SET 5

```
1) from sklearn.datasets import load_boston
   boston = load_boston()
   import pandas as pd
data = pd.DataFrame(boston.data, columns = boston.feature_
                                                  names)
data['MEDV'] = pd.DataFrame(boston.target)
data
pd.DataFrame(data.corr().round(3))
x = data[['CHAS', 'NOX']]
 x
 y = data['MEDV'] , y
from sklearn.model_selection import train_test_split
x = pd.DataFrame(x)
y = pd.DataFrame(y)
x_train, x_test, y_train, y_test = train_test_split  0.3
                                  (x, y, test_size = 30/70)

y_train
from sklearn.linear_model import Ridge
ridge1 = Ridge(alpha = 1)
ridge1.fit(x_train, y_train) , y_pred1 = ridge1.predict
                                              (x_test)
np.sqrt(mean_squared_error(y_test, y_pred1)
ridge2 = Ridge(alpha = 100)
ridge2.fit(x_train, y_train)
y_pred2 = ridge2.predict(x_test)
np.sqrt(mean_squared_error(y_test, y_pred2))
ridge2.score(x_test, y_test)
from sklearn.~~to~~linear_model import Lasso.
Lasso1 = Lasso(alpha = 4)
lasso.fit(x_train, y_train)
y_predL1 = Lasso1.predict(x_test)
np.sqrt(mean_squared_error~~8~~(y_test, y_predL1))
Lasso1.score(x_test, y_test)
2) from sklearn.datasets import load_digits
digits = load_digits()
print(digits.DESCR)
digits.data , digits.data.shape , d=digits.data[0:500]
d.shape , digits.target.shape
image = digits.data[102]
print(image), import numpy as np, np.reshape
import matplotlib.pyplot as plt              (image,(8,8))
plt.~~ta~~imshow(np.reshape(image, (8,8)), cmap = 'gray')
from sklearn.model_selection import train_test_split.
x_train, x_test, y_train, y_test = train_test_split
                             (digits.data, digits.target,
                               test_size = ~~8~~0.28)
```

ASSIGNMENT ON DEPARTMENT : CSE

```
from sklearn. tree import Decision Tree Classifier
from sklearn. datasets import load_digits.
digits = load_digits()
x_train, x_test, y_train, ~~y~~ y_test = train_test_split
 (digits. data, digits. target, test_size = 0.25)
dt2 = Decision Tree Classifier (criterion = "entropy")
dt2. fit (x_train, ~~y_test)~~ y_train)
dt2. score (x_test, y-test)
dt3 = Decision Tree Classifier (max_depth = 50)
dt3. fit (x_train, y_train)
dt3. score (x_test, y-test)
from sklearn. ensemble import RandomForest Regressor
rf1 = RandomForest Regressor ()
rf1. fit(x_train, y-train)
rf1. score (x_test, y-test), y-pred2 = rf1. predict (x_test)
np.sqrt(mean_squared_error (y-test, y-pred2))
3) from sklearn. datasets import load_digits
digits = load_digits (), digits. data. shape
from sklearn. model_selection import train_test_split
x_train, ~~y_train~~, x_test, y_train, y_test = train_test
  split (digits. data, digits. target, test_size= 0.3)
~~indexList~~ = np. arrange
import numpy as np.
index = np. arrange (len (x_train)).
print (index)
x_labelled = x_train [indexList [:319]]
y_labelled = y_train [indexList [:319]]
nonLabelled Indices = indexList [319:]
y_train_nonLabel = np. copy (y_train)
y_train_nonlabel [nonlabelled Indices] = -1
from sklearn. semi_supervised import Label Propagation
lp = LabelPropagation ()
lp. fit(x_train, y_train_nonlabel), lp. score (x_test, y-test)
from sklearn. semi_supervised import LabelSpreading
ls = LabelSpreading (gamma = 0.3)
ls. fit (x_train, y_train_non Label)
ls. score (x_test, y_test)
y_labelled = ls. transduction_ [nonLabelled Indices].
from sklearn. metrics import confusion_matrix
conf = confusion_matrix (y_train [nonLabelled Indices],
  labels = ls. classes
```