#### PRACTICAL-5

# NAME-PRIYANSH PANCHAL ROLLNO-24BEE106

```
import pandas as pd
import numpy as np
data = pd.read_csv("BostonHousing.csv", header='infer')
X = data.iloc[:, :-1] # All columns except the last one
y = data.iloc[:, -1] # Last column
print("Total Rows:", data.shape[0])
test_split = 0.1
indices = np.random.permutation(data.shape[0])
n_train = int((1 - test_split) * data.shape[0])
X_train = X.iloc[indices[:n_train]]
y_train = y.iloc[indices[:n_train]]
X_test = X.iloc[indices[n_train:]]
y_test = y.iloc[indices[n_train:]]
print("Shape of X_train:", X_train.shape)
print("Shape of X_test:", X_test.shape)
train indices = set(indices[:n train])
test_indices = set(indices[n_train:])
print("Union of train and test indices:", train_indices.union(test_indices))
print("Intersection between train and test indices:", train_indices.intersection(test_indices))
```

### **OUTPUT-**

Total Rows: 506

Shape of X\_train: (455, 13) Shape of X test: (51, 13)

Union of train and test indices: {np.int32(0), np.int32(1), np.int32(2), np.int32(3), np.int32(4), np.int32(5), np.int32(6), np.int32(7), np.int32(8), np.int32(8 p.int32(9), np.int32(10), np.int32(11), np.int32(12), np.int32(13), np.int32(14), np.int32(15), np.int32(16), np.int32(17), np.int32(18), np.int32(19), np.i p.int32(20), np.int32(21), np.int32(22), np.int32(23), np.int32(24), np.int32(25), np.int32(26), np.int32(27), np.int32(28), np.int32(29), np.int32(30), np.int32(31), np.int32(32), np.int32(33), np.int32(34), np.int32(35), np.int32(36), np.int32(37), np.int32(38), np.int32(39), np.int32(40), np.int32(41) np.int32(42), np.int32(43), np.int32(44), np.int32(45), np.int32(47), np.int32(48), np.int32(49), np.int32(52) np.int32(53), np.int32(54), np.int32(55), np.int32(56), np.int32(57), np.int32(58), np.int32(59), np.int32(60), np.int32(61), np.int32(62), np.int32(63) np.int32(64), np.int32(65), np.int32(66), np.int32(67), np.int32(68), np.int32(69), np.int32(70), np.int32(71), np.int32(72), np.int32(73), np.int32(74) np.int32(75), np.int32(76), np.int32(77), np.int32(78), np.int32(79), np.int32(80), np.int32(81), np.int32(82), np.int32(83), np.int32(84), np.int32(85) np.int32(86), np.int32(87), np.int32(88), np.int32(99), np.int32(90), np.int32(91), np.int32(92), np.int32(93), np.int32(94), np.int32(95), np.int32(96) np.int32(97), np.int32(98), np.int32(99), np.int32(100), np.int32(101), np.int32(102), np.int32(103), np.int32(104), np.int32(105), np.int32(106), np.int32(106) 32(107), np.int32(108), np.int32(109), np.int32(110), np.int32(111), np.int32(112), np.int32(113), np.int32(114), np.int32(115), np.int32(116), np.int32 (117), np.int32(118), np.int32(119), np.int32(120), np.int32(121), np.int32(122), np.int32(123), np.int32(124), np.int32(125), np.int32(126), 7), np.int32(128), np.int32(129), np.int32(130), np.int32(131), np.int32(132), np.int32(133), np.int32(134), np.int32(135), np.int32(136), np.int32(137) np.int32(138), np.int32(139), np.int32(140), np.int32(141), np.int32(142), np.int32(143), np.int32(144), np.int32(145), np.int32(146), np.int32(147), np.int int32(148), np.int32(149), np.int32(150), np.int32(151), np.int32(152), np.int32(153), np.int32(154), np.int32(155), np.int32(156), np.int32(157), np.int 32(158), np.int32(159), np.int32(160), np.int32(161), np.int32(162), np.int32(163), np.int32(164), np.int32(165), np.int32(166), np.int32(167), np.int32 (168), np.int32(169), np.int32(170), np.int32(171), np.int32(172), np.int32(173), np.int32(174), np.int32(175), np.int32(176), np.int32(177), np.int32(177), np.int32(178), 8), np.int32(179), np.int32(180), np.int32(181), np.int32(182), np.int32(183), np.int32(184), np.int32(185), np.int32(186), np.int32(187), np.int32(188) np.int32(189), np.int32(190), np.int32(191), np.int32(192), np.int32(193), np.int32(194), np.int32(195), np.int32(196), np.int32(197), np.int32(198), np int32(199), np.int32(200), np.int32(201), np.int32(202), np.int32(203), np.int32(204), np.int32(205), np.int32(206), np.int32(207), np.int32(208), np.int 32(209), np.int32(210), np.int32(211), np.int32(212), np.int32(213), np.int32(214), np.int32(215), np.int32(216), np.int32(217), np.int32(218), np.int32 (219), np.int32(220), np.int32(221), np.int32(222), np.int32(223), np.int32(224), np.int32(225), np.int32(226), np.int32(227), np.int32(228), np.int32(230), np.int32(231), np.int32(232), np.int32(233), np.int32(234), np.int32(235), np.int32(236), np.int32(237), np.int32(238), np.int32(239) np.int32(240), np.int32(241), np.int32(242), np.int32(243), np.int32(244), np.int32(245), np.int32(246), np.int32(247), np.int32(248), np.int32(249), np.int int32(250), np.int32(251), np.int32(252), np.int32(253), np.int32(254), np.int32(255), np.int32(256), np.int32(257), np.int32(258), np.int32(259), np.int32( 32(260), np.int32(261), np.int32(262), np.int32(263), np.int32(264), np.int32(265), np.int32(266), np.int32(267), np.int32(268), np.int32(269), np.int32 (270), np.int32(271), np.int32(272), np.int32(273), np.int32(274), np.int32(275), np.int32(276), np.int32(277), np.int32(278), np.int32(279), 0), np.int32(281), np.int32(282), np.int32(283), np.int32(284), np.int32(285), np.int32(286), np.int32(287), np.int32(288), np.int32(289), np.int32(290) np.int32(291), np.int32(292), np.int32(293), np.int32(294), np.int32(295), np.int32(296), np.int32(297), np.int32(298), np.int32(299), np.int32(300), np.int int32(301), np.int32(302), np.int32(303), np.int32(304), np.int32(305), np.int32(306), np.int32(307), np.int32(308), np.int32(309), np.int32(310), np.int32(300) 32(311), np.int32(312), np.int32(313), np.int32(314), np.int32(315), np.int32(316), np.int32(317), np.int32(318), np.int32(319), np.int32(320), np.int32 (321), np.int32(322), np.int32(323), np.int32(324), np.int32(325), np.int32(326), np.int32(327), np.int32(328), np.int32(329), np.int32(330), np.int32(326) np.int32(332), np.int32(333), np.int32(334), np.int32(335), np.int32(336), np.int32(337), np.int32(338), np.int32(339), np.int32(340), np.int32(341)

