**PART A**

(PART A : TO BE REFFERED BY STUDENTS)

**EXPERIMENT NO. 7**

**A.1 AIM: -** To Implementing Predicting Energy Efficiency for Residential Buildings

**A.2 Prerequisite**

* Different programming language (Python or Java), Understanding of Machine Learning Algorithms, Machine Learning Algorithms

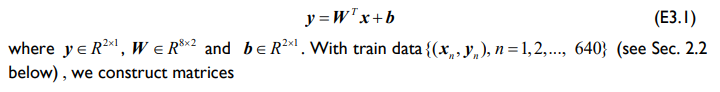
**A.3 Outcome**

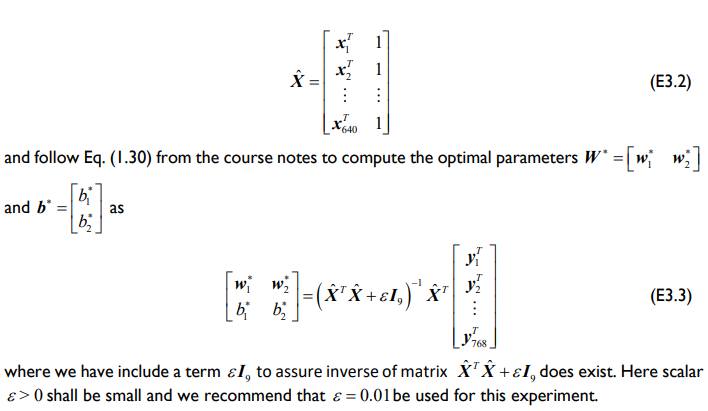
After successful completion of this experiment students will be able to Optimize the problem.

**A.4 Theory**

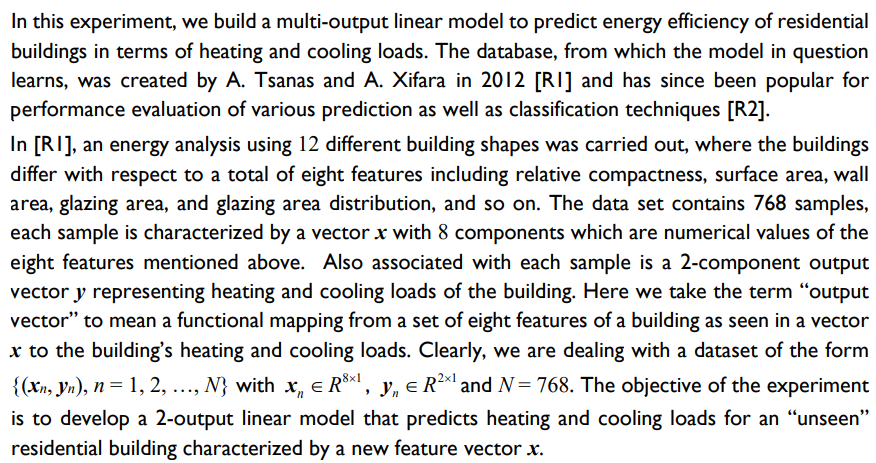
**Multi-output linear model for prediction**

