Московский государственный технический
университет им. Н. Э. Баумана

Курс «Технологии машинного обучения»
Отчёт по лабораторной работе №4
«Линейные модели, SVM и деревья решений»

Выполнила:	Проверил:
Шимолина П.К.,	Нардид А.Н.,
группа ИУ5-61Б	каф. ИУ5
Дата:	Дата:
Подпись:	Подпись:

2023
Москва

lab-4

June 14, 2023

[1]:
```
import pandas as pd
import numpy as np
import plotly.express as px
import seaborn as sns
import matplotlib.pyplot as plt
```

[2]:
```
df = pd.read_csv("Life Expectancy Data.csv")
```

[3]:
```
df.head()
```

[3]:
```
       Country  Year      Status  Life expectancy   Adult Mortality  \
0  Afghanistan  2015  Developing             65.0             263.0
1  Afghanistan  2014  Developing             59.9             271.0
2  Afghanistan  2013  Developing             59.9             268.0
3  Afghanistan  2012  Developing             59.5             272.0
4  Afghanistan  2011  Developing             59.2             275.0

   infant deaths  Alcohol  percentage expenditure  Hepatitis B  Measles   …  \
0             62     0.01               71.279624         65.0     1154   …
1             64     0.01               73.523582         62.0      492   …
2             66     0.01               73.219243         64.0      430   …
3             69     0.01               78.184215         67.0     2787   …
4             71     0.01                7.097109         68.0     3013   …

   Polio  Total expenditure  Diphtheria    HIV/AIDS         GDP  Population  \
0    6.0               8.16        65.0         0.1  584.259210  33736494.0
1   58.0               8.18        62.0         0.1  612.696514    327582.0
2   62.0               8.13        64.0         0.1  631.744976  31731688.0
3   67.0               8.52        67.0         0.1  669.959000   3696958.0
4   68.0               7.87        68.0         0.1   63.537231   2978599.0

    thinness  1-19 years   thinness 5-9 years  \
0                   17.2                 17.3
1                   17.5                 17.5
2                   17.7                 17.7
3                   17.9                 18.0
4                   18.2                 18.2
```

```
   Income composition of resources  Schooling
0                            0.479       10.1
1                            0.476       10.0
2                            0.470        9.9
3                            0.463        9.8
4                            0.454        9.5

[5 rows x 22 columns]
```

[4]:
```
df.columns
```

[4]:
```
Index(['Country', 'Year', 'Status', 'Life expectancy ', 'Adult Mortality',
       'infant deaths', 'Alcohol', 'percentage expenditure', 'Hepatitis B',
       'Measles ', ' BMI ', 'under-five deaths ', 'Polio', 'Total expenditure',
       'Diphtheria ', ' HIV/AIDS', 'GDP', 'Population',
       ' thinness  1-19 years', ' thinness 5-9 years',
       'Income composition of resources', 'Schooling'],
      dtype='object')
```

[5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2938 entries, 0 to 2937
Data columns (total 22 columns):
 #   Column                           Non-Null Count  Dtype
---  ------                           --------------  -----
 0   Country                          2938 non-null   object
 1   Year                             2938 non-null   int64
 2   Status                           2938 non-null   object
 3   Life expectancy                  2928 non-null   float64
 4   Adult Mortality                  2928 non-null   float64
 5   infant deaths                    2938 non-null   int64
 6   Alcohol                          2744 non-null   float64
 7   percentage expenditure           2938 non-null   float64
 8   Hepatitis B                      2385 non-null   float64
 9   Measles                          2938 non-null   int64
 10   BMI                             2904 non-null   float64
 11  under-five deaths                2938 non-null   int64
 12  Polio                            2919 non-null   float64
 13  Total expenditure                2712 non-null   float64
 14  Diphtheria                       2919 non-null   float64
 15   HIV/AIDS                        2938 non-null   float64
 16  GDP                              2490 non-null   float64
 17  Population                       2286 non-null   float64
 18   thinness  1-19 years            2904 non-null   float64
 19   thinness 5-9 years              2904 non-null   float64
 20  Income composition of resources  2771 non-null   float64
 21  Schooling                        2775 non-null   float64
dtypes: float64(16), int64(4), object(2)
memory usage: 505.1+ KB
```

[6]:

```
df.isnull().sum()
```

[6]:

```
Country                              0
Year                                 0
Status                               0
Life expectancy                     10
Adult Mortality                     10
infant deaths                        0
Alcohol                            194
percentage expenditure               0
Hepatitis B                        553
Measles                              0
 BMI                                34
under-five deaths                    0
Polio                               19
Total expenditure                  226
Diphtheria                          19
 HIV/AIDS                            0
GDP                                448
Population                         652
 thinness  1-19 years               34
 thinness 5-9 years                 34
Income composition of resources    167
Schooling                          163
dtype: int64
```

[7]:

```
y= df["Life expectancy "]
X=  df.drop(["Life expectancy "], axis=1)
```

[8]:

```
y.fillna(y.median(), inplace=True)
```

[9]:

```
X.Year = pd.to_datetime(X.Year).dt.year
```

0.1

[10]:

```
import sklearn
import category_encoders as ce
bin_enc = ce.BinaryEncoder(drop_invariant=True)
X = bin_enc.fit_transform(X)
```

[11]:

```
X.head()
```

```
[11]:    Country_0  Country_1  Country_2  Country_3  Country_4  Country_5  \
      0          0          0          0          0          0          0
      1          0          0          0          0          0          0
      2          0          0          0          0          0          0
      3          0          0          0          0          0          0
      4          0          0          0          0          0          0

         Country_6  Country_7  Year  Status_0  …  Polio  Total expenditure  \
      0          0          1  1970         0  …    6.0               8.16
      1          0          1  1970         0  …   58.0               8.18
      2          0          1  1970         0  …   62.0               8.13
      3          0          1  1970         0  …   67.0               8.52
      4          0          1  1970         0  …   68.0               7.87

         Diphtheria     HIV/AIDS         GDP  Population   thinness  1-19 years  \
      0        65.0        0.1  584.259210  33736494.0                   17.2
      1        62.0        0.1  612.696514    327582.0                   17.5
      2        64.0        0.1  631.744976  31731688.0                   17.7
      3        67.0        0.1  669.959000   3696958.0                   17.9
      4        68.0        0.1   63.537231   2978599.0                   18.2

          thinness 5-9 years  Income composition of resources  Schooling
      0                 17.3                             0.479       10.1
      1                 17.5                             0.476       10.0
      2                 17.7                             0.470        9.9
      3                 18.0                             0.463        9.8
      4                 18.2                             0.454        9.5

      [5 rows x 29 columns]
```

```
[12]: X.fillna(X.mean(), inplace=True)
```

0.2

```
[13]: from sklearn.model_selection import train_test_split
      X_train, X_test, y_train, y_test= train_test_split(X, y, test_size= 0.30,␣
       ↪random_state=9)
```

0.3

```
[14]: from sklearn.linear_model import LinearRegression
      lr = LinearRegression()
      lr.fit(X_train, y_train)
      sc = lr.score(X_train, y_train)
      print('coefficient of determination:', sc)
      print('intercept:', lr.intercept_)
      print('slope:', lr.coef_)
```

```
coefficient of determination: 0.8246578016681347
intercept: 55.06661652520731
slope: [ 5.39868078e-01  3.00562333e-01  7.53268236e-01 -1.75790685e-01
 -4.00059136e-01 -1.16217061e-01  2.54676690e-01  1.92664428e-02
  1.58206781e-15  8.82491412e-01 -8.82491412e-01 -1.96947113e-02
  1.00510203e-01  9.71513717e-02 -3.83416146e-05 -1.61047720e-02
 -2.10888131e-05  4.48954665e-02 -7.46079555e-02  2.41455560e-02
  1.77176532e-02  4.56656976e-02 -4.47773693e-01  4.23917554e-05
 -5.49815561e-11 -6.89028877e-02 -3.17531192e-03  4.96058442e+00
  7.22793633e-01]
```

[15]:

```
y_lr = lr.predict(X_test)
```

0.4 SVM

[16]:

```
from sklearn import svm
s = svm.SVR()
s.fit(X_train, y_train)
```

[16]:

```
SVR()
```

[17]:

```
y_svr = s.predict(X_test)
```

0.5

[18]:

```
from sklearn.tree import DecisionTreeRegressor
dt = DecisionTreeRegressor(random_state=0)
dt.fit(X_train, y_train)
```

[18]:

```
DecisionTreeRegressor(random_state=0)
```

[19]:

```
y_dt = dt.predict(X_test)
```

0.6

[20]:

```
# mean squared error
from sklearn.metrics import mean_squared_error
print("          : ",mean_squared_error(y_test, y_lr))
print("SVM: ",mean_squared_error(y_test, y_svr))
print("        : ",mean_squared_error(y_test, y_dt))
```

```
          :  17.062129156285682
SVM:  100.89982155896014
        :  7.4750000000000005
```

[21]:

```
# mean absolute error
from sklearn.metrics import mean_absolute_error
```

```
print("          : ",mean_absolute_error(y_test, y_lr))
print("SVM: ",mean_absolute_error(y_test, y_svr))
print("        : ",mean_absolute_error(y_test, y_dt))
```

```
          :  3.0889831815712046
SVM:  7.579025871219417
        :  1.6577097505668936
```

0.7

```
[22]: list(zip(X_train.columns.values, dt.feature_importances_))
```

```
[22]: [('Country_0', 0.003586939599600534),
       ('Country_1', 0.00043528641487956213),
       ('Country_2', 0.0012445965189790977),
       ('Country_3', 0.0002955266491424785),
       ('Country_4', 0.0007594598269913248),
       ('Country_5', 0.0004473477471656755),
       ('Country_6', 5.904270988838898e-05),
       ('Country_7', 0.0004912794219232724),
       ('Year', 0.0),
       ('Status_0', 2.2158014778123868e-05),
       ('Status_1', 0.00015274333443504666),
       ('Adult Mortality', 0.13575760786841157),
       ('infant deaths', 0.002369175986155562),
       ('Alcohol', 0.008467301710122648),
       ('percentage expenditure', 0.006271696143048364),
       ('Hepatitis B', 0.0019369721723790198),
       ('Measles ', 0.0015214550273649533),
       (' BMI ', 0.006154391731776707),
       ('under-five deaths ', 0.029396373097045313),
       ('Polio', 0.003143098419361568),
       ('Total expenditure', 0.004901823552178708),
       ('Diphtheria ', 0.003381902051066721),
       (' HIV/AIDS', 0.5912459304592244),
       ('GDP', 0.0020723185008717114),
       ('Population', 0.002860086568717561),
       (' thinness  1-19 years', 0.006544627632363877),
       (' thinness 5-9 years', 0.0024582661817284606),
       ('Income composition of resources', 0.16736158716070615),
       ('Schooling', 0.016661005499693057)]
```