## SPLITTING WITH USAGE OF LIBRARY-

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
# Load the dataset
data = pd.read_csv("BostonHousing.csv", header='infer')
# Separate features (X) and target (y)
X = data.iloc[:, :-1].values # All columns except the last one
y = data.iloc[:, -1].values # Last column as target
# Get the total number of rows
nrows = data.shape[0]
print("Total Rows:", nrows)
# Ask for test split ratio
test_split =0.2
# Split the data into training and testing sets
X train, X test, y train, y test = train test split(X, y, test size=test split)
# Print the shapes of the train and test sets
print("Shapes of the splits:")
print(f"X_train shape: {X_train.shape}, y_train shape: {y_train.shape}")
print(f"X test shape: {X test.shape}, y test shape: {y test.shape}")
Total Rows: 506
Shapes of the splits:
X_train shape: (404, 13), y_train shape: (404,)
X_test shape: (102, 13), y_test shape: (102,)
```

# SPLITTING WITH DIFFERENT TEST SIZE

Outputs-

Size=0.5

```
Total Rows: 506

Shapes of the splits:

X_train shape: (253, 13), y_train shape: (253,)

X_test shape: (253, 13), y_test shape: (253,)
```

```
Total Rows: 506

Shapes of the splits:

X_train shape: (101, 13), y_train shape: (101,)

X_test shape: (405, 13), y_test shape: (405,)
```