### 컬럼 type 확인

train.info()

### 요약정보 확인

train.describe()

### 결측치 확인

train.isnull().sum()

### 결측치 제거

df.dropna()

### 결측치 변경

df.fillna(value)

### 사분위

- 1사분위

print(df['f1'].quantile(.25))

- 3사분위

print(df['f1'].quantile(.75))

### sort

df = df.sort\_values('순전입 학생', ascending=False)

### 해당 열의 고유값 카운트(고유한 행의 수)

df['Roof'].value\_counts()

### iloc

- 첫번째 행

df.iloc[0]

- 첫번째 열

df.iloc[:,0]

- 첫번째 행 마지막 열

df.iloc[0, -1]

- 응용

df['총범죄'] = df.iloc[:,1:-1].sum(axis=1)

### loc

df.loc['row name', 'cloumn name']

### GroupBy

result = df.groupby('소방서')['diff'].mean()

### 스케일링

from sklearn.preprocessing import minmax\_scale

data["qsec"] = minmax\_scale(data["qsec"])

### 레이블인코딩

from sklearn.preprocessing import LabelEncoder

cols1 = ['주구매상품', '주구매지점']

le = LabelEncoder()

for col in cols1:

train[col] = le.fit\_transform(train[col])

test[col] = le.fit\_transform(test[col])

### 원핫 인코딩

train = pd.get\_dummies(train)

test = pd.get\_dummies(test)

### 트레인데이터테스트데이터 나누기(검증데이터 분리)

from sklearn.model\_selection import train\_test\_split

X\_tr, X\_val, y\_tr, y\_val = train\_test\_split(

train, target, test\_size=0.2, random\_state=0

)

### 모델학습

- 분류

from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier()

model.fit(X\_tr, y\_tr)

pred = model.predict\_proba(X\_val)

- 회귀

from sklearn.ensemble import RandomForestRegressor

rf = RandomForestRegressor()

rf.fit(X\_tr, y\_tr)

pred = rf.predict(X\_val)

### 평가

- import roc\_auc\_score

from sklearn.metrics import roc\_auc\_score

print(roc\_auc\_score(y\_val, pred[:, 1]))

- f1\_score

from sklearn.metrics import f1\_score

f1\_score(y\_val, pred, average='macro')

### csv생성

result = pd.DataFrame({

'pred': pred

})

result.to\_csv('result.csv', index=False)

### 카이제곱

from scipy import stats

stats.chisquare(관찰 빈도 리스트, 기대 빈도 리스트)

### 회귀/분류 구분

- 평가지표

- 회귀 : MAE, MSE, RMSE, RMSLE, R2... (거의 다 E로 끝남, predict 사용)

- 분류 : [Accuracy, Precison, Recall, F1](predict 사용), ROC\_AUC(predict\_proba 사용)...

### 주요 검정

- 가설검정, 분산분석, 범주형 데이터 검정, 회귀모형 등

- 가설검정

from scipy import stats

- 단일표본검정 : 표본이 하나

stats.ttest\_1samp(df['무게'], 120)

alternative는 가설에서 df가 크다고 하면 greater 작다고 하면 less

- 대응표본검정 : 표본이 둘

stats.ttest\_rel(df['before'], df['after'], alternative='less')

alternative는 앞의 df 기준으로 대립가설 식이 0보다 작으면(마이너스가 나와야하면) less 0보다크면 (플러스면) greater

- 독립표본검정 : 표본이 둘, 길이가 다르다

stats.ttest\_ind(A, B)

stats.ttest\_ind(A, B, equal\_var=False) 분산이 다를 경우

stats.ttest\_ind(A, B, alternative='less') 대립가설 B그룹 시험 평균 점수가 더 높다

stats.ttest\_ind(A, B, alternative='greater') 대립가설 A그룹 시험 평균 점수가 더 높다

- 범주형 데이터 검정(카이제곱검정)

- 적합도 검정

#관찰

ob = [150, 120, 30]

#기대

ex = [0.5\*300, 0.35\*300, 0.15\*300]

from scipy import stats

stats.chisquare(ob, ex)

- 독립성 검정

- 교차표

df = pd.DataFrame(

{

'합격': [80, 90],

'불합격': [20, 10]

