Machine Learning with Python

Life is too short, You need Python

실습 내용

- K-Fold Cross Validation을 사용해 모델의 성능을 예측합니다.

1.환경 준비

- 기본 라이브러리와 대상 데이터를 가져와 이후 과정을 준비합니다.

In [1]:
```
# 라이브러리 불러오기
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings

warnings.filterwarnings(action='ignore')
%config InlineBackend.figure_format='retina'
```

In [2]:
```
# 데이터 읽어오기
path = 'https://raw.githubusercontent.com/Jangrae/csv/master/boston.csv'
data = pd.read_csv(path)
```

2.데이터 이해

- 분석할 데이터를 충분히 이해할 수 있도록 다양한 탐색 과정을 수행합니다.

In [3]:
```
# 데이터 살펴보기
data.head()
```

Out[3]:

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat	medv
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	36.2

In [4]:
```
# 기술통계 확인
data.describe()
```

Out[4]:

	crim	zn	indus	chas	nox	rm	age	dis
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.574901	3.795043
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.148861	2.105710
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1.129600
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025000	2.100175
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500000	3.207450
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.075000	5.188425
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.126500

3.데이터 준비

- 전처리 과정을 통해 머신러닝 알고리즘에 사용할 수 있는 형태의 데이터를 준비합니다.

1) x, y 분리

In [5]:
```
# Target 확인
target = 'medv'

# x, y 분리
x = data.drop(target, axis=1)
y = data.loc[:, target]
```

2) 학습용, 평가용 데이터 분리

In [6]:
```
# 라이브러리 불러오기
from sklearn.model_selection import train_test_split
```

```
# 학습용, 평가용 데이터 7:3으로 분리
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=1)
```

3) 정규화

In [7]:
```
# 모듈 불러오기
from sklearn.preprocessing import MinMaxScaler

# 정규화
scaler = MinMaxScaler()
scaler.fit(x_train)
x_train_s = scaler.transform(x_train)
x_test_s = scaler.transform(x_test)
```

4.성능 예측

- K분할 교차 검증 방법으로 모델 성능을 예측합니다.
- cross_val_score(model, x_train, y_train, cv=n) 형태로 사용합니다.
- cv 옵션에 k값(분할 개수, 기본값=5)을 지정합니다.
- cross_val_score 함수는 넘파이 배열 형태의 값을 반환합니다.
- cross_val_score 함수 반환 값의 평균을 해당 모델의 예측 성능으로 볼 수 있습니다.

1) Linear Regression

In [9]:
```
# 불러오기
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import cross_val_score

# 선언하기
model = LinearRegression()

# 검증하기
cv_score = cross_val_score(model, x_train_s, y_train, cv=10)

# 확인
print(cv_score.mean())

# 성능정보 저장
result = {} #초기화
result['Linear Regression'] = cv_score.mean()
print(result)
```

```
0.6681620144824075
{'Linear Regression': 0.6681620144824075}
```

2) KNN

In [11]:
```
# 불러오기
from sklearn.neighbors import KNeighborsRegressor
from sklearn.model_selection import cross_val_score

# 선언하기
model = KNeighborsRegressor()
```

```
# 검증하기
cv_score = cross_val_score(model, x_train_s, y_train, cv=10)

# 확인
print(cv_score)
print(cv_score.mean())

# 예측 결과 저장
result['KNN'] = cv_score.mean()
print(result)
```

```
[0.65736358 0.52515086 0.75283937 0.44262082 0.78083425 0.64172252
 0.56042514 0.65223096 0.4254619  0.81733353]
0.6255982942610578
{'Linear Regression': 0.6681620144824075, 'KNN': 0.6255982942610578}
```

