표명
pl.xlim(xx.min(), xx.max())
pl.ylim(yy.min(), yy.max())
pl.xticks(())
pl.yticks(())
plt.title('Decision Boundary')
```

3. 분류: k-최근접 이웃(4/4)

● 분석 결과

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=1, n_neighbors=5, p=2,
           weights='uniform')
```

	precision	recall	f1-score	support
0	0.96	0.99	0.97	254
1	0.97	0.90	0.93	111
avg / total	0.96	0.96	0.96	365



● 분석결과 해석

- kNN.fit(X, y) 를 실행하면 모형의 옵션 구성 결과가 출력됩니다. 모형의 옵션은 값을 지정하여 고정할 수 있습니다. n_neighbors 옵션 예시는 아래와 같습니다(k=3일 때).

```
kNN = neighbors.KNeighborsClassifier(n_neighbors=3)
kNN.fit(X,y)
```

- 분석 결과표에서 정밀도(precision)란 타겟이 1일 경우, 모형에서 예측한 발생(1) 사건 중 실제 발생(1) 사건의 비율을 뜻합니다. 위 모형의 평균 정밀도는 96%입니다.
- 분석 결과표에서 재현율(recall)이란 타겟이 1일 경우, 실제 발생(1) 사건 중 모형이 예측한 발생(1) 사건의 비율을 뜻합니다. 위 모형의 평균 재현율은 96% 입니다.
- 분석 결과표에서 f1 점수는 정밀도와 재현율 두 지수를 종합해 계산한 결과로 그 식은 아래와 같습니다.

$$F1 = 2 \times \frac{\text{정밀도} \times \text{재현율}}{\text{정밀도} + \text{재현율}}$$

f1 점수는 1에 가까울 수록 좋으며 위의 f1 점수는 0.96 입니다.

- 미래의 사건을 예측하는 모형을 구축하기 위해선 독립변수와 종속변수 간 시간 간격을 조정, 변수 추가 등 여러 데이터 분석 방법을 적용 할 수 있습니다.

본 문서의 내용은 기상청의 날씨마루(http://big.kma.go.kr) 내
Python 기초 교육 자료 입니다.