Importação das bibliotecas necessárias

In [1]:

```
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sn
from sklearn.datasets import load_boston
from sklearn.metrics import *
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPRegressor
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler
```

In [2]:

```
def highlight_max(s):
    """
    highlight the maximum in a Series yellow.
    """
    is_max = s == s.max()
    is_min = s == s.min()

    return_arr = []

    for i in range(len(is_max)):
        if is_max[i]:
            return_arr.append('background-color: red')
        elif is_min[i]:
            return_arr.append('background-color: blue')
        else:
            return_arr.append('')

    return return_arr
```

In [3]:

```
def normalization(data):
    scaler = MinMaxScaler(feature_range=(0.01, 1))
    scaler.fit(data)
    data = scaler.transform(data)
    return data

def standardization(data):
    scaler = StandardScaler()
    scaler.fit(data)
    data = scaler.transform(data)
    return data

def train(X_train, y_train, X_test):
    regr = MLPRegressor(max_iter=5000, solver='adam', activation='relu', hidden_layer_size
s=(100, 10))
    regr.fit(X_train, y_train)
    y_pred = regr.predict(X_test)
    mse = mean_squared_error(y_test, y_pred)
    r2 = regr.score(X_test, y_test)
    return regr, mse, r2
```

Carregar dos dados

O conjunto de dados descreve 13 features numéricas de casas nos subúrbios de Boston.

O objectivo é estimar o preço das casas nesses subúrbios em milhares de dólares, através das features de entrada.

As features de entrada incluem coisas como taxa de criminalidade, concentrações de produtos químicos, entre outros.

In [4]:

```
boston = load_boston()
```

In [5]:

```
X = boston.data
y = boston.target

feature_names = boston.feature_names
```

In [6]:

```
X_read = pd.DataFrame(boston.data, columns=feature_names)
X_read.head()
```

Out[6]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LST
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	4.
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	9.
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	4.
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	2.
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	5.

In [7]:

```
y_read = pd.DataFrame(boston.target, columns=['Target'])
y_read.head()
```

Out[7]:

	Target
0	24.0
1	21.6
2	34.7
3	33.4
4	36.2

Divisão dos dados em conjunto de treino e de teste

> Vamos usar 70 % dos dados para treino e 30 % para teste.

In [8]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

Modelo base

In [9]:

```
regr_base, mse_base, r2_base = train(X_train, y_train, X_test)
```

Data normalization

In [10]:

```
X_train_norm = normalization(X_train)
X_test_norm = normalization(X_test)
regr_norm, mse_norm, r2_norm = train(X_train_norm, y_train, X_test_norm)
```

Data standardization

A standardization de um conjunto de dados envolve o reescalonamento da distribuição de valores de modo que a média dos valores observados seja 0 e o desvio padrão seja 1.

Isso pode ser considerado como subtração do valor médio ou centralização dos dados.

É especialmente útil quando as features de entrada tem valores com escalas diferentes.

In [11]:

```
X_train_std = standardization(X_train)
X_test_std = standardization(X_test)
regr_std, mse_std, r2_std = train(X_train_std, y_train, X_test_std)
```

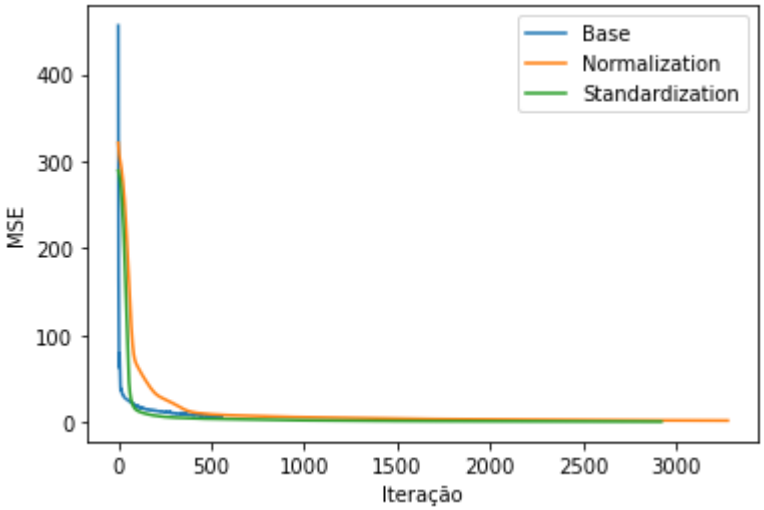
Comparação entre as 3 abordagens

In [12]:

```
fig, ax = plt.subplots()

line1 = ax.plot(regr_base.loss_curve_, label='Base')
line2 = ax.plot(regr_norm.loss_curve_, label='Normalization')
line3 = ax.plot(regr_std.loss_curve_, label='Standardization')

ax.legend()
plt.xlabel('Iteração')
plt.ylabel('MSE')
plt.show()
```



In [13]:

```
results = {'MSE': [mse_base, mse_norm, mse_std] , 'R^2': [r2_base, r2_norm, r2_std]}
results = pd.DataFrame(data=results, index=['Base', 'Normalization', 'Standardization'])

results.style.apply(highlight_max)
```

Out[13]:

	MSE	R^2
Base	23.345836	0.714003
Normalization	17.176084	0.789586
Standardization	13.898641	0.829736

Feature selection

In [14]:

```
data = pd.concat([pd.DataFrame(X, columns=feature_names), pd.DataFrame(y, columns=['Targe
t'])], axis=1)
```

Baseado na variância

In [15]:

```
data.var().to_frame(name='Variance').sort_values(by='Variance', ascending=False)
```

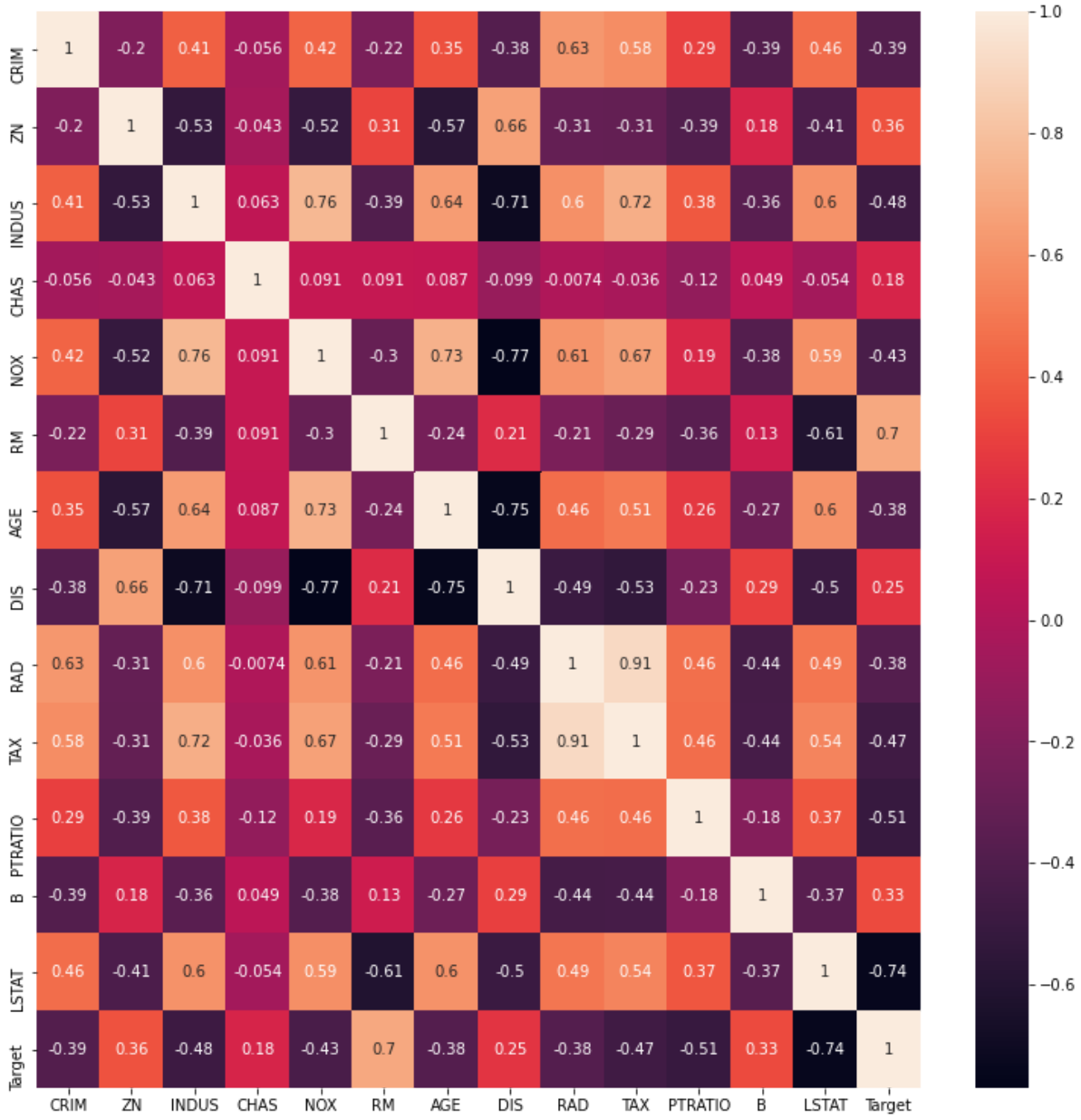
Out[15]:

	Variance
TAX	28404.759488
B	8334.752263
AGE	792.358399
ZN	543.936814
Target	84.586724
RAD	75.816366
CRIM	73.986578
LSTAT	50.994760
INDUS	47.064442
PTRATIO	4.686989
DIS	4.434015
RM	0.493671
CHAS	0.064513
NOX	0.013428

Baseado na correlação

In [16]:

```
plt.figure(figsize=(13,13))
corr_matrix = data.corr()
sn.heatmap(corr_matrix, annot=True)
plt.show()
```



In [17]:

```
all_correlation_target = abs(corr_matrix.loc['Target'][:-1]).to_frame()
```

In [18]:

```
n_best = 5
all_correlation_target.nlargest(n_best, columns='Target')
```

Out[18]:

	Target
LSTAT	0.737663
RM	0.695360
PTRATIO	0.507787
INDUS	0.483725
TAX	0.468536

- LSTAT - % status inferior da população
- RM - número médio de cômodos por habitação
- PTRATIO - proporção professor-aluno por cidade
- INDUS - proporção de lojas por cidade
- TAX - taxa de imposto sobre a propriedade