

데이터 분석 기초

빅 데이터 아카데미





데이터 분석 기초

1. Scikit-Learn 라이브러리
2. 데이터 분할
3. 교차 검증

Scikit-Learn 라이브러리

- 머신러닝 알고리즘을 구현한 오픈소스 라이브러리 중 가장 유명한 라이브러리 중 하나
- 일관되고 간결한 API가 강점이며, 문서화가 잘되어 있음
- 알고리즘은 파이썬 클래스로 구현되고, 데이터 셋은 Numpy 배열, Pandas DataFrame 및 SciPy 희소행렬 등 사용가능



scikit-learn
Machine Learning in Python

- Simple and efficient tools for data mining and data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Scikit-Learn 데이터 표현 – Feature Matrix

- 표본(sample)은 데이터셋이 설명하는 개별 객체를 나타냄
- 특징(feature)은 각 표본을 연속적인 수치값, 부울값, 이산값으로 표현하는 개별 관측치를 의미
- 표본 : 행렬의 행
- 행의 개수 : `n_samples`
- 특징(feature) : 행렬의 열
- 열의 개수 : `n_features`
- 관례적으로 특징행렬은 변수 `X`에 저장
- `[n_samples, n_features]` 형태의 2차원 배열 구조를 사용
(주로 Numpy 배열, Pandas DataFrame, SciPy 희소행렬을 사용)

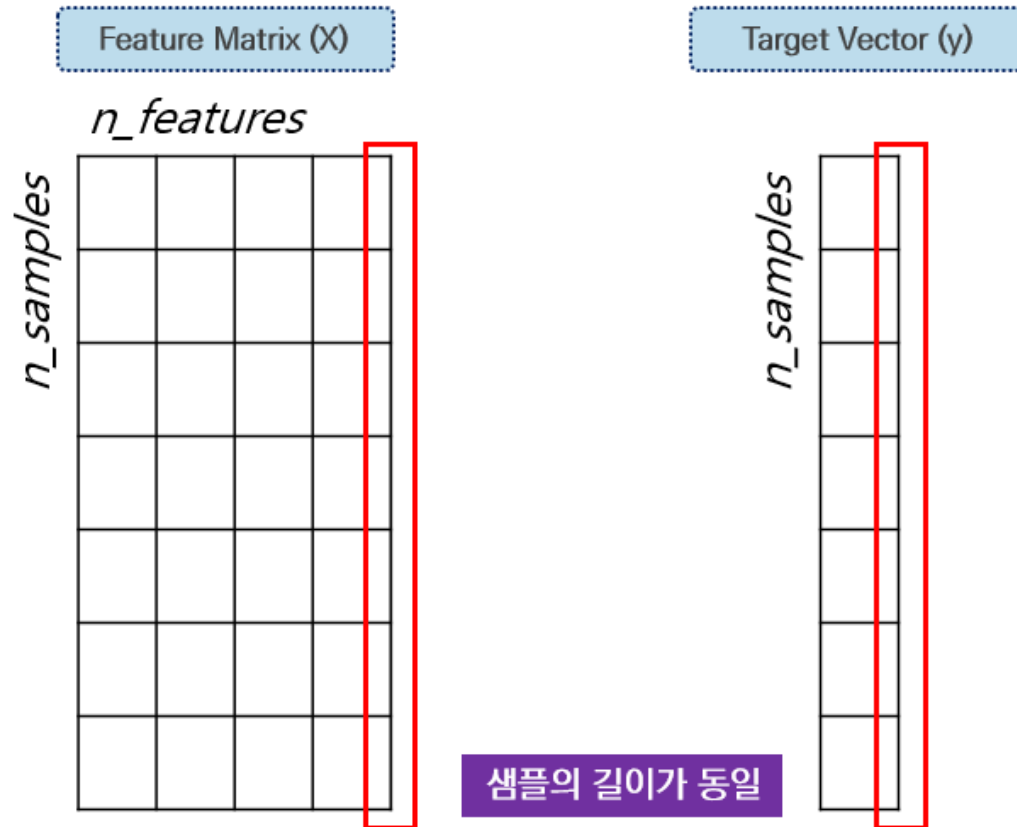
Scikit-Learn 데이터 표현 – Target Vector

- 연속적인 수치값, 이산 클래스/레이블을 가짐
- 길이 : `n_samples`
- 관례적으로 대상벡터는 변수 `y`에 저장
- 1차원 배열 구조를 사용 (주로 Numpy 배열, Pandas Series를 사용)
- 특징 행렬로부터 예측하고자 하는 값의 벡터
- 종속 변수, 출력 변수, 결과 변수, 반응 변수 라고도 함

