Data-X Spring 2019: Homework 05

Linear regression & Logistic regression ¶

Name: Shun Lin

SID: 26636176

In this homework, you will do some exercises on prediction using sklearn.

REMEMBER TO DISPLAY ALL OUTPUTS. If the question asks you to do something, make sure to print your results.

Part 1 - Regression

Data:

Data Source: Data file is uploaded to bCourses and is named: **Energy.csv** (Link in the Assignment details page on Bcourses)

The dataset was created by Angeliki Xifara (Civil/Structural Engineer) and was processed by Athanasios Tsanas, Oxford Centre for Industrial and Applied Mathematics, University of Oxford, UK).

Data Description:

The dataset contains eight attributes of a building (or features, denoted by X1...X8) and response being the heating load on the building, y1.

- X1 Relative Compactness
- X2 Surface Area
- X3 Wall Area
- X4 Roof Area
- X5 Overall Height
- X6 Orientation
- X7 Glazing Area
- X8 Glazing Area Distribution
- y1 Heating Load

Q1.1

Read the data file from the csv.

Print the count of NaN values for each attribute in the dataset.

Print the Range (min, max) and percentiles (25th, 50th, and 75th) of each attribute in the dataset

In [1]:

```
# your code
# your code

# Load required modules
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

data=pd.read_csv('Energy.csv')

sum_of_nans = sum(len(data) - data.count())
print("There are " + str(sum_of_nans) + " Nan values in the dataframe")
print('Number of NaNs in the dataframe:\n',data.isnull().sum())

data.describe(include='all')
```

```
There are 0 Nan values in the dataframe
Number of NaNs in the dataframe:
 X1    0
X2    0
X3    0
X4    0
X5    0
X6    0
X7    0
X8    0
Y1    0
dtype: int64
```

Out[1]:

	X1	X2	X3	X4	X5	X6	X7	)
count	768.000000	768.000000	768.000000	768.000000	768.00000	768.000000	768.000000	768.000(
mean	0.764167	671.708333	318.500000	176.604167	5.25000	3.500000	0.234375	2.812!
std	0.105777	88.086116	43.626481	45.165950	1.75114	1.118763	0.133221	1.550!
min	0.620000	514.500000	245.000000	110.250000	3.50000	2.000000	0.000000	0.000(
25%	0.682500	606.375000	294.000000	140.875000	3.50000	2.750000	0.100000	1.750(
50%	0.750000	673.750000	318.500000	183.750000	5.25000	3.500000	0.250000	3.000(
75%	0.830000	741.125000	343.000000	220.500000	7.00000	4.250000	0.400000	4.000(
max	0.980000	808.500000	416.500000	220.500000	7.00000	5.000000	0.400000	5.000(

REGRESSION:

Using the data, we want to predict "Heating load". The output variable is continuous. Hence, we need to use a regression algorithm.

Q 1.2:

Split the dataset randomly into train and test. Train a **Linear Regression** model on 80% of the data (80-20 split). What is the intercept and coefficient values?

In [2]:
```
# your code
from sklearn.model_selection import train_test_split
from sklearn.utils import shuffle
from sklearn import linear_model

# shuffle data and drop Nan values
data= shuffle(data).reset_index(drop=True)

# get X and Y
X=data.iloc[:,:-1]
Y=data['Y1']

# split the train and test data
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, ra
print ('Number of samples in training data:',len(x_train))
print ('Number of samples in test data:',len(x_test))

# building a linear regression model
LinearRegressionModel= linear_model.LinearRegression()

# fit the model
LinearRegressionModel.fit(x_train, y_train)

# The coefficients
print('Coefficients:', LinearRegressionModel.coef_)
print('Intercepts:', LinearRegressionModel.intercept_)
```

```
Number of samples in training data: 614
Number of samples in test data: 154
Coefficients: [-6.74246975e+01 -1.18372098e+12  1.18372098e+12  2.3674419
6e+12
  4.11956879e+00  1.31071107e-02  1.99659409e+01  1.90180564e-01]
Intercepts: 88.93754071661238
```

Q.1.3:

Create a function which takes arrays of prediction and actual values of the output as parameters to calculate **'Root Mean Square error'** (RMSE) metric:

