```
import pandas as pd
import numpy as np
from sklearn.datasets import load_boston
import random
from sklearn import linear_model
from sklearn import preprocessing

boston_dataset = load_boston()
df = pd.DataFrame(boston_dataset.data, columns=boston_dataset.feature_names)
df['MEDV'] = boston_dataset.target
df.head()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	L
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	

```
x = df.drop(columns=['MEDV'])
x.shape
```

```
(506, 13)
```

```
x = x.values #returns a numpy array
min_max_scaler = preprocessing.MinMaxScaler()
x_scaled = min_max_scaler.fit_transform(x)
df = pd.DataFrame(x_scaled, columns=boston_dataset.feature_names)
df['MEDV'] = boston_dataset.target
df.head()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	
0	0.000000	0.18	0.067815	0.0	0.314815	0.577505	0.641607	0.269203	0.000000	0.20
1	0.000236	0.00	0.242302	0.0	0.172840	0.547998	0.782698	0.348962	0.043478	0.10
2	0.000236	0.00	0.242302	0.0	0.172840	0.694386	0.599382	0.348962	0.043478	0.10
3	0.000293	0.00	0.063050	0.0	0.150206	0.658555	0.441813	0.448545	0.086957	0.06
4	0.000705	0.00	0.063050	0.0	0.150206	0.687105	0.528321	0.448545	0.086957	0.06

```
def get_subsets(train_ratio,x):
  nsamples = int(np.round(len(x)*0.8))
  ind = random.sample(range(len(x)),nsamples)
  ind = list(np.sort(ind))
  x_sub_train = x.iloc[ind]
  y_sub_train = x_sub_train['MEDV']
  x_sub_train = x_sub_train.drop(columns=['MEDV'])
```

```
    ind_test = []
    for i in range(len(x)):
      if i not in ind:
        ind_test.append(i)

    ind_test = list(np.sort(ind_test))
    x_sub_test = x.iloc[ind_test]
    y_sub_test = x_sub_test['MEDV']
    x_sub_test = x_sub_test.drop(columns=['MEDV'])

    return x_sub_train, y_sub_train, x_sub_test, y_sub_test


x_train, y_train, x_test, y_test = get_subsets(0.8,df)


x_train.shape,y_train.shape,x_test.shape, y_test.shape
```

```
((405, 13), (405,), (101, 13), (101,))
```

```
ridgeReg = linear_model.Ridge(alpha=0.01)
ridgeReg.fit(x_train,y_train)
```

```
Ridge(alpha=0.01, copy_X=True, fit_intercept=True, max_iter=None,
      normalize=False, random_state=None, solver='auto', tol=0.001)
```

```
scores = []
for i in range(0,10):
  x_train, y_train, x_test, y_test = get_subsets(0.8,df)
  ridgeReg = linear_model.Ridge(alpha=0.01, tol=1e-4)
  ridgeReg.fit(x_train,y_train)
  ridgeReg.predict(x_test)
  score = ridgeReg.score(x_test,y_test)
  scores.append(score)
mean_r2 = np.mean(scores)
print(f'Mean r-Squared value over 10 iterations = {mean_r2}')
```

```
Mean r-Squared value over 10 iterations = 0.7510342463142445
```