Scikit-Learn 데이터 표현 – Target Vector

- 연속적인 수치값, 이산 클래스/레이블을 가짐
- 길이 : `n_samples`
- 관례적으로 대상벡터는 변수 `y`에 저장
- 1차원 배열 구조를 사용 (주로 Numpy 배열, Pandas Series를 사용)
- 특징 행렬로부터 예측하고자 하는 값의 벡터
- 종속 변수, 출력 변수, 결과 변수, 반응 변수 라고도 함

Scikit-Learn 데이터 표현 - Layout



Scikit-Learn 데이터 표현 - Layout

```
import seaborn as sd
```

```
iris = sd.load_dataset('iris')
```

```
iris.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 150 entries, 0 to 149  
Data columns (total 5 columns):  
sepal_length    150 non-null float64  
sepal_width     150 non-null float64  
petal_length    150 non-null float64  
petal_width     150 non-null float64  
species         150 non-null object  
dtypes: float64(4), object(1)  
memory usage: 5.9+ KB
```

```
iris.head()
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

Feature Matrix와 Target Vector를 나누는 작업 필요

Scikit-Learn 데이터 표현 - Layout

```
X = iris.drop('species', axis=1)
X.head()
```

“species” 열이 삭제된 X 데이터프레임(Feature Matrix) 생성

	sepal_length	sepal_width	petal_length	petal_width
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

```
y = iris['species']
y.head()
```

“species” 열을 추출하여 y Series(Target Vector) 생성

```
0    setosa
1    setosa
2    setosa
3    setosa
4    setosa
Name: species, dtype: object
```

Scikit-Learn 데이터 표현 - Layout

```
X = iris.drop('species', axis=1)
X.head()
```

“species” 열이 삭제된 X 데이터프레임(Feature Matrix) 생성

	sepal_length	sepal_width	petal_length	petal_width
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

```
y = iris['species']
y.head()
```

“species” 열을 추출하여 y Series(Target Vector) 생성

```
0    setosa
1    setosa
2    setosa
3    setosa
4    setosa
Name: species, dtype: object
```

07-1. add_feature.ipynb 실습

Scikit-Learn을 활용한 머신러닝 수행 절차

1

데이터 준비

2

모델 클래스 선택

3

모델 인스턴스 생성과 하이퍼파라미터 선택

4

특징 행렬과 대상 벡터 준비

5

모델을 데이터에 적합

6

새로운 데이터를 이용해 예측

7

모델 평가

Scikit-Learn을 활용한 머신러닝 수행 절차

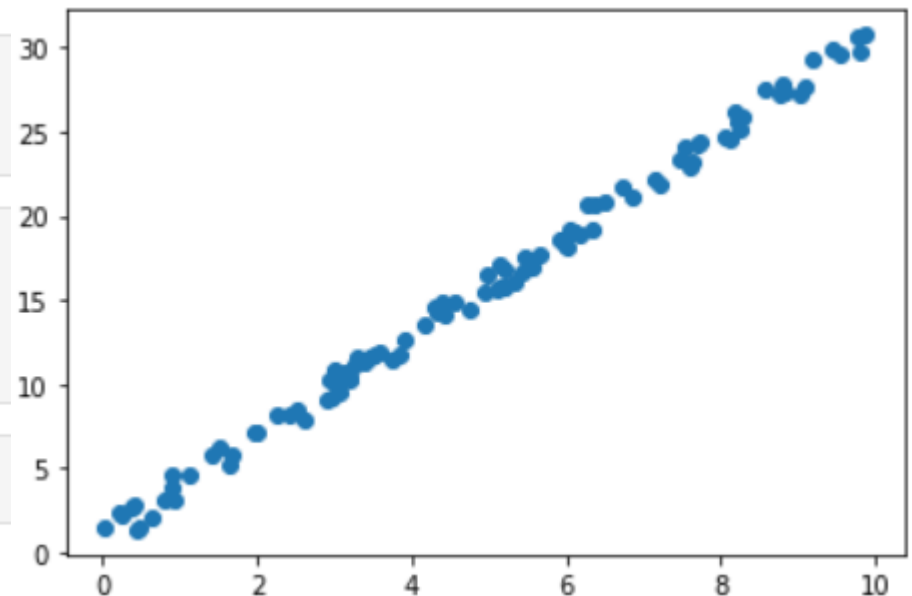
1

데이터 준비