1. Use the function to calculate the training RMSE
2. Use the function to calculate the test RMSE

In [3]:
```
# your code
import numpy as np

def RMSE(pred_y, true_y):
    return np.sqrt(((pred_y - true_y) ** 2).mean())

train_RMSE = RMSE(LinearRegressionModel.predict(x_train), y_train)
test_RMSE = RMSE(LinearRegressionModel.predict(x_test), y_test)

print("The training RMSE is: " + str(train_RMSE))
print("The testing RMSE is: " + str(test_RMSE))
```

```
The training RMSE is: 2.8864199100237773
The testing RMSE is: 3.100140402234829
```

Q1.4:

Let's see the effect of amount of data on the performance of prediction model. Use varying amounts of data (100,200,300,400,500,all) from the training data you used previously to train different regression models. Report training error and test error in each case. Test data is the same as above for all these cases.

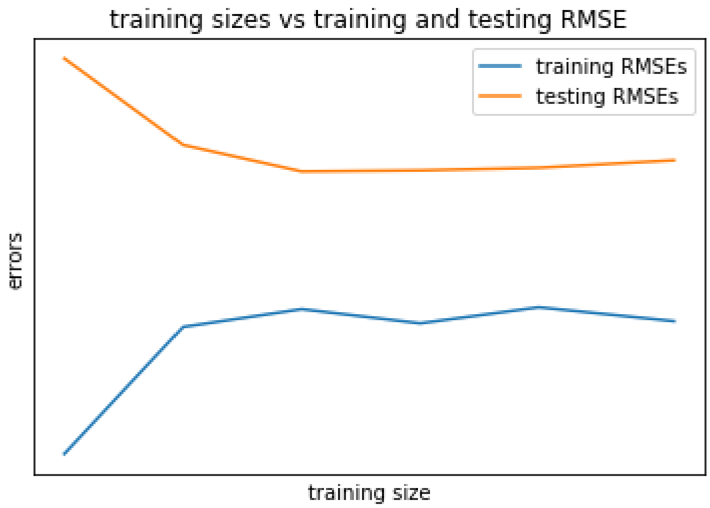
Plot error rates vs number of training examples. Both the training error and the test error should be plotted. Comment on the relationship you observe between the amount of data used to train the model and the test accuracy of the model.

Hint: Use array indexing to choose varying data amounts

In [4]:
```
import matplotlib.pyplot as plt

# your code
training_sizes = [100,200,300,400,500,614]

# initialize training and testing errors
training_errors = []
testing_errors = []

for training_size in training_sizes:
    print("training size: " + str(training_size))
    small_x_train = x_train[:training_size]
    small_y_train = y_train[:training_size]

    LinearRegressionModel= linear_model.LinearRegression()
    LinearRegressionModel.fit(small_x_train, small_y_train)
    train_RMSE = RMSE(LinearRegressionModel.predict(small_x_train), small_y
    test_RMSE = RMSE(LinearRegressionModel.predict(x_test), y_test)

    print("The training RMSE is: " + str(train_RMSE))
    print("The testing RMSE is: " + str(test_RMSE))

    training_errors.append(train_RMSE)
    testing_errors.append(test_RMSE)

# plotting
plt.plot(training_sizes, training_errors)
plt.plot(training_sizes, testing_errors)
plt.xlabel('training size')
plt.ylabel('errors')
plt.title('training sizes vs training and testing RMSE')
plt.xticks(())
plt.yticks(())
plt.legend(["training RMSEs", "testing RMSEs"])
plt.show()

# comments
comment = "As we can see from the above plot, as we increase the training s
print(comment)
```

```
training size: 100
The training RMSE is: 2.7095712172184503
The testing RMSE is: 3.235891136463703
training size: 200
The training RMSE is: 2.8787337786435097
The testing RMSE is: 3.120584020836615
training size: 300
The training RMSE is: 2.902438563872487
The testing RMSE is: 3.0853029268638497
training size: 400
The training RMSE is: 2.8835567592723597
The testing RMSE is: 3.0867271255837707
training size: 500
The training RMSE is: 2.904901388491885
The testing RMSE is: 3.0901971874475205
training size: 614
```

```
The training RMSE is: 2.8864199100237773
The testing RMSE is: 3.100140402234829
```




```
As we can see from the above plot, as we increase the training size, the
training error increases while the testing error decreases. Thus as we in
crease the amount of data to train we generally get a more accurate mode
l!
```

Part 2 - Classification

CLASSIFICATION: LABELS ARE DISCRETE VALUES.