The linear model of interest is given by





**A5. Task**

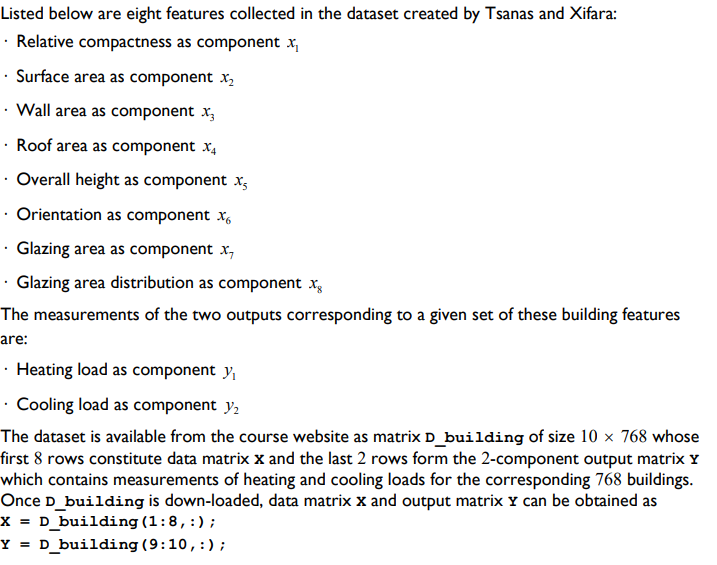


﻿

In this experiment, we build a multi-output linear model to predict energy efficiency of residential buildings in terms of heating and cooling loads. The database, from which the model in question learns, was created by A. Tsanas and A. Xifara in 2012 [RI] and has since been popular for performance evaluation of various prediction as well as classification techniques [R2].

In [RI], an energy analysis using 12 different building shapes was carried out, where the buildings differ with respect to a total of eight features including relative compactness, surface area, wall area, glazing area, and glazing area distribution, and so on. The data set contains 768 samples, each sample is characterized by a vector x with 8 components which are numerical values of the eight features mentioned above. Also associated with each sample is a 2-component output vector y representing heating and cooling loads of the building. Here we take the term "output vector" to mean a functional mapping from a set of eight features of a building as seen in a vector x to the building's heating and cooling loads. Clearly, we are dealing with a dataset of the form {(xn, yn), n = 1, 2, ..., N} with x1 = R3×1, y1 € R2×1 and N = 768. The objective of the experiment is to develop a 2-output linear model that predicts heating and cooling loads for an “unseen" residential building characterized by a new feature vector x.

2x1



﻿

Listed below are eight features collected in the dataset created by Tsanas and Xifara:

· Relative compactness as component x

· Surface area as component x2

· Wall area as component x,

· Roof area as component x

Overall height as component x,

Orientation as component x

⚫ Glazing area as component x,

Glazing area distribution as component x

The measurements of the two outputs corresponding to a given set of these building features

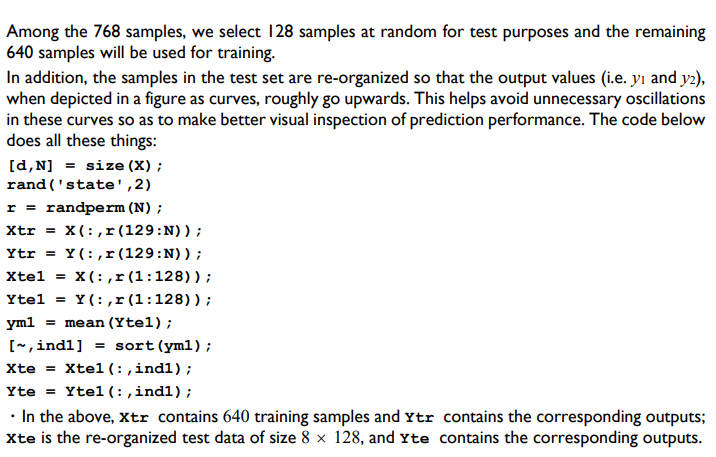
are:

Heating load as component y

Cooling load as component y2

The dataset is available from the course website as matrix D\_building of size 10 x 768 whose first 8 rows constitute data matrix x and the last 2 rows form the 2-component output matrix Y which contains measurements of heating and cooling loads for the corresponding 768 buildings. Once D\_building is down-loaded, data matrix x and output matrix Y can be obtained as X = D\_building (1:8, :);

Y = D\_building (9:10, :);



﻿

Among the 768 samples, we select 128 samples at random for test purposes and the remaining 640 samples will be used for training.