```
import numpy as np
import matplotlib.pyplot as plt
```

```
rs = np.random.RandomState(10)
X = 10 * rs.rand(100)
y = 3 * X + 2 * rs.rand(100)
```

```
plt.scatter(X, y)
```



Scikit-Learn을 활용한 머신러닝 수행 절차

2

모델 클래스 선택

3

모델 인스턴스 생성과 하이퍼파라미터 선택

입력데이터(x), 출력데이터(y)가 모두 연속형 수치 데이터이므로 그에 맞는 분석 모델을 선택

```
from sklearn.linear_model import LinearRegression  
regr = LinearRegression()
```

선형회귀 객체(인스턴스) 생성 - 디폴트

```
from sklearn.linear_model import LinearRegression  
regr = LinearRegression(fit_intercept = True)
```

선형회귀 객체(인스턴스) 생성 -
(fit_intercept=True라는 하이퍼파라미터를 제공)

Scikit-Learn을 활용한 머신러닝 수행 절차

4

특징 행렬과 대상 벡터 준비

```
X = x.reshape(-1, 1)  
print(X.shape, y.shape)
```

(100, 1) (100,)

5

모델을 데이터에 적합

```
regr.fit(X, y)
```

X, y에 맞는 선형회귀 모델을 적합(모델 생성)

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

```
regr.coef_
```

모델의 기울기

```
array([2.9855087])
```

```
regr.intercept_
```

모델의 y 절편

```
0.9878534341975644
```

Scikit-Learn을 활용한 머신러닝 수행 절차

6

새로운 데이터를 이용해 예측

```
x_new = np.linspace(-1, 11, num=100)
```

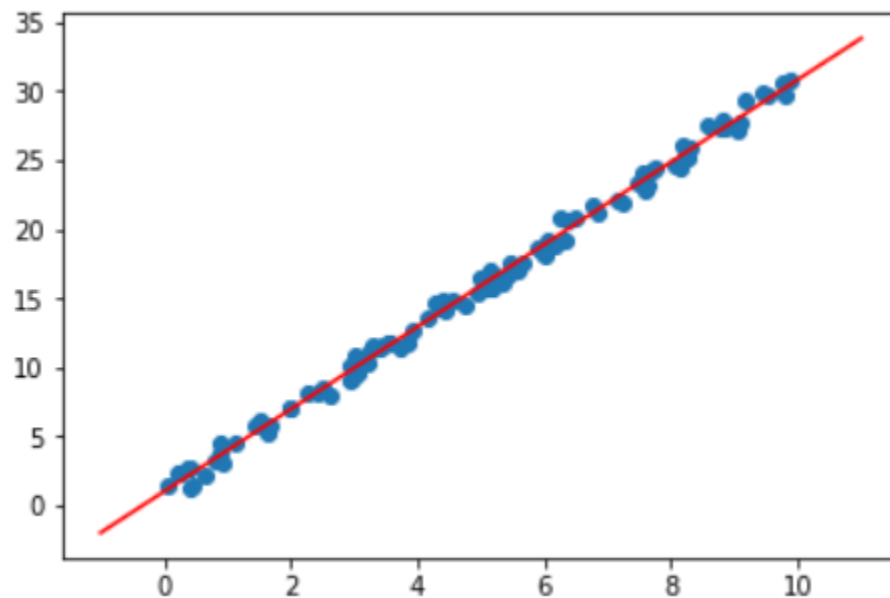
```
X_new = x_new.reshape(-1, 1)  
X_new.shape
```

```
(100, 1)
```

```
y_pred = regr.predict(X_new)
```

새로 입력된 X_new에 대한
모델 예측값(y_pred) 생성

```
plt.plot(X_new, y_pred, c='red')  
plt.scatter(X, y)
```



Scikit-Learn을 활용한 머신러닝 수행 절차

7

모델 평가