```
[23]: sum(dt.feature_importances_)
```

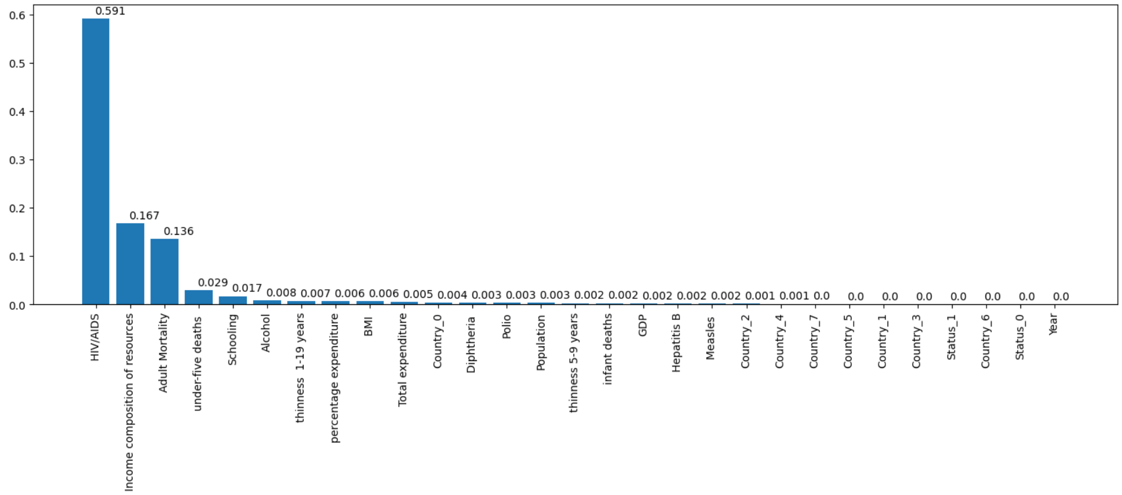
```
[23]: 1.0
```

[24]:
```
from operator import itemgetter

def draw_feature_importances(tree_model, X_dataset, figsize=(18,5)):
    """


    """
    #
    list_to_sort = list(zip(X_dataset.columns.values, tree_model.
 ↪feature_importances_))
    sorted_list = sorted(list_to_sort, key=itemgetter(1), reverse = True)
    #
    labels = [x for x,_ in sorted_list]
    #
    data = [x for _,x in sorted_list]
    #
    fig, ax = plt.subplots(figsize=figsize)
    ind = np.arange(len(labels))
    plt.bar(ind, data)
    plt.xticks(ind, labels, rotation='vertical')
    #
    for a,b in zip(ind, data):
        plt.text(a-0.05, b+0.01, str(round(b,3)))
    plt.show()
    return labels, data
```

[25]:
```
dt_fl, dt_fd = draw_feature_importances(dt, X_train)
```



0.8

[ ]:

```
from sklearn.model_selection import GridSearchCV
params = {
    'max_depth': [3, 4, 5, 6],
    'min_samples_leaf': [0.04, 0.06, 0.08],
    'max_features': [0.2, 0.4, 0.6, 0.8]
}
grid = GridSearchCV(estimator=DecisionTreeRegressor(random_state=1),
                    param_grid=params, scoring='neg_mean_squared_error', cv=3,␣
 ↪n_jobs=-1)
grid.fit(X_train, y_train)
```

[27]:

```
-grid.best_score_, grid.best_params_
```

[27]:

```
(12.980019491379105,
 {'max_depth': 6, 'max_features': 0.8, 'min_samples_leaf': 0.04})
```

[28]:

```
dt_opt = DecisionTreeRegressor(random_state=0, max_depth=6)
dt_opt.fit(X_train, y_train)
```

[28]:

```
DecisionTreeRegressor(max_depth=6, random_state=0)
```

[33]:

```
from IPython.core.display import HTML
from sklearn.tree import export_text
tree_rules = export_text(dt_opt, feature_names=list(X.columns))
HTML('<pre>' + tree_rules + '</pre>')
```

[33]:

```
<IPython.core.display.HTML object>
```

[35]:

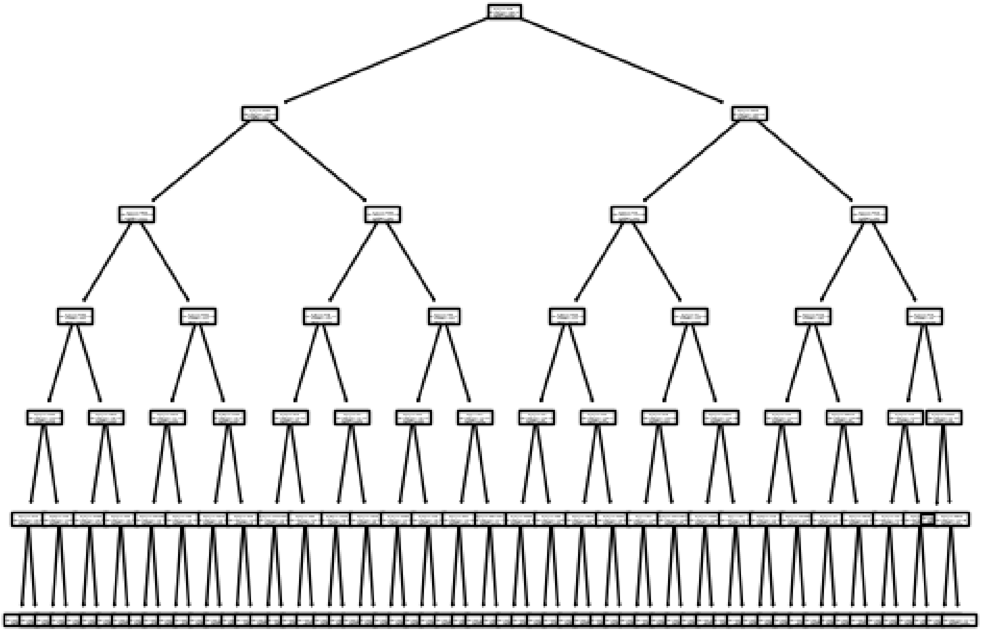
```
from sklearn import tree
```

[37]:

```
tree.plot_tree(dt_opt)
```

[37]:

```
[Text(0.514616935483871, 0.9285714285714286, 'X[22] <= 0.65\nsquared_error =
89.76\nsamples = 2056\nvalue = 69.288'),
 Text(0.25806451612903225, 0.7857142857142857, 'X[27] <= 0.802\nsquared_error =
34.271\nsamples = 1527\nvalue = 73.5'),
 Text(0.12903225806451613, 0.6428571428571429, 'X[11] <= 198.5\nsquared_error =
23.592\nsamples = 1144\nvalue = 71.365'),
 Text(0.06451612903225806, 0.5, 'X[27] <= 0.623\nsquared_error = 13.919\nsamples
= 933\nvalue = 72.854'),
 Text(0.03225806451612903, 0.35714285714285715, 'X[13] <= 1.595\nsquared_error =
14.833\nsamples = 218\nvalue = 69.158'),
 Text(0.016129032258064516, 0.21428571428571427, 'X[17] <= 37.0\nsquared_error =
9.862\nsamples = 127\nvalue = 67.494'),
 Text(0.008064516129032258, 0.07142857142857142, 'squared_error = 5.847\nsamples
= 79\nvalue = 66.108'),
 Text(0.024193548387096774, 0.07142857142857142, 'squared_error = 8.094\nsamples
= 48\nvalue = 69.777'),
 Text(0.04838709677419355, 0.21428571428571427, 'X[18] <= 14.5\nsquared_error =
12.515\nsamples = 91\nvalue = 71.48'),
 Text(0.04032258064516129, 0.07142857142857142, 'squared_error = 6.994\nsamples
= 62\nvalue = 72.547'),
 Text(0.056451612903225805, 0.07142857142857142, 'squared_error =
16.688\nsamples = 29\nvalue = 69.2'),
 Text(0.0967741935483871, 0.35714285714285715, 'X[11] <= 150.0\nsquared_error =
8.206\nsamples = 715\nvalue = 73.981'),
 Text(0.08064516129032258, 0.21428571428571427, 'X[27] <= 0.628\nsquared_error =
6.924\nsamples = 513\nvalue = 74.761'),
 Text(0.07258064516129033, 0.07142857142857142, 'squared_error = 10.418\nsamples
= 58\nvalue = 78.093'),
 Text(0.08870967741935484, 0.07142857142857142, 'squared_error = 4.883\nsamples
= 455\nvalue = 74.336'),
 Text(0.11290322580645161, 0.21428571428571427, 'X[13] <= 3.53\nsquared_error =
5.997\nsamples = 202\nvalue = 72.001'),
 Text(0.10483870967741936, 0.07142857142857142, 'squared_error = 7.282\nsamples
= 58\nvalue = 70.493'),
 Text(0.12096774193548387, 0.07142857142857142, 'squared_error = 4.194\nsamples
= 144\nvalue = 72.609'),
 Text(0.1935483870967742, 0.5, 'X[27] <= 0.544\nsquared_error = 13.218\nsamples
= 211\nvalue = 64.782'),
 Text(0.16129032258064516, 0.35714285714285715, 'X[11] <= 267.5\nsquared_error =
7.733\nsamples = 107\nvalue = 62.306'),
 Text(0.14516129032258066, 0.21428571428571427, 'X[14] <= 232.281\nsquared_error
= 3.106\nsamples = 75\nvalue = 63.697'),
 Text(0.13709677419354838, 0.07142857142857142, 'squared_error = 2.56\nsamples =
71\nvalue = 63.504'),
 Text(0.1532258064516129, 0.07142857142857142, 'squared_error = 0.387\nsamples =
4\nvalue = 67.125'),
 Text(0.1774193548387097, 0.21428571428571427, 'X[12] <= 76.5\nsquared_error =
3.397\nsamples = 32\nvalue = 59.044'),
 Text(0.1693548387096774, 0.07142857142857142, 'squared_error = 2.006\nsamples =
26\nvalue = 59.646'),
 Text(0.18548387096774194, 0.07142857142857142, 'squared_error = 1.039\nsamples
= 6\nvalue = 56.433'),
 Text(0.22580645161290322, 0.35714285714285715, 'X[28] <= 14.95\nsquared_error =
6.06\nsamples = 104\nvalue = 67.33'),
 Text(0.20967741935483872, 0.21428571428571427, 'X[11] <= 262.5\nsquared_error =
3.45\nsamples = 99\nvalue = 66.994'),
 Text(0.20161290322580644, 0.07142857142857142, 'squared_error = 2.532\nsamples
= 80\nvalue = 67.501'),
 Text(0.21774193548387097, 0.07142857142857142, 'squared_error = 1.671\nsamples
= 19\nvalue = 64.858'),
 Text(0.24193548387096775, 0.21428571428571427, 'X[19] <= 94.5\nsquared_error =
11.282\nsamples = 5\nvalue = 73.98'),
 Text(0.23387096774193547, 0.07142857142857142, 'squared_error = 0.0\nsamples =
2\nvalue = 78.0'),
 Text(0.25, 0.07142857142857142, 'squared_error = 0.847\nsamples = 3\nvalue =
71.3'),
 Text(0.3870967741935484, 0.6428571428571429, 'X[27] <= 0.843\nsquared_error =
11.928\nsamples = 383\nvalue = 79.874'),
 Text(0.3225806451612903, 0.5, 'X[25] <= 0.95\nsquared_error = 8.322\nsamples =
131\nvalue = 77.052'),
 Text(0.2903225806451613, 0.35714285714285715, 'X[20] <= 9.43\nsquared_error =
6.067\nsamples = 31\nvalue = 80.513'),
 Text(0.27419354838709675, 0.21428571428571427, 'X[24] <=
17308169.0\nsquared_error = 0.978\nsamples = 26\nvalue = 79.685'),
 Text(0.2661290322580645, 0.07142857142857142, 'squared_error = 0.312\nsamples =
22\nvalue = 79.355'),
 Text(0.28225806451612906, 0.07142857142857142, 'squared_error = 0.75\nsamples =
4\nvalue = 81.5'),
 Text(0.3064516129032258, 0.21428571428571427, 'X[19] <= 98.5\nsquared_error =
10.41\nsamples = 5\nvalue = 84.82'),
 Text(0.29838709677419356, 0.07142857142857142, 'squared_error = 1.556\nsamples
= 3\nvalue = 87.333'),