In addition, the samples in the test set are re-organized so that the output values (i.e. yi and y2), when depicted in a figure as curves, roughly go upwards. This helps avoid unnecessary oscillations in these curves so as to make better visual inspection of prediction performance. The code below does all these things:

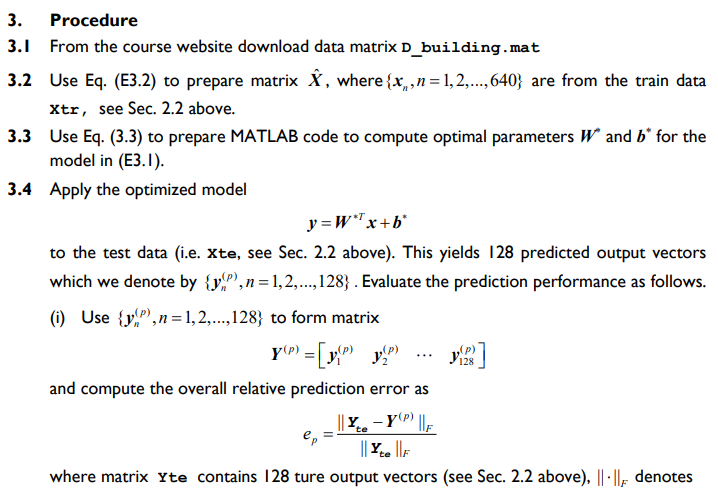
[d, N] = size (X); rand('state', 2) r = randperm (N);

Xtr = X(:,r (129:N)); Ytr Y(:,r (129:N)); Xtel = X(:,r (1:128)); Ytel = Y(:,r(1:128)); ym1 = mean (Ytel);

[~, ind1] = sort (yml); Xte = Xtel (:, indl);

Yte = Ytel (:, indl);

⚫ In the above, xtr contains 640 training samples and Ytr contains the corresponding outputs; xte is the re-organized test data of size 8 x 128, and yte contains the corresponding outputs.



﻿

3.

Procedure

3.1 From the course website download data matrix D\_building.mat

3.2 Use Eq. (E3.2) to prepare matrix ✰, where {x,, n = 1, 2,..., 640} are from the train data Xtr, see Sec. 2.2 above.

3.3 Use Eq. (3.3) to prepare MATLAB code to compute optimal parameters W and b\* for the model in (E3.1).

3.4 Apply the optimized model

y=WTx+b\*

to the test data (i.e. xte, see Sec. 2.2 above). This yields 128 predicted output vectors which we denote by {y), n = 1,2,..., 128}. Evaluate the prediction performance as follows.

(i) Use {y), n = 1, 2,...,128} to form matrix

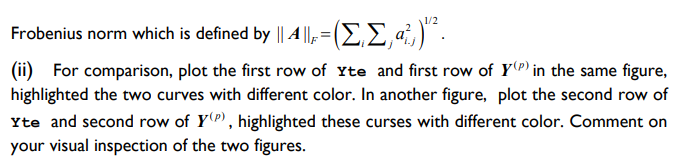
y =[y y

]

and compute the overall relative prediction error as

Yte - YP F ||Yte F

where matrix Yte contains 128 ture output vectors (see Sec. 2.2 above), || ||, denotes



﻿

1/2

Frobenius norm which is defined by || A4 ||,= (Σ,Σ,a,)2.

(ii) For comparison, plot the first row of Yte and first row of YP) in the same figure, highlighted the two curves with different color. In another figure, plot the second row of Yte and second row of Y(P), highlighted these curses with different color. Comment on your visual inspection of the two figures.

Links: <http://archive.ics.uci.edu/ml/datasets/Energy+efficiency?ref=datanews.io>

Or

<https://www.kaggle.com/datasets/elikplim/eergy-efficiency-dataset>

PART B

(PART B : TO BE COMPLETED BY STUDENTS)

***(Students must submit the soft copy as per following segments within two hours of the practical. The soft copy must be uploaded on the Blackboard or emailed to the concerned lab in charge faculties at the end of the practical in case there is no Black board access available)***

|  |  |
| --- | --- |
| Roll No. C050 | Name: Nisha Kini |
| Class : BTI B | Batch : EB2 |
| Date of Experiment: 20.2.24 | Date of Submission: 20.2.24 |
| Grade : |  |