Here the model is trained to classify each instance into a set of predefined discrete classes. On inputting a feature vector into the model, the trained model is able to predict a class of that instance.

Q2.1

Bucket the values of 'y1' i.e 'Heating Load' from the original dataset into 3 classes:

0: 'Low' (< 14),
1: 'Medium' (14-28),
2: 'High' (>28)

HINT: Use pandas.cut

This converts the given dataset into a classification problem. Use this dataset with transformed 'heating load' to create a **logistic regression** classifiction model that predicts heating load type of a building. Split the data randomly into training and test set. Train the model on 80% of the data (80-20 split).

In [5]:
```
# your code
# get X and Y
X=data.iloc[:,:-1]
Y=data['Y1']
Y_bucketed=pd.cut(Y,[-100,14,28,100], right=False, labels=['Low', 'Medium',
print(Y_bucketed.head())

# split the train and test data
x_train, x_test, y_train, y_test = train_test_split(X, Y_bucketed, test_siz

LogisticRegressionModel = linear_model.LogisticRegression()
print ('Training a logistic Regression Model..')
LogisticRegressionModel.fit(x_train, y_train)
```

```
0      High
1    Medium
2    Medium
3      High
4    Medium
Name: Y1, dtype: category
Categories (3, object): [Low < Medium < High]
Training a logistic Regression Model..

/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py:4
33: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Spe
cify a solver to silence this warning.
  FutureWarning)
/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py:4
60: FutureWarning: Default multi_class will be changed to 'auto' in 0.22.
Specify the multi_class option to silence this warning.
  "this warning.", FutureWarning)
```

Out[5]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=Tr
ue,
          intercept_scaling=1, max_iter=100, multi_class='warn',
          n_jobs=None, penalty='l2', random_state=None, solver='warn',
          tol=0.0001, verbose=0, warm_start=False)
```

Q2.2

- Print the training and test accuracies
- Print the confusion matrix
- Print the precision and recall numbers for all the classes

In [6]:
```
# your code

# TRAINING ACCURACY

training_accuracy=LogisticRegressionModel.score(x_train,y_train)
print ('Training Accuracy:',training_accuracy)

# TESTING ACCURACY

testing_accuracy=LogisticRegressionModel.score(x_test,y_test)
print ('Testing Accuracy:',testing_accuracy, '\n')

# confusion matrix
from sklearn.metrics import confusion_matrix
y_true = y_test
y_pred = LogisticRegressionModel.predict(x_test)
ConfusionMatrix=pd.DataFrame(confusion_matrix(y_true, y_pred),columns=['Pre
print ('Confusion matrix of test data is: \n',ConfusionMatrix,'\n')

# precision
from sklearn.metrics import precision_score
print("Average precision for the 3 classes is - ", precision_score(y_true,

# recall
from sklearn.metrics import recall_score
print("Average recall for the 3 classes is - ", recall_score(y_true, y_pred
```

```
Training Accuracy: 0.8110749185667753
Testing Accuracy: 0.7402597402597403

Confusion matrix of test data is:
           Predicted 0  Predicted 1  Predicted 2
Actual 0           59            0            0
Actual 1            0           33            3
Actual 2           22           15           22

Average precision for the 3 classes is -  [0.72839506 0.6875     0.88
]
Average recall for the 3 classes is -  [1.         0.91666667 0.37288136]
```

Q2.3

K Fold Cross Validation

In k-fold cross-validation, the shuffled training data is partitioned into k disjoint sets and the model is trained on k –1 sets and validated on the kth set. This process is repeated k times with each set chosen as the validation set once. The cross-validation accuracy is reported as the average accuracy of the k iterations

Use 7-fold cross validation on the training data. Print the average accuracy

In [25]:
```
# your code
from sklearn.model_selection import KFold

accuracies = []

kf = KFold(n_splits=7)
i = 0
model_storage = []
for train_index, val_index in kf.split(x_train):
    # keep track of iteration
    i += 1
    x_train_small, x_val_small = x_train.iloc[train_index], x_train.iloc[va
    y_train_small, y_val_small = y_train.iloc[train_index], y_train.iloc[va
    LogisticRegressionModel = linear_model.LogisticRegression()
    print ('Iteration ' + str(i) + ': Training a logistic Regression Model.
    LogisticRegressionModel.fit(x_train_small, y_train_small)
    model_storage.append(LogisticRegressionModel)
    val_accuracy=LogisticRegressionModel.score(x_val_small,y_val_small)
    accuracies.append(val_accuracy)
    print("Validation accuracy at iteration " + str(i) + " is " + str(val_a

cv_accuracy = np.mean(accuracies)
print("The cross validation accuracy is " + str(cv_accuracy))
```