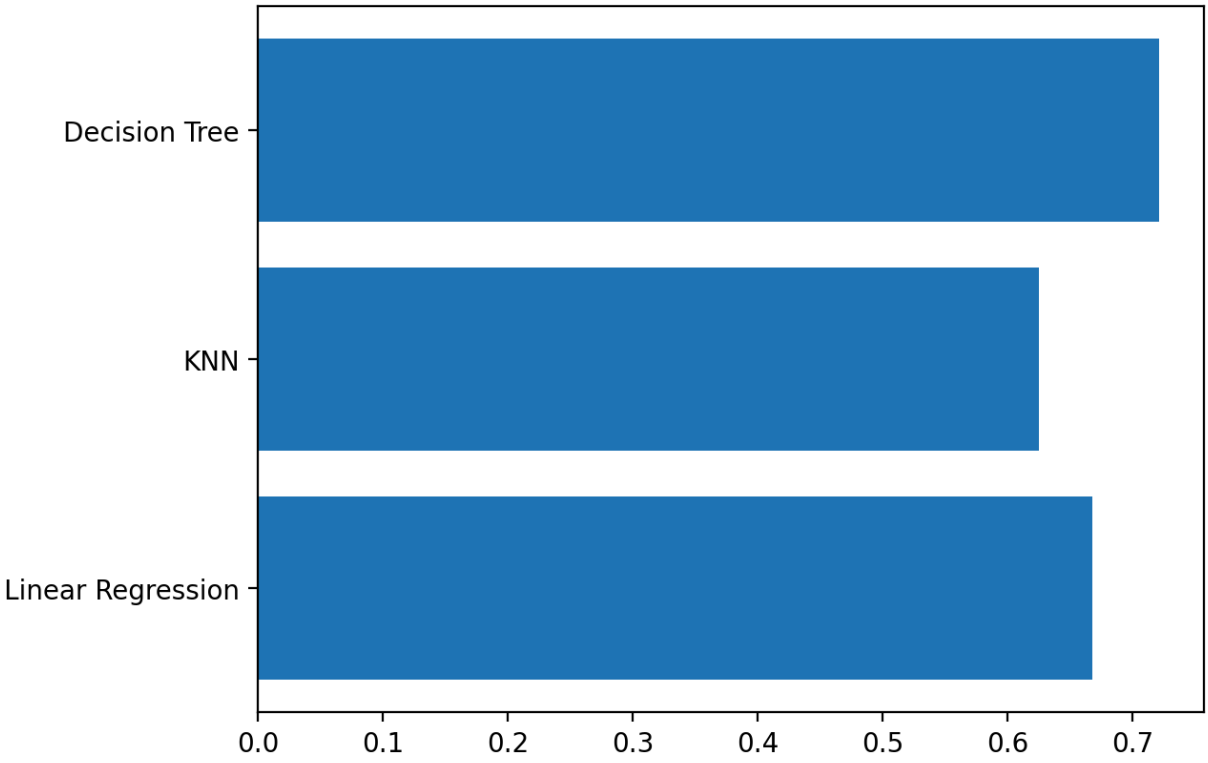
3) Decision Tree

In [13]:
```
# 불러오기
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import cross_val_score

# 선언하기
model = DecisionTreeRegressor(random_state=1)

# 검증하기
cv_score = cross_val_score(model, x_train, y_train, cv=10)

# 확인
print(cv_score.mean())

# 예측 결과 저장
result['Decision Tree'] = cv_score.mean()

# 확인
print(result)
```

```
0.7216522274237595
{'Linear Regression': 0.6681620144824075, 'KNN': 0.6255982942610578, 'Decision Tree': 0.721652
2274237595}
```

4) 성능 시각화 비교

In [14]:
```
plt.barh(y=list(result), width=result.values(), data=data)
plt.show()
```



5. 성능 평가

In [16]:
```
from sklearn.metrics import mean_absolute_error, r2_score

# 모델 선언
model = DecisionTreeRegressor()

# 학습
model.fit(x_train, y_train)

# 예측
y_pred = model.predict(x_test)

# 평가
print(mean_absolute_error(y_test, y_pred))
print(r2_score(y_test, y_pred))
```

```
3.0210526315789474
0.774490611185636
```

In [ ]:
```

```