**B.1 Documentation written by student:**

from ucimlrepo import fetch\_ucirepo

# fetch dataset

energy\_efficiency = fetch\_ucirepo(id=242)

# data (as pandas dataframes)

X = energy\_efficiency.data.features

y = energy\_efficiency.data.targets

# metadata

print(energy\_efficiency.metadata)

# variable information

print(energy\_efficiency.variables)

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

# Load the dataset from CSV

df = pd.read\_excel('/content/ENB2012\_data.xlsx')  # Assuming 'data.csv' is the correct file name

# Extract features (X) and outputs (Y)

X = df.iloc[:, :8].values.T

Y = df.iloc[:, 8:].values.T

# Randomly select 128 samples for testing

d, N = X.shape

np.random.seed(2)

r = np.random.permutation(N)

Xtr = X[:, r[128:]]

Ytr = Y[:, r[128:]]

Xtel = X[:, r[:128]]

Ytel = Y[:, r[:128]]

# Calculate mean of Ytel and reorganize the test set

ym1 = np.mean(Ytel, axis=1)

ind1 = np.argsort(ym1)

Xte = Xtel[:, ind1]

Yte = Ytel[:, ind1]

# Prepare matrix ✰ using Eq. (E3.2)

Xtr\_tilde = np.vstack([Xtr, np.ones((1, Xtr.shape[1]))])

# Compute optimal parameters W and b\* using Eq. (3.3)

W = np.linalg.lstsq(Xtr\_tilde.T, Ytr.T, rcond=None)[0]

# Apply the optimized model to the test data

Xte\_tilde = np.vstack([Xte, np.ones((1, Xte.shape[1]))])

Yp = np.dot(W.T, Xte\_tilde)

# Calculate the overall relative prediction error

error = np.linalg.norm(Yte - Yp, 'fro') / np.linalg.norm(Yte, 'fro')

# Plot the first row of Yte and Yp

plt.figure(figsize=(10, 8))

plt.subplot(2, 1, 1)

plt.plot(Yte[0, :], 'b', linewidth=2)

plt.plot(Yp[0, :], 'r--', linewidth=2)

plt.title('Heating Load Prediction')

plt.legend(['True Heating Load', 'Predicted Heating Load'])

# Plot the second row of Yte and Yp

plt.subplot(2, 1, 2)

plt.plot(Yte[1, :], 'g', linewidth=2)

plt.plot(Yp[1, :], 'm--', linewidth=2)