}, index= ['R', 'Py']

)

from scipy import stats

stats.chi2\_contingency(df)

- 로우데이터

data = {

'언어': ['R']\*100 + ['Python']\*100,

'합격여부': ['합격']\*80 + ['불합격']\*20 + ['합격']\*90 + ['불합격']\*10

}

df = pd.DataFrame(data)

df = pd.crosstab(df['언어'], df['합격여부'])

stats.chi2\_contingency(df)

- 회귀분석

- 상관관계

- 상관계수

df.corr() //피어슨

키 몸무게

키 1.000000 0.901012

몸무게 0.901012 1.000000

print(df['키'].corr(df['몸무게']))

print(df['몸무게'].corr(df['키']))

print(df.corr(method='spearman')) //스피어맨

print(df.corr(method='kendall')) //켄달타우

- t검정

from scipy import stats

print(stats.pearsonr(df['몸무게'],df['키'])) //피어슨, 순서상관X

print(stats.spearmanr(df['몸무게'],df['키'])) //스피어맨, 순서상관X

print(stats.kendalltau(df['몸무게'],df['키'])) //켄달타우

- 단순선형회귀분석

- 모델 학습 summary 출력

from statsmodels.formula.api import ols

model = ols('키 ~ 몸무게', data=df).fit() //R스타일

print(model.summary())

- OLS Regression Results

R-squared //결정계수, Adj. R-squared //수정결정계수, coef //회귀계수(기울기),

Intercept //절편, t //검정통계량, P //P-value, [0.025 0.975] //95%신뢰구간

- 결정계수 : model.rsquared

- 기울기(회귀계수) : model.params['몸무게']

- 절편(회귀계수) : model.params['Intercept']

- 몸무게의 회귀계수가 통계적으로 유의한지 pvalue : model.pvalues['몸무게']

- 잔차 제곱합

잔차 = 관측(실제)값 - 예측값

df['잔차'] = df['키'] - model.predict(df['몸무게'])

sum(df['잔차'] \*\* 2)

- MSE

(df['잔차'] \*\* 2).mean()

- 몸무게가 50일 때 예측키에 대한 신뢰구간, 예측구간

newdata = pd.DataFrame({'몸무게':[50]})

pred = model.get\_prediction(newdata)

pred.summary\_frame(alpha=0.05)

- 다중 선형 회귀 분석

- 모델 학습 summary 출력 (종속변수: 매출액, 독립변수: 광고비, 플랫폼)

from statsmodels.formula.api import ols

model = ols('매출액 ~ 광고비 + 플랫폼', data=df).fit()

print(model.summary())

- 결정계수(R-squared)

model.rsquared

- 회귀계수(광고비)

model.params['광고비']

- 회귀계수(플랫폼)

model.params['플랫폼']

- 회귀계수(절편)

model.params['Intercept']

- 광고비의 회귀계수가 통계적으로 유의한지 pvalue

model.pvalues['광고비']

- 플랫폼의 회귀계수가 통계적으로 유의한지 pvalue

model.pvalues['플랫폼']

- 광고비 50, 플랫폼 20일 때 매출액 예측

newdata = pd.DataFrame({

'광고비':[50],

'플랫폼':[20]

})

model.predict(newdata)

- 잔차 제곱합

(model.resid \*\* 2).sum()

- MSE

(model.resid \*\* 2).mean()

- 광고비, 플랫폼 회귀계수의 95% 신뢰구간

|광고비: 0.117 3.522|

|플랫폼: 2.912 8.945|

model.conf\_int(alpha=0.05)

- 범주형 변수 : ols가 자동으로 원핫인코딩

- 분산분석

- 일원분산분석

from scipy import stats

stats.f\_oneway(df['A'], df['B'], df['C'], df['D'])

- 이원분산분석

- anova 테이블

from statsmodels.formula.api import ols

from statsmodels.stats.anova import anova\_lm

model = ols('토마토수 ~ 종자 + 비료 + 종자:비료', data=df).fit()

anova\_lm(model)

- 범주형 데이터 처리

from statsmodels.formula.api import ols

from statsmodels.stats.anova import anova\_lm

model = ols('토마토수 ~ C(종자) + C(비료) + C(종자):C(비료)', data=df).fit()

anova\_lm(model)