```
Iteration 1: Training a logistic Regression Model..
Validation accuracy at iteration 1 is 0.7954545454545454

Iteration 2: Training a logistic Regression Model..
Validation accuracy at iteration 2 is 0.7840909090909091

Iteration 3: Training a logistic Regression Model..
Validation accuracy at iteration 3 is 0.7727272727272727

Iteration 4: Training a logistic Regression Model..
Validation accuracy at iteration 4 is 0.75

Iteration 5: Training a logistic Regression Model..
Validation accuracy at iteration 5 is 0.8181818181818182

Iteration 6: Training a logistic Regression Model..
Validation accuracy at iteration 6 is 0.8275862068965517

Iteration 7: Training a logistic Regression Model..
Validation accuracy at iteration 7 is 0.8275862068965517

The cross validation accuracy is 0.7965181370353784

/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py:4
33: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Spe
cify a solver to silence this warning.
  FutureWarning)
/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py:4
60: FutureWarning: Default multi_class will be changed to 'auto' in 0.22.
Specify the multi_class option to silence this warning.
  "this warning.", FutureWarning)
```

```
/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py:4
33: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Spe
cify a solver to silence this warning.
  FutureWarning)
/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py:4
60: FutureWarning: Default multi_class will be changed to 'auto' in 0.22.
Specify the multi_class option to silence this warning.
  "this warning.", FutureWarning)
/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py:4
33: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Spe
cify a solver to silence this warning.
  FutureWarning)
/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py:4
60: FutureWarning: Default multi_class will be changed to 'auto' in 0.22.
Specify the multi_class option to silence this warning.
  "this warning.", FutureWarning)
/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py:4
33: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Spe
cify a solver to silence this warning.
  FutureWarning)
/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py:4
60: FutureWarning: Default multi_class will be changed to 'auto' in 0.22.
Specify the multi_class option to silence this warning.
  "this warning.", FutureWarning)
/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py:4
33: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Spe
cify a solver to silence this warning.
  FutureWarning)
/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py:4
60: FutureWarning: Default multi_class will be changed to 'auto' in 0.22.
Specify the multi_class option to silence this warning.
  "this warning.", FutureWarning)
/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py:4
33: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Spe
cify a solver to silence this warning.
  FutureWarning)
/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py:4
60: FutureWarning: Default multi_class will be changed to 'auto' in 0.22.
Specify the multi_class option to silence this warning.
  "this warning.", FutureWarning)
/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py:4
33: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Spe
cify a solver to silence this warning.
  FutureWarning)
/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py:4
60: FutureWarning: Default multi_class will be changed to 'auto' in 0.22.
Specify the multi_class option to silence this warning.
  "this warning.", FutureWarning)
```

Q2.4

One of the preprocessing steps in Data science is Feature Scaling i.e getting all our data on the same scale by setting same Min-Max of feature values. This makes training less sensitive to the scale of features . Scaling is important in algorithms that use distance functions as a part of classification. If we Scale features in the range [0,1] it is called unity based normalization.

Perform unity based normalization on the above dataset and train the model again, compare model performance in training and validation with your previous model.

refer:http://scikit-learn.org/stable/modules/preprocessing.html#preprocessing-scaler (http://scikit-learn.org/stable/modules/preprocessing.html#preprocessing-scaler)
more at: https://en.wikipedia.org/wiki/Feature_scaling (https://en.wikipedia.org/wiki/Feature_scaling)

In [28]:
```
# your code
from sklearn import preprocessing
min_max_scaler = preprocessing.MinMaxScaler()
x_train_minmax = min_max_scaler.fit_transform(x_train)

print("min max scaled training data\n")
print(x_train_minmax)
print()

accuracies = []

kf = KFold(n_splits=7)
i = 0
for train_index, val_index in kf.split(x_train):
    # keep track of iteration
    i += 1
    x_train_small, x_val_small = x_train_minmax[train_index], x_train_minma
    y_train_small, y_val_small = y_train.iloc[train_index], y_train.iloc[va
    LogisticRegressionModel = linear_model.LogisticRegression()
    print ('Iteration ' + str(i) + ': Training a logistic Regression Model.
    LogisticRegressionModel.fit(x_train_small, y_train_small)
    val_accuracy=LogisticRegressionModel.score(x_val_small,y_val_small)
    accuracies.append(val_accuracy)
    print("Validation accuracy at iteration " + str(i) + " is " + str(val_a

cv_accuracy = np.mean(accuracies)
print("The cross validation accuracy is " + str(cv_accuracy) + '\n')

comment = "As we can see from the comparasion of the model accuracy with ar
print(comment)
```