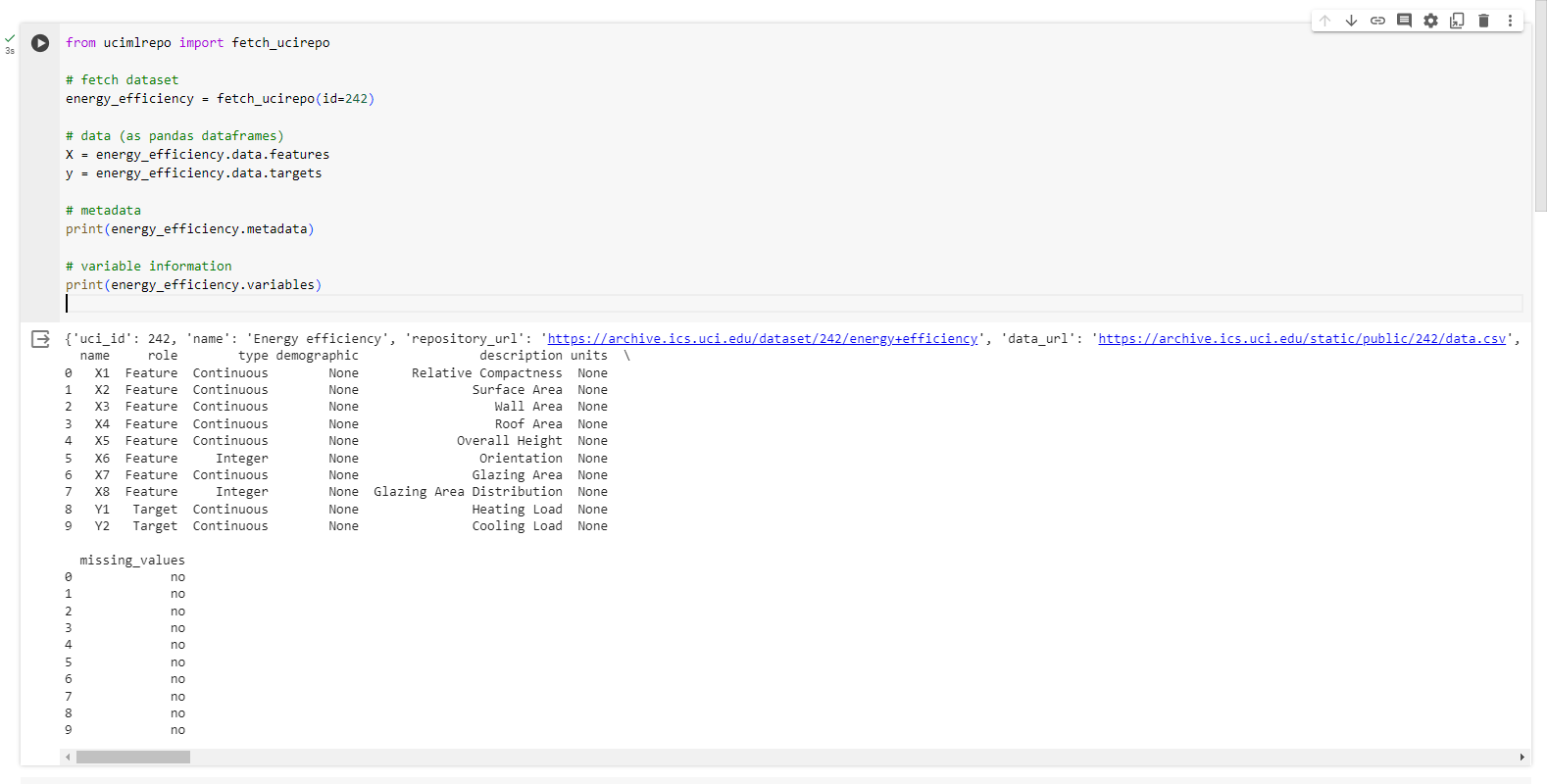
plt.title('Cooling Load Prediction')

plt.legend(['True Cooling Load', 'Predicted Cooling Load'])