```
from sklearn.metrics import mean_squared_error
```

```
rmse = np.sqrt(mean_squared_error(y, y_pred))
```

```
print(rmse)
```

```
13.708237122486333
```


데이터 레이블링

```
import seaborn as sd
import pandas as pd
```

```
iris = sd.load_dataset('iris')
```

```
X = iris.drop('species', axis=1)
y = iris['species']
```

```
iris['species'].value_counts()
```

```
virginica    50
setosa       50
versicolor  50
```

```
Name: species, dtype: int64
```

```
from sklearn.preprocessing import LabelEncoder
```

```
encoder = LabelEncoder()
y = encoder.fit_transform(y)
```

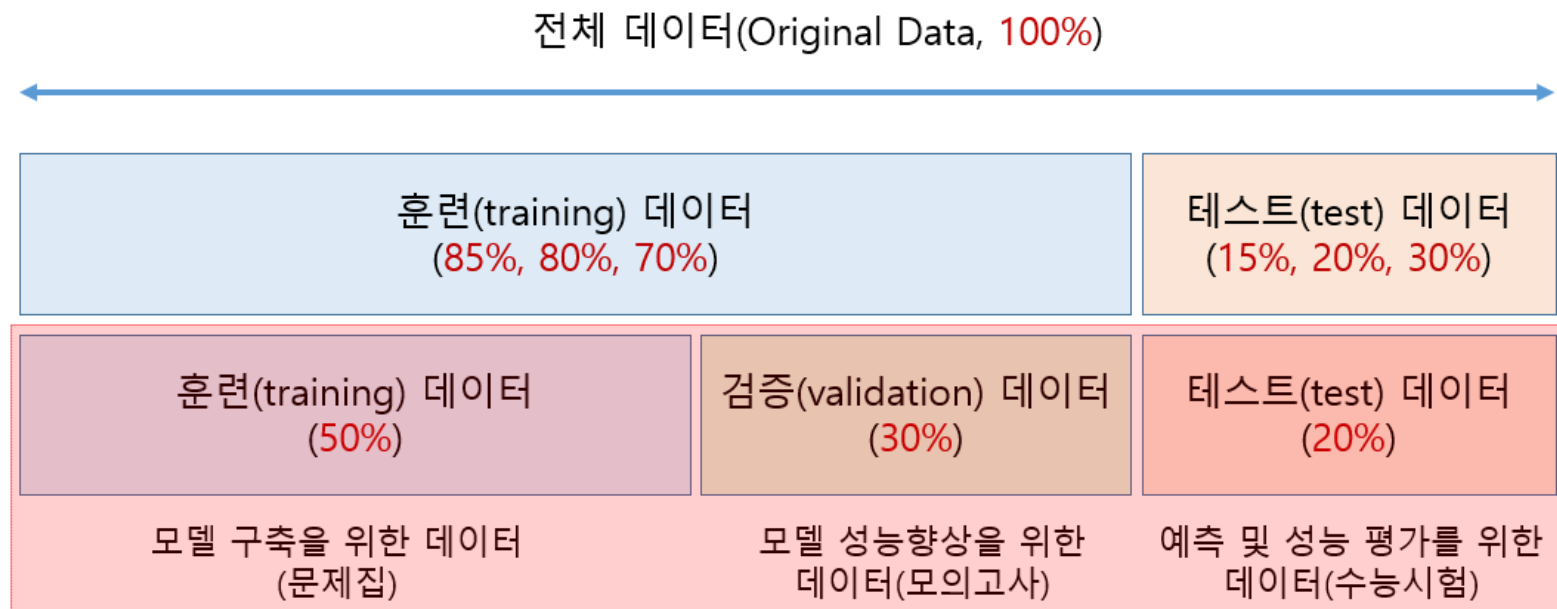
```
y
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2], dtype=int64)
```

TargetVector 에 대한 Labeling 필요

데이터 분할

머신러닝/딥러닝 학습데이터는 훈련(training) 데이터, 검증(validation) 데이터 및 테스트(test)로 분할(split)하여 사용



훈련 데이터 : 모델의 훈련 및 가중치 업데이트 등의 목적으로 사용

검증 데이터 : 훈련된 모델의 평가 및 최종 모델을 선정하기 위해 사용

테스트 데이터 : 모델의 예측 및 평가를 위해 사용

데이터 분할

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state=25)
```

```
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

```
(120, 4) (30, 4) (120,) (30,)
```

```
pd.Series(y_train).value_counts()
```

```
2    42  
0    41  
1    37
```

```
dtype: int64
```

데이터 분할

- 랜덤 층화 샘플링

- 각 층별 배분된 표본을 단순임의추출의 방법으로 표본을 추출

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, shuffle=True, stratify=y, random_state=25)
```

```
pd.Series(y_train).value_counts()
```

```
2    40  
1    40  
0    40
```

```
dtype: int64
```

