Задание 1.
Исходные данные:
Импортируйте библиотеки pandas и numpy.
Загрузите "Boston House Prices dataset" из встроенных наборов данных библиотеки sklearn.
Создайте датафреймы X и y из этих данных.
Разбейте эти датафреймы на тренировочные (X_train, y_train) и тестовые (X_test, y_test) с помощью функции train_test_split так, чтобы размер тестовой выборки
составлял 30% от всех данных, при этом аргумент random_state должен быть равен 42.
Создайте модель линейной регрессии под названием lr с помощью класса LinearRegression из модуля sklearn.linear_model.
Обучите модель на тренировочных данных (используйте все признаки) и сделайте предсказание на тестовых.
Вычислите R2 полученных предказаний с помощью r2_score из модуля sklearn.metrics.
Решение:

```
Python 3.8.10 (default, Jun  2 2021, 10:49:15)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy as np
>>> import pandas as pd
>>> import sklearn
>>> from sklearn.datasets import load_boston
>>> city_boston = load_boston()
>>> city_boston.keys()
dict_keys(['data', 'target', 'feature_names', 'DESCR', 'filename'])
>>> data = city_boston["data"]
>>> data.shape
(506, 13)
>>> feature_names = city_boston["feature_names"]
>>> feature_names
array(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD',
       'TAX', 'PTRATIO', 'B', 'LSTAT'], dtype='<U7')
>>> target = city_boston["target"]
>>> target[:10]
array([24. , 21.6, 34.7, 33.4, 36.2, 28.7, 22.9, 27.1, 16.5, 18.9])
>>> X = pd.DataFrame(data, columns=feature_names)
>>> X.head()
      CRIM    ZN  INDUS  CHAS    NOX     RM   AGE     DIS  RAD    TAX  PTRATIO       B
LSTAT
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   4.98
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   9.14
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   4.03
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   2.94
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   5.33
>>> X.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 13 columns):
CRIM       506 non-null float64
ZN         506 non-null float64
INDUS      506 non-null float64
CHAS       506 non-null float64
NOX        506 non-null float64
```

```
RM         506 non-null float64
AGE        506 non-null float64
DIS        506 non-null float64
RAD        506 non-null float64
TAX        506 non-null float64
PTRATIO    506 non-null float64
B          506 non-null float64
LSTAT      506 non-null float64
dtypes: float64(13)
memory usage: 51.5 KB
>>> y = pd.DataFrame(target, columns=["price"])
>>> y.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 1 columns):
price    506 non-null float64
dtypes: float64(1)
memory usage: 4.1 KB
>>> from sklearn.model_selection import train_test_split
>>> X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state =42)
>>> from sklearn.linear_model import LinearRegression
>>> regressor = LinearRegression()
>>> regressor.fit(X_train, y_train)
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
>>> y_pred = regressor.predict(X_test)
>>> check_test = pd.DataFrame({"y_test": y_test["price"], "y_pred": y_pred.flatten()})
>>> check_test.head(10)
     y_test     y_pred
173    23.6  28.648960
274    32.4  36.495014
491    13.6  15.411193
72     22.8  25.403213
452    16.1  18.855280
76     20.0  23.146689
316    17.8  17.392124
140    14.0  14.078599
471    19.6  23.036927
500    16.8  20.599433
>>> y_test, y_pred
(     price
173   23.6
274   32.4
491   13.6
72    22.8
452   16.1
..     ...
441   17.1
23    14.5
225   50.0
433   14.3
447   12.6
```

[152 rows x 1 columns], array([[28.64896005],
[36.49501384],
[15.4111932 ],
[25.40321303],
[18.85527988],
[23.14668944],
[17.3921241 ],
[14.07859899],
[23.03692679],
[20.59943345],
[24.82286159],
[18.53057049],
[-6.86543527],
[21.80172334],
[19.22571177],
[26.19191985],
[20.27733882],
[ 5.61596432],
[40.44887974],
[17.57695918],
[27.44319095],
[30.1715964 ],
[10.94055823],
[24.02083139],
[18.07693812],
[15.934748  ],
[23.12614028],
[14.56052142],
[22.33482544],
[19.3257627 ],
[22.16564973],
[25.19476081],
[25.31372473],
[18.51345025],
[16.6223286 ],
[17.50268505],
[30.94992991],
[20.19201752],
[23.90440431],
[24.86975466],
[13.93767876],
[31.82504715],
[42.56978796],
[17.62323805],
[27.01963242],
[17.19006621],
[13.80594006],
[26.10356557],
[20.31516118],
[30.08649576],
[21.3124053 ],
[34.15739602],