 Text(0.31451612903225806, 0.07142857142857142, 'squared_error = 0.002\nsamples
= 2\nvalue = 81.05'),
 Text(0.3548387096774194, 0.35714285714285715, 'X[17] <= 4.3\nsquared_error =
4.157\nsamples = 100\nvalue = 75.979'),
 Text(0.3387096774193548, 0.21428571428571427, 'X[28] <= 13.25\nsquared_error =
6.25\nsamples = 2\nvalue = 84.5'),
 Text(0.33064516129032256, 0.07142857142857142, 'squared_error = 0.0\nsamples =
1\nvalue = 82.0'),
 Text(0.3467741935483871, 0.07142857142857142, 'squared_error = 0.0\nsamples =
1\nvalue = 87.0'),
 Text(0.3709677419354839, 0.21428571428571427, 'X[11] <= 129.5\nsquared_error =
2.602\nsamples = 98\nvalue = 75.805'),
 Text(0.3629032258064516, 0.07142857142857142, 'squared_error = 1.704\nsamples =
72\nvalue = 76.371'),
 Text(0.3790322580645161, 0.07142857142857142, 'squared_error = 1.75\nsamples =
26\nvalue = 74.238'),
 Text(0.45161290322580644, 0.5, 'X[11] <= 78.5\nsquared_error = 7.51\nsamples =
252\nvalue = 81.341'),
 Text(0.41935483870967744, 0.35714285714285715, 'X[11] <= 33.0\nsquared_error =
6.509\nsamples = 194\nvalue = 81.983'),
 Text(0.4032258064516129, 0.21428571428571427, 'X[27] <= 0.883\nsquared_error =
4.205\nsamples = 37\nvalue = 80.0'),
 Text(0.3951612903225806, 0.07142857142857142, 'squared_error = 2.423\nsamples =
24\nvalue = 78.942'),
 Text(0.4112903225806452, 0.07142857142857142, 'squared_error = 1.609\nsamples =
13\nvalue = 81.954'),
 Text(0.43548387096774194, 0.21428571428571427, 'X[20] <= 1.64\nsquared_error =
5.907\nsamples = 157\nvalue = 82.45'),
 Text(0.4274193548387097, 0.07142857142857142, 'squared_error = 4.962\nsamples =
12\nvalue = 84.608'),
 Text(0.4435483870967742, 0.07142857142857142, 'squared_error = 5.567\nsamples =
145\nvalue = 82.272'),
 Text(0.4838709677419355, 0.35714285714285715, 'X[5] <= 0.5\nsquared_error =
4.874\nsamples = 58\nvalue = 79.195'),
 Text(0.46774193548387094, 0.21428571428571427, 'X[17] <= 63.1\nsquared_error =
2.515\nsamples = 41\nvalue = 78.549'),
 Text(0.4596774193548387, 0.07142857142857142, 'squared_error = 1.512\nsamples =
40\nvalue = 78.388'),
 Text(0.47580645161290325, 0.07142857142857142, 'squared_error = 0.0\nsamples =
1\nvalue = 85.0'),
 Text(0.5, 0.21428571428571427, 'X[14] <= 6871.232\nsquared_error =
7.13\nsamples = 17\nvalue = 80.753'),
 Text(0.49193548387096775, 0.07142857142857142, 'squared_error = 3.059\nsamples
= 16\nvalue = 80.238'),
 Text(0.5080645161290323, 0.07142857142857142, 'squared_error = -0.0\nsamples =
1\nvalue = 89.0'),
 Text(0.7711693548387096, 0.7857142857142857, 'X[11] <= 333.0\nsquared_error =
50.949\nsamples = 529\nvalue = 57.131'),
 Text(0.6451612903225806, 0.6428571428571429, 'X[18] <= 24.5\nsquared_error =
47.204\nsamples = 320\nvalue = 60.175'),
 Text(0.5806451612903226, 0.5, 'X[28] <= 10.25\nsquared_error = 34.322\nsamples
= 175\nvalue = 63.726'),
 Text(0.5483870967741935, 0.35714285714285715, 'X[22] <= 2.8\nsquared_error =
19.729\nsamples = 89\nvalue = 60.393'),
 Text(0.532258064516129, 0.21428571428571427, 'X[27] <= 0.539\nsquared_error =
8.34\nsamples = 56\nvalue = 62.479'),
 Text(0.5241935483870968, 0.07142857142857142, 'squared_error = 5.557\nsamples =
54\nvalue = 62.146'),
 Text(0.5403225806451613, 0.07142857142857142, 'squared_error = 0.023\nsamples =
2\nvalue = 71.45'),
 Text(0.5645161290322581, 0.21428571428571427, 'X[22] <= 28.8\nsquared_error =
19.153\nsamples = 33\nvalue = 56.855'),
 Text(0.5564516129032258, 0.07142857142857142, 'squared_error = 11.919\nsamples
= 31\nvalue = 57.571'),
 Text(0.5725806451612904, 0.07142857142857142, 'squared_error = 0.022\nsamples =
2\nvalue = 45.75'),
 Text(0.6129032258064516, 0.35714285714285715, 'X[22] <= 2.55\nsquared_error =
26.03\nsamples = 86\nvalue = 67.176'),
 Text(0.5967741935483871, 0.21428571428571427, 'X[11] <= 205.0\nsquared_error =
17.094\nsamples = 59\nvalue = 69.446'),
 Text(0.5887096774193549, 0.07142857142857142, 'squared_error = 9.977\nsamples =
37\nvalue = 71.719'),
 Text(0.6048387096774194, 0.07142857142857142, 'squared_error = 5.759\nsamples =
22\nvalue = 65.623'),
 Text(0.6290322580645161, 0.21428571428571427, 'X[27] <= 0.532\nsquared_error =
9.685\nsamples = 27\nvalue = 62.215'),
 Text(0.6209677419354839, 0.07142857142857142, 'squared_error = 6.26\nsamples =
3\nvalue = 56.1'),
 Text(0.6370967741935484, 0.07142857142857142, 'squared_error = 4.855\nsamples =
24\nvalue = 62.979'),
 Text(0.7096774193548387, 0.5, 'X[22] <= 4.7\nsquared_error = 29.154\nsamples =
145\nvalue = 55.888'),
 Text(0.6774193548387096, 0.35714285714285715, 'X[27] <= 0.39\nsquared_error =
20.192\nsamples = 110\nvalue = 57.587'),
 Text(0.6612903225806451, 0.21428571428571427, 'X[0] <= 0.5\nsquared_error =
16.903\nsamples = 41\nvalue = 54.405'),
 Text(0.6532258064516129, 0.07142857142857142, 'squared_error = 9.209\nsamples =
38\nvalue = 55.182'),
 Text(0.6693548387096774, 0.07142857142857142, 'squared_error = 9.929\nsamples =
3\nvalue = 44.567'),
 Text(0.6935483870967742, 0.21428571428571427, 'X[24] <=
23135387.0\nsquared_error = 12.553\nsamples = 69\nvalue = 59.478'),
 Text(0.6854838709677419, 0.07142857142857142, 'squared_error = 10.747\nsamples
= 59\nvalue = 58.805'),
 Text(0.7016129032258065, 0.07142857142857142, 'squared_error = 4.759\nsamples =
10\nvalue = 63.45'),
 Text(0.7419354838709677, 0.35714285714285715, 'X[14] <= 72.617\nsquared_error =
19.736\nsamples = 35\nvalue = 50.549'),
 Text(0.7258064516129032, 0.21428571428571427, 'X[22] <= 12.65\nsquared_error =
13.553\nsamples = 30\nvalue = 49.407'),
 Text(0.717741935483871, 0.07142857142857142, 'squared_error = 11.739\nsamples =
21\nvalue = 50.833'),
 Text(0.7338709677419355, 0.07142857142857142, 'squared_error = 1.953\nsamples =
9\nvalue = 46.078'),
 Text(0.7580645161290323, 0.21428571428571427, 'X[4] <= 0.5\nsquared_error =
2.068\nsamples = 5\nvalue = 57.4'),
 Text(0.75, 0.07142857142857142, 'squared_error = 0.25\nsamples = 2\nvalue =
55.8'),
 Text(0.7661290322580645, 0.07142857142857142, 'squared_error = 0.436\nsamples =
3\nvalue = 58.467'),
 Text(0.8971774193548387, 0.6428571428571429, 'X[11] <= 504.5\nsquared_error =
20.791\nsamples = 209\nvalue = 52.472'),
 Text(0.8387096774193549, 0.5, 'X[11] <= 379.5\nsquared_error = 13.342\nsamples
= 169\nvalue = 53.816'),
 Text(0.8064516129032258, 0.35714285714285715, 'X[21] <= 71.5\nsquared_error =
11.94\nsamples = 82\nvalue = 55.315'),
 Text(0.7903225806451613, 0.21428571428571427, 'X[18] <= 77.5\nsquared_error =
4.961\nsamples = 48\nvalue = 53.712'),
 Text(0.782258064516129, 0.07142857142857142, 'squared_error = 2.797\nsamples =
21\nvalue = 55.005'),
 Text(0.7983870967741935, 0.07142857142857142, 'squared_error = 4.336\nsamples =
27\nvalue = 52.707'),
 Text(0.8225806451612904, 0.21428571428571427, 'X[14] <= 401.581\nsquared_error
= 13.052\nsamples = 34\nvalue = 57.576'),
 Text(0.8145161290322581, 0.07142857142857142, 'squared_error = 8.683\nsamples =
31\nvalue = 56.952'),
 Text(0.8306451612903226, 0.07142857142857142, 'squared_error = 12.469\nsamples
= 3\nvalue = 64.033'),
 Text(0.8709677419354839, 0.35714285714285715, 'X[14] <= 68.173\nsquared_error =
10.552\nsamples = 87\nvalue = 52.403'),
 Text(0.8548387096774194, 0.21428571428571427, 'X[22] <= 2.7\nsquared_error =
9.106\nsamples = 67\nvalue = 51.482'),
 Text(0.8467741935483871, 0.07142857142857142, 'squared_error = 6.571\nsamples =
11\nvalue = 48.173'),
 Text(0.8629032258064516, 0.07142857142857142, 'squared_error = 7.03\nsamples =
56\nvalue = 52.132'),
 Text(0.8870967741935484, 0.21428571428571427, 'X[11] <= 454.0\nsquared_error =
3.026\nsamples = 20\nvalue = 55.49'),
 Text(0.8790322580645161, 0.07142857142857142, 'squared_error = 1.935\nsamples =
11\nvalue = 56.555'),
 Text(0.8951612903225806, 0.07142857142857142, 'squared_error = 1.281\nsamples =
9\nvalue = 54.189'),
 Text(0.9556451612903226, 0.5, 'X[22] <= 5.45\nsquared_error = 12.375\nsamples =
40\nvalue = 46.793'),
 Text(0.9354838709677419, 0.35714285714285715, 'X[26] <= 1.15\nsquared_error =
17.87\nsamples = 5\nvalue = 41.92'),
 Text(0.9193548387096774, 0.21428571428571427, 'X[22] <= 1.8\nsquared_error =
1.823\nsamples = 2\nvalue = 46.65'),
 Text(0.9112903225806451, 0.07142857142857142, 'squared_error = 0.0\nsamples =
1\nvalue = 48.0'),
 Text(0.9274193548387096, 0.07142857142857142, 'squared_error = 0.0\nsamples =
1\nvalue = 45.3'),
 Text(0.9516129032258065, 0.21428571428571427, 'X[12] <= 26.0\nsquared_error =
3.709\nsamples = 3\nvalue = 38.767'),
 Text(0.9435483870967742, 0.07142857142857142, 'squared_error = 0.0\nsamples =
1\nvalue = 36.3'),
 Text(0.9596774193548387, 0.07142857142857142, 'squared_error = 1.0\nsamples =
2\nvalue = 40.0'),
 Text(0.9758064516129032, 0.35714285714285715, 'X[23] <= 19.653\nsquared_error =
7.714\nsamples = 35\nvalue = 47.489'),
 Text(0.967741935483871, 0.21428571428571427, 'squared_error = 0.0\nsamples =
1\nvalue = 58.0'),
 Text(0.9838709677419355, 0.21428571428571427, 'X[11] <= 600.0\nsquared_error =
4.595\nsamples = 34\nvalue = 47.179'),
 Text(0.9758064516129032, 0.07142857142857142, 'squared_error = 3.727\nsamples =
18\nvalue = 48.511'),
 Text(0.9919354838709677, 0.07142857142857142, 'squared_error = 1.333\nsamples =
16\nvalue = 45.681')]
```

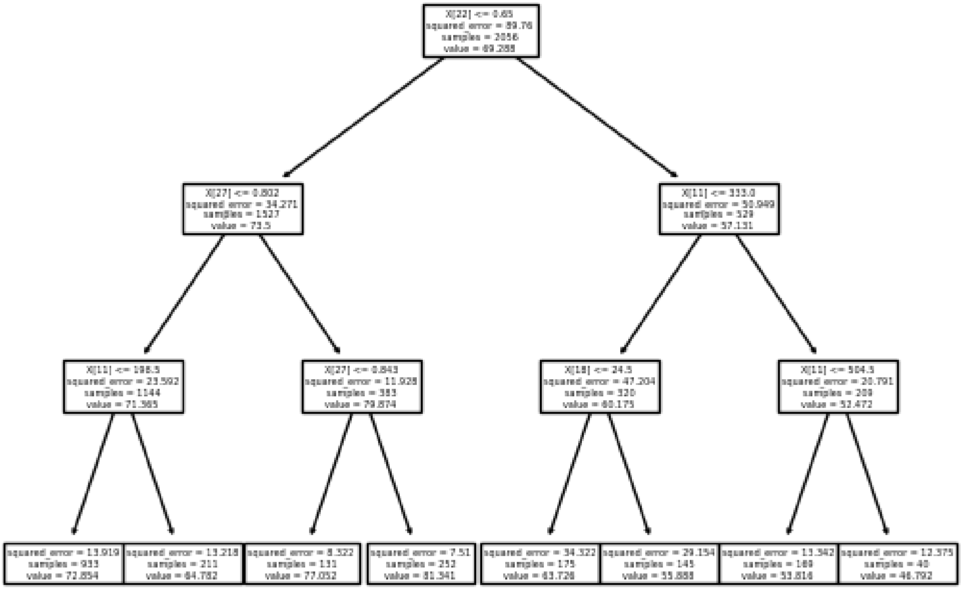


[45]:

```
dt_new = DecisionTreeRegressor(random_state=0, max_depth=3)
dt_new.fit(X_train, y_train)
tree.plot_tree(dt_new)
```

[45]:

```
[Text(0.5, 0.875, 'X[22] <= 0.65\nsquared_error = 89.76\nsamples = 2056\nvalue =
69.288'),
 Text(0.25, 0.625, 'X[27] <= 0.802\nsquared_error = 34.271\nsamples =
1527\nvalue = 73.5'),
 Text(0.125, 0.375, 'X[11] <= 198.5\nsquared_error = 23.592\nsamples =
1144\nvalue = 71.365'),
 Text(0.0625, 0.125, 'squared_error = 13.919\nsamples = 933\nvalue = 72.854'),
 Text(0.1875, 0.125, 'squared_error = 13.218\nsamples = 211\nvalue = 64.782'),
 Text(0.375, 0.375, 'X[27] <= 0.843\nsquared_error = 11.928\nsamples =
383\nvalue = 79.874'),
 Text(0.3125, 0.125, 'squared_error = 8.322\nsamples = 131\nvalue = 77.052'),
 Text(0.4375, 0.125, 'squared_error = 7.51\nsamples = 252\nvalue = 81.341'),
 Text(0.75, 0.625, 'X[11] <= 333.0\nsquared_error = 50.949\nsamples = 529\nvalue
= 57.131'),
 Text(0.625, 0.375, 'X[18] <= 24.5\nsquared_error = 47.204\nsamples = 320\nvalue
= 60.175'),
 Text(0.5625, 0.125, 'squared_error = 34.322\nsamples = 175\nvalue = 63.726'),
 Text(0.6875, 0.125, 'squared_error = 29.154\nsamples = 145\nvalue = 55.888'),
 Text(0.875, 0.375, 'X[11] <= 504.5\nsquared_error = 20.791\nsamples =
209\nvalue = 52.472'),
 Text(0.8125, 0.125, 'squared_error = 13.342\nsamples = 169\nvalue = 53.816'),
 Text(0.9375, 0.125, 'squared_error = 12.375\nsamples = 40\nvalue = 46.792')]
```


X[22] <= 0.65
squared_error = 89.76
samples = 2056
value = 69.288
X[27] <= 0.802
squared_error = 34.271
samples = 1527
value = 73.5
X[11] <= 333.0
squared_error = 50.949
samples = 529
value = 57.131
X[11] <= 198.5
squared_error = 23.592
samples = 1144
value = 71.365
X[27] <= 0.843
squared_error = 11.928
samples = 383
value = 79.874
X[18] <= 24.5
squared_error = 47.204
samples = 320
value = 60.175
X[11] <= 504.5
squared_error = 20.791
samples = 209
value = 52.472
squared_error = 13.919
samples = 933
value = 72.854
squared_error = 13.218
samples = 211
value = 64.782
squared_error = 8.322
samples = 131
value = 77.052
squared_error = 7.51
samples = 252
value = 81.341
squared_error = 34.322
samples = 175
value = 63.726
squared_error = 29.154
samples = 145
value = 55.888
squared_error = 13.342
samples = 169
value = 53.816
squared_error = 12.375
samples = 40
value = 46.792