교차 검증(Cross Validation)

- 교차검증의 개념 및 절차

- ① 교차 검증 1단계에서는 데이터를 학습용과 테스트용으로 나눔
- ② 모델의 테스트 성능을 기록
- ③ 교차 검증의 매 단계마다 다른 파티션으로 위의 작업을 수행
- ④ 모델의 최종 성능은 매 단계의 테스트 성능을 평균 계산

교차 검증은 모델의 변동성을 줄여주며 오버피팅 방지 효과

교차 검증을 통해 모든 데이터를 학습용 데이터로 사용할 수 있음

교차 검증(Cross Validation)

- k 폴드 교차 검증(K-fold Cross Validation)

- 데이터를 무작위로 k개의 동일한 크기인 폴드로 분할 (보통 k값으로 3, 5, 10을 많이 사용)
- 각 시행 단계에서 특정 폴드를 테스트용으로, 나머지는 학습용으로 사용
- 각 폴드를 테스트 세트로 한 번씩 사용하고 이 과정을 k번 반복 시행함
- 최종적으로 모델 성능의 평균을 계산

- $k = 5$, repeat = 5

반복시행	폴드1	폴드2	폴드3	폴드4	폴드5
1	테스트	학습	학습	학습	학습
2	학습	테스트	학습	학습	학습
3	학습	학습	테스트	학습	학습
4	학습	학습	학습	테스트	학습
5	학습	학습	학습	학습	테스트

교차 검증(Cross Validation)

```
from sklearn.datasets import load_iris
iris = load_iris()
X = iris.data
y = iris.target
```

```
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier(n_neighbors=3)
```

```
from sklearn.model_selection import cross_val_score
cross_val_score(model, X, y, cv = 5)
```

```
array([0.96666667, 0.96666667, 0.93333333, 0.96666667, 1.      ])
```

교차 검증(Cross Validation)

▪ 단일 관측치 제거 방식(LOOCV)

– Leave-one-out cross validation

검증을 시행할 때 마다 한 지점을 제외한 모든 지점에서 훈련

– 매 시행 단계에서 테스트 샘플을 고정하는 방식

– 데이터를 n개의 서브세트로 분할하고, n개 중 1개를 테스트용으로 두고 n-1개로 학습을 수행

– 데이터 크기가 n이면 n번의 교차 검증을 수행



$$CV_{(n)} = \frac{1}{n} \sum_{i=1}^n MSE_i = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

교차 검증(Cross Validation)

```
from sklearn.model_selection import LeaveOneOut
scores = cross_val_score(model, X, y, cv = LeaveOneOut())
scores
```

```
array([1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,  
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,  
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,  
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,  
       1., 1., 0., 1., 0., 1., 1., 1., 1., 1., 1., 1., 1., 1., 0., 1.,  
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,  
       1., 1., 1., 1., 0., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,  
       0., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 0., 1.,  
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.]
```

Grid Search

```
from sklearn.datasets import load_iris
import numpy as np
iris = load_iris()
X = iris.data
y = iris.target
```

```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import GridSearchCV
```

```
params = {"fit_intercept": [True, False],
          "normalize": [True, False]}
grid = GridSearchCV(LinearRegression(), params, iid=True, cv=7)
```

Grid Search

```
grid.fit(X, y)
```

```
GridSearchCV(cv=7, error_score='raise-deprecating',  
             estimator=LinearRegression(copy_X=True, fit_intercept=True,  
                                         n_jobs=None, normalize=False),  
             iid=True, n_jobs=None,  
             param_grid={'fit_intercept': [True, False],  
                         'normalize': [True, False]},  
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,  
             scoring=None, verbose=0)
```

```
grid.best_params_
```

```
{'fit_intercept': True, 'normalize': False}
```

```
model = grid.best_estimator_  
model
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

Q&A

A 3D blue 'Q&A' text is centered on a white surface. A computer mouse with a grey cord is positioned to the right of the text, with the cord looping around the base of the 'Q'.

Thank you

A hand-drawn 'Thank you' note is shown on a white surface. The text 'Thank you' is written in a cursive script, and a pen is visible at the end of the line.