[15.60444981],
[26.11247588],
[39.31613646],
[22.99282065],
[18.95764781],
[33.05555669],
[24.85114223],
[12.91729352],
[22.68101452],
[30.80336295],
[31.63522027],
[16.29833689],
[21.07379993],
[16.57699669],
[20.36362023],
[26.15615896],
[31.06833034],
[11.98679953],
[20.42550472],
[27.55676301],
[10.94316981],
[16.82660609],
[23.92909733],
[ 5.28065815],
[21.43504661],
[41.33684993],
[18.22211675],
[ 9.48269245],
[21.19857446],
[12.95001331],
[21.64822797],
[ 9.3845568 ],
[23.06060014],
[31.95762512],
[19.16662892],
[25.59942257],
[29.35043558],
[20.13138581],
[25.57297369],
[ 5.42970803],
[20.23169356],
[15.1949595 ],
[14.03241742],
[20.91078077],
[24.82249135],
[-0.47712079],
[13.70520524],
[15.69525576],
[22.06972676],
[24.64152943],
[10.7382866 ],
[19.68622564],

```
       [23.63678009],
       [12.07974981],
       [18.47894211],
       [25.52713393],
       [20.93461307],
       [24.6955941 ],
       [ 7.59054562],
       [19.01046053],
       [21.9444339 ],
       [27.22319977],
       [32.18608828],
       [15.27826455],
       [34.39190421],
       [12.96314168],
       [21.01681316],
       [28.57880911],
       [15.86300844],
       [24.85124135],
       [ 3.37937111],
       [23.90465773],
       [25.81792146],
       [23.11020547],
       [25.33489201],
       [33.35545176],
       [20.60724498],
       [38.4772665 ],
       [13.97398533],
       [25.21923987],
       [17.80946626],
       [20.63437371],
       [ 9.80267398],
       [21.07953576],
       [22.3378417 ],
       [32.32381854],
       [31.48694863],
       [15.46621287],
       [16.86242766],
       [28.99330526],
       [24.95467894],
       [16.73633557],
       [ 6.12858395],
       [26.65990044],
       [23.34007187],
       [17.40367164],
       [13.38594123],
       [39.98342478],
       [16.68286302],
       [18.28561759]]))
>>> check_test["error"] = check_test["y_pred"] - check_test["y_test"]
>>> check_test.head()
     y_test     y_pred     error
173    23.6  28.648960  5.048960
```

```
274    32.4  36.495014  4.095014
491    13.6  15.411193  1.811193
72     22.8  25.403213  2.603213
452    16.1  18.855280  2.755280
>>> from sklearn.metrics import r2_score
>>> r2_score_1 = r2_score(check_test["y_pred"], check_test["y_test"])
>>> r2_score_1
0.6693702691495631
>>>
```

Задание 2.
Исходные данные:
Создайте модель под названием model с помощью RandomForestRegressor из модуля sklearn.ensemble.
Сделайте агрумент n_estimators равным 1000,
max_depth должен быть равен 12 и random_state сделайте равным 42.
Обучите модель на тренировочных данных аналогично тому, как вы обучали модель LinearRegression,
но при этом в метод fit вместо датафрейма y_train поставьте y_train.values[:, 0],
чтобы получить из датафрейма одномерный массив Numpy,
так как для класса RandomForestRegressor в данном методе для аргумента y предпочтительно применение массивов вместо датафрейма.
Сделайте предсказание на тестовых данных и посчитайте R2. Сравните с результатом из предыдущего задания.
Напишите в комментариях к коду, какая модель в данном случае работает лучше.
Решение:

```
Python 3.8.10 (default, Jun  2 2021, 10:49:15)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy as np
>>> import pandas as pd
>>> import sklearn
>>> from sklearn.ensemble import RandomForestRegressor
>>> model = RandomForestRegressor(n_estimators=1000, max_depth=12, random_state=42)
>>> from sklearn.datasets import load_boston
>>> city_boston = load_boston()
>>> city_boston.keys()
dict_keys(['data', 'target', 'feature_names', 'DESCR', 'filename'])
>>> data = city_boston["data"]
>>> data.shape
(506, 13)
>>> feature_names = city_boston["feature_names"]
>>> feature_names
array(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD',
       'TAX', 'PTRATIO', 'B', 'LSTAT'], dtype='<U7')
>>> target = city_boston["target"]
>>> target[:10]
array([24. , 21.6, 34.7, 33.4, 36.2, 28.7, 22.9, 27.1, 16.5, 18.9])
>>> X = pd.DataFrame(data, columns=feature_names)
>>> X.head()
      CRIM    ZN  INDUS  CHAS    NOX     RM   AGE     DIS  RAD    TAX  PTRATIO       B
LSTAT
```

```
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   4.98
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   9.14
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   4.03
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   2.94
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   5.33
>>> X.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 13 columns):
CRIM       506 non-null float64
ZN         506 non-null float64
INDUS      506 non-null float64
CHAS       506 non-null float64
NOX        506 non-null float64
RM         506 non-null float64
AGE        506 non-null float64
DIS        506 non-null float64
RAD        506 non-null float64
TAX        506 non-null float64
PTRATIO    506 non-null float64
B          506 non-null float64
LSTAT      506 non-null float64
dtypes: float64(13)
memory usage: 51.5 KB
>>> y = pd.DataFrame(target, columns=["price"])
>>> y.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 1 columns):
price    506 non-null float64
dtypes: float64(1)
memory usage: 4.1 KB
>>> from sklearn.model_selection import train_test_split
>>> X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state =42)
>>> model.fit(X_train, y_train.values[:, 0])
RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                      max_depth=12, max_features='auto', max_leaf_nodes=None,
                      max_samples=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=1,
                      min_samples_split=2, min_weight_fraction_leaf=0.0,
                      n_estimators=1000, n_jobs=None, oob_score=False,
                      random_state=42, verbose=0, warm_start=False)
>>> y_pred = model.predict(X_test)
>>> y_pred.shape
(152,)
>>> check_test = pd.DataFrame({"y_test": y_test["price"], "y_pred": y_pred.flatten()})
>>> check_test.head(10)
     y_test     y_pred
173    23.6  22.846138
274    32.4  31.156114
491    13.6  16.297226
72     22.8  23.821036
```

```
452    16.1  17.212148
76     20.0  21.820092
316    17.8  19.866369
140    14.0  14.759938
471    19.6  21.235224
500    16.8  20.883103
>>> from sklearn.metrics import r2_score
>>> r2_score_2=r2_score(check_test["y_pred"], check_test["y_test"])
>>> r2_score_2
0.8481499145965063
>>>

Python 3.8.10 (default, Jun  2 2021, 10:49:15)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> r2_score_1 = 0.6693702691495631
>>> r2_score_2 = 0.8481499145965063
>>> r2_score_1<r2_score_2
True
>>>
```