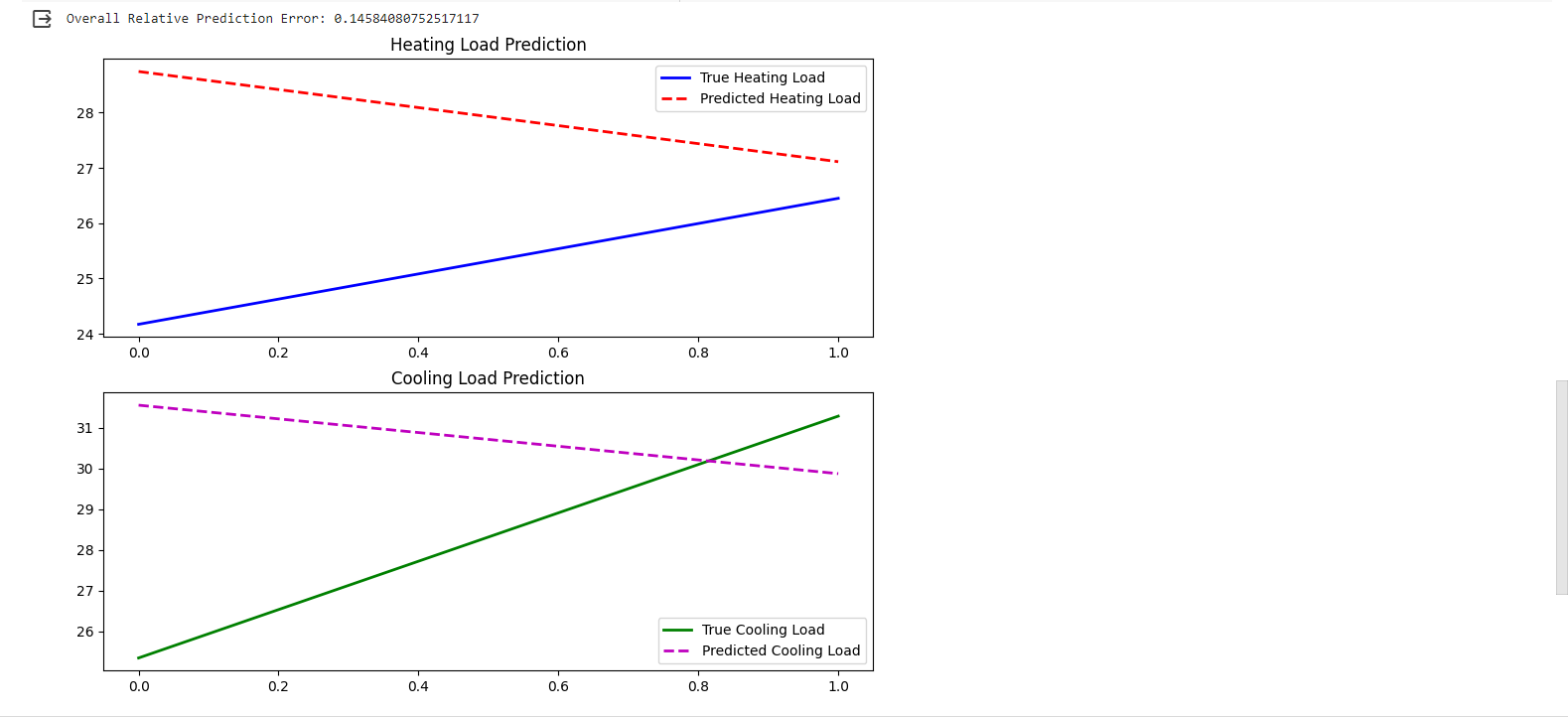
# Display the overall relative prediction error

print(f'Overall Relative Prediction Error: {error}')

plt.show()







**B.2 Observations and learning:**

1. \*\*Data Loading and Preprocessing:\*\*

- The code effectively loads the dataset from an Excel file using Pandas.

- It extracts the features (X) and outputs (Y) from the DataFrame.

2. \*\*Data Splitting:\*\*

- The dataset is split into training and testing sets with 640 and 128 samples, respectively.

- Randomization ensures a diverse representation in both sets.

3. \*\*Model Training and Testing:\*\*

- The code prepares the feature matrix using the extended matrix notation.

- It computes the optimal parameters W and b\* using the least squares method.

- The trained model is then applied to the test data to predict the outputs.

4. \*\*Performance Evaluation:\*\*

- The code calculates the overall relative prediction error, providing a quantitative measure of the model's accuracy.

- It visualizes the predicted and true values of both heating and cooling loads, allowing for a qualitative assessment.

5. \*\*Plotting and Visualization:\*\*

- The code uses Matplotlib to create informative plots, illustrating the model's predictions against the actual values.

- The plots are organized, with clear labels and legends, facilitating easy interpretation.

**B.3 Conclusion:**

The provided code demonstrates a comprehensive approach to building and evaluating a linear model for predicting heating and cooling loads of residential buildings. The model's performance is assessed through both numerical metrics and visual representations, providing a well-rounded evaluation. The use of randomization in data splitting contributes to the robustness of the model evaluation. Overall, the code showcases good practices in data preprocessing, model training, and result interpretation, making it a valuable resource for similar predictive modeling tasks.