```
min max scaled training data

[[0.19444444 0.75       0.28571429 ... 0.33333333 1.         0.2        ]
 [0.11111111 0.83333333 0.42857143 ... 0.33333333 0.625      0.4        ]
 [0.38888889 0.5        1.         ... 0.         0.625      1.         ]
 ...
 [0.         1.         0.71428571 ... 0.33333333 1.         0.6        ]
 [0.25       0.66666667 0.14285714 ... 0.33333333 0.625      0.4        ]
 [0.55555556 0.33333333 0.42857143 ... 0.66666667 0.25       0.4        ]]

Iteration 1: Training a logistic Regression Model..
Validation accuracy at iteration 1 is 0.7954545454545454

Iteration 2: Training a logistic Regression Model..
Validation accuracy at iteration 2 is 0.8181818181818182

Iteration 3: Training a logistic Regression Model..
Validation accuracy at iteration 3 is 0.8181818181818182

Iteration 4: Training a logistic Regression Model..
Validation accuracy at iteration 4 is 0.7386363636363636

Iteration 5: Training a logistic Regression Model..
Validation accuracy at iteration 5 is 0.8295454545454546

Iteration 6: Training a logistic Regression Model..
```

```
Validation accuracy at iteration 6 is 0.8850574712643678

Iteration 7: Training a logistic Regression Model..
Validation accuracy at iteration 7 is 0.8505747126436781

The cross validation accuracy is 0.819376026272578

As we can see from the comparasion of the model accuracy with and without
min max preprocessing that the model with the min max preprocessing is ab
le to achieve 82% validation accuracy while the one without can only achi
eve 79% accuracy. Thus shows the min-max scaling preprocessing is able to
help us with training our classification model.

/anaconda3/lib/python3.7/site-packages/sklearn/preprocessing/data.py:323:
DataConversionWarning: Data with input dtype int64, float64 were all conv
erted to float64 by MinMaxScaler.
  return self.partial_fit(X, y)
/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py:4
33: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Spe
cify a solver to silence this warning.
  FutureWarning)
/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py:4
60: FutureWarning: Default multi_class will be changed to 'auto' in 0.22.
Specify the multi_class option to silence this warning.
  "this warning.", FutureWarning)
/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py:4
33: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Spe
cify a solver to silence this warning.
  FutureWarning)
/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py:4
60: FutureWarning: Default multi_class will be changed to 'auto' in 0.22.
Specify the multi_class option to silence this warning.
  "this warning.", FutureWarning)
/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py:4
33: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Spe
cify a solver to silence this warning.
  FutureWarning)
/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py:4
60: FutureWarning: Default multi_class will be changed to 'auto' in 0.22.
Specify the multi_class option to silence this warning.
  "this warning.", FutureWarning)
/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py:4
33: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Spe
cify a solver to silence this warning.
  FutureWarning)
/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py:4
60: FutureWarning: Default multi_class will be changed to 'auto' in 0.22.
Specify the multi_class option to silence this warning.
  "this warning.", FutureWarning)
/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py:4
33: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Spe
cify a solver to silence this warning.
  FutureWarning)
/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py:4
60: FutureWarning: Default multi_class will be changed to 'auto' in 0.22.
Specify the multi_class option to silence this warning.
  "this warning.", FutureWarning)
```

```
/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py:4
33: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Spe
cify a solver to silence this warning.
  FutureWarning)
/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py:4
60: FutureWarning: Default multi_class will be changed to 'auto' in 0.22.
Specify the multi_class option to silence this warning.
  "this warning.", FutureWarning)
/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py:4
33: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Spe
cify a solver to silence this warning.
  FutureWarning)
/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py:4
60: FutureWarning: Default multi_class will be changed to 'auto' in 0.22.
Specify the multi_class option to silence this warning.
  "this warning.", FutureWarning)
```

In [ ]: