Lab 6 (pt. 2): Multivariate Linear Regression

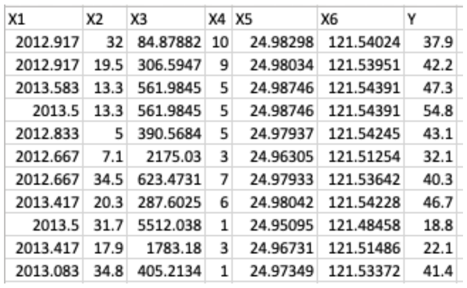
This is an INDIVIDUAL assignment. Due date is as indicated on BeachBoard. Follow ALL instructions otherwise you may lose points. In this lab, you will be making predictions based on more elaborate data. You will also analyze the accuracy of your model. You will need to use numpy and pandas for this lab. Please note that since the labs will be graded separately, there will be separate resubmissions for lab 6 (pt1) and lab 6 (pt2).

Review:
In the previous lab, we discussed how you can find the slope and y-intercept of univariate data by utilizing the formula below…

$$\begin{bmatrix} m \\ b \end{bmatrix} = (X^T X)^{-1} X^T y$$

However, what would you do if the data that was presented to you contains more columns of data?

Multivariate Linear Regression:
The presented data was in a form such that there was only one variable (x). However, it is very realistic for data to rely on multiple independent variables. For this example, we will use the famous real estate data set. I have attached an edited version on BeachBoard for your convenience. Please use the data that I provided for you on BeachBoard. Otherwise, your answers and your formatting will not match. A portion of the csv file (`real estate train.csv`) is shown below



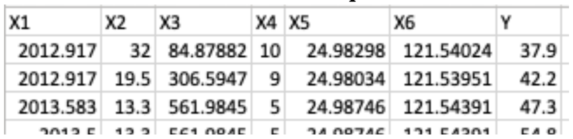
X1	X2	X3	X4	X5	X6	Y
2012.917	32	84.87882	10	24.98298	121.54024	37.9
2012.917	19.5	306.5947	9	24.98034	121.53951	42.2
2013.583	13.3	561.9845	5	24.98746	121.54391	47.3
2013.5	13.3	561.9845	5	24.98746	121.54391	54.8
2012.833	5	390.5684	5	24.97937	121.54245	43.1
2012.667	7.1	2175.03	3	24.96305	121.51254	32.1
2012.667	34.5	623.4731	7	24.97933	121.53642	40.3
2013.417	20.3	287.6025	6	24.98042	121.54228	46.7
2013.5	31.7	5512.038	1	24.95095	121.48458	18.8
2013.417	17.9	1783.18	3	24.96731	121.51486	22.1
2013.083	34.8	405.2134	1	24.97349	121.53372	41.4

X1: Transaction date
X2: House age
X3: Distance to the nearest MRT station
X4: Number of convenience stores
X5: Latitude
X6: Longitude
Y: house price of unit area

The goal is to find a multivariate linear equation that can predict the house price of the unit area using all of the mentioned features.

$$Y = m1x1 + m2x2 + m3x3 + m4x4 + m5x5 + m6x6 + b$$

We simply apply the linear algebra method above, but in a larger scale. Based on the snippet from the table, we can create our matrix equations from this.



X1	X2	X3	X4	X5	X6	Y
2012.917	32	84.87882	10	24.98298	121.54024	37.9
2012.917	19.5	306.5947	9	24.98034	121.53951	42.2
2013.583	13.3	561.9845	5	24.98746	121.54391	47.3

$$\begin{bmatrix}37.9\\42.2\\47.3\\\vdots\end{bmatrix} = \begin{bmatrix}2012.917 & 32 & 84.87882 & 10 & 24.98298 & 121.54024 & 1\\2012.9178 & 19.5 & 306.5947 & 9 & 24.98034 & 121.53951 & 1\\2013.583 & 13.3 & 56109845 & 5 & 24.98746 & 121.54391 & 1\\\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots\end{bmatrix}\begin{bmatrix}m1\\m2\\m3\\m4\\m5\\m6\\b\end{bmatrix}$$

Looks familiar? This is still the same thing as our $y = X\begin{bmatrix}m\\b\end{bmatrix}$ setup from before. However, we have multiple m values instead of just one. So instead of $y = X\begin{bmatrix}m\\b\end{bmatrix}$, our matrix equation looks more like…

$$y = X\begin{bmatrix}m1\\m2\\m3\\m4\\m5\\m6\\b\end{bmatrix}$$

Thus, the approach is still the same.

$$\begin{bmatrix}m1\\m2\\m3\\m4\\m5\\m6\\b\end{bmatrix} = (X^TX)^{-1}X^Ty$$

$$where\ X = \begin{bmatrix}2012.917 & 32 & 84.87882 & 10 & 24.98298 & 121.54024 & 1\\2012.9178 & 19.5 & 306.5947 & 9 & 24.98034 & 121.53951 & 1\\2013.583 & 13.3 & 56109845 & 5 & 24.98746 & 121.54391 & 1\\\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots\end{bmatrix}, \mathbf{y} = \begin{bmatrix}37.9\\42.2\\47.3\\\vdots\end{bmatrix}$$

The result is $\begin{bmatrix}m1\\m2\\m3\\m4\\m5\\m6\\b\end{bmatrix} = \begin{bmatrix}4.95347\\-0.2696252\\-0.0044963\\1.11479937\\230.797555\\-13.593203\\-14039.67836\end{bmatrix} \approx \begin{bmatrix}4.9535\\-0.2696\\-0.0045\\1.1148\\230.7976\\-13.5932\\-14039.6784\end{bmatrix}$ *when rounded to 4 decimal places*

Therefore, if I wanted to predict what the house price for a unit with the following data (from first data row in `real estate test.csv`):
X1: Transaction date is **2013.25**
X2: House age is **26.8**
X3: Distance to the nearest MRT station is **482.7581**
X4: Number of convenience store is **5**
X5: Latitude is **24.97433**
X6: Longitude is **121.53863**
We can easily predict the house price for the unit (y) by plugging in what we know into the formula

$$y = X \begin{bmatrix} m1 \\ m2 \\ m3 \\ m4 \\ m5 \\ m6 \\ b \end{bmatrix}$$

$$y = [2013.25 \quad 26.8 \quad 482.7581 \quad 5 \quad 24.97433 \quad 121.53863 \quad 1] \times \begin{bmatrix} 4.9535 \\ -0.2696 \\ -0.0045 \\ 1.1148 \\ 230.7976 \\ -13.5932 \\ -14039.6784 \end{bmatrix} = 401.0483 \ldots$$

$$\approx 41.0483 \textit{ when rounded to four decimal places}$$

If you look at the actual price, from the csv file (`real estate test.csv`), you'll find that the actual value is 35.5. That means that we have an **absolute error** by using the following formula:

$$abs\ err = |predicted - actual|$$
$$= |41.0483 - 35.5| = 5.5483$$

We can also calculate the **relative error** by using the following formula:

$$rel\ err = \frac{|predicted - actual|}{actual}$$
$$= \frac{|41.0483 - 35.5|}{35.5} = 0.1563 \textit{ when rounded to 4 decimal places}$$

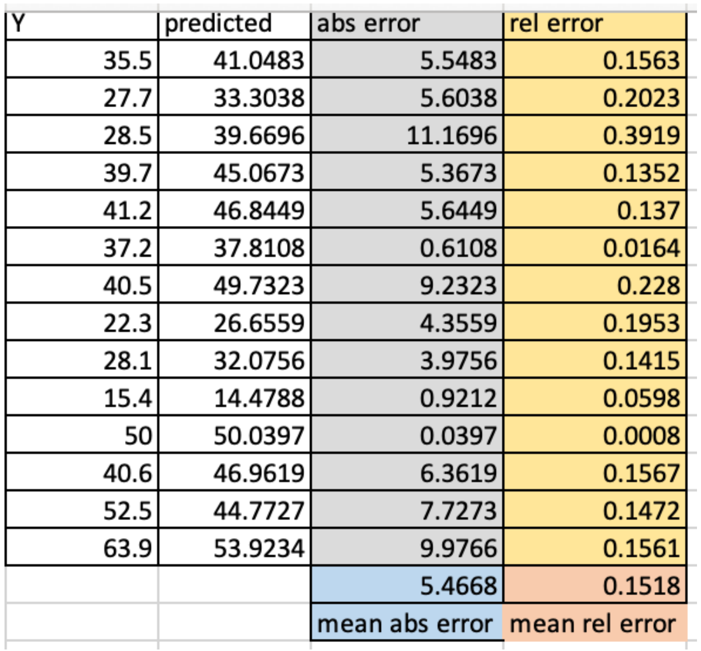
This means that the predicted answer is off by 15.63%

We can better test the accuracy of the linear regression model by finding the **mean absolute error (MAE)** and the **mean relative error (MRE).**

$$MAE = \frac{1}{n}\sum_{i}^{n} |predicted_i - actual_i|$$

$$MRE = \frac{1}{n}\sum_{i}^{n} \frac{|predicted_i - actual_i|}{actual_i}$$

Where *n* is the total number of cases and *i* is the iteration/case number.
You can see a summary of the test results on the next page.



Y	predicted	abs error	rel error
35.5	41.0483	5.5483	0.1563
27.7	33.3038	5.6038	0.2023
28.5	39.6696	11.1696	0.3919
39.7	45.0673	5.3673	0.1352
41.2	46.8449	5.6449	0.137
37.2	37.8108	0.6108	0.0164
40.5	49.7323	9.2323	0.228
22.3	26.6559	4.3559	0.1953
28.1	32.0756	3.9756	0.1415
15.4	14.4788	0.9212	0.0598
50	50.0397	0.0397	0.0008
40.6	46.9619	6.3619	0.1567
52.5	44.7727	7.7273	0.1472
63.9	53.9234	9.9766	0.1561
		5.4668	0.1518
		mean abs error	mean rel error

After calculating all of the absolute errors and the relative errors, we can find the MAE and the MRE. Overall, based on the test data, our model has a 15.18% error on average.

Task:
The purpose of the second part of the lab is to create a framework that will take a csv file with any number of columns and will create a linear regression model. You will also analyze the accuracy of this linear regression model.

1. Take a close look at the `multi_lin_reg.py` file. There are four empty functions: `multivar_linreg(file_name)` and `predict(inputs, file_name) and` `MAE(inputs, file_name)` and `MRE(inputs, file_name)`. Read through all of their descriptions carefully. Remember, you will lose points if you do not follow the instructions. We are using a grading script

Summary of function tasks

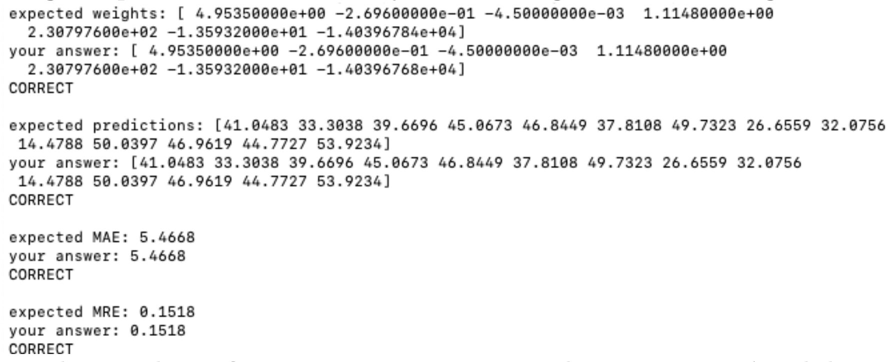
`Multivar_linreg(file_name):` Given the csv `file_name`, find all of the weights and return these values in a numpy array. This `1xn` numpy array should contain `[m1, m2, m3, … , b]` in this order. Round all values to four decimal places.
`predict(inputs, file_name):` Given `inputs`, which is a numpy array of all of the weights `[m1, m2, m3, … , b]`, make predictions from the data given in `file_name`. The predictions will be stored in a `1xm` numpy array `[y1, y2, y3, …]`. Each row of data from the csv should have a prediction. Round all values to four decimal places
`MAE(inputs, file_name):` Find the mean absolute error of the linear regression model given by `inputs`, which is a numpy array of all of the weights `[m1, m2, m3, … , b]`. The mean absolute error will be calculated by testing the linear regression model with the data from `file_name`. Round all values to four decimal places.

`MRE(inputs, file_name):`
Find the mean relative error of the linear regression model given by `inputs`, which is a numpy array of all of the weights `[m1, m2, m3, … , b]`. The mean relative error will be calculated by testing the linear regression model with the data from `file_name`. Round all values to four decimal places.

Some important notes:

- Though this example uses six columns (six independent variables), other test cases may use more or less columns. However, there will be at least one independent variables.
- For consistency's sake, do not round until the very end. Meaning you should not round anything until you return your answers.
- If you want to create extra functions/methods to assist you, feel free to do so. However, we will only be testing the three functions that are originally in the file.
- **If you use any library's linear regression or least squares method function, you will get an automatic zero. You must implement this on your own!**

2. Your job is to implement all four of these functions so that it passes all test cases. We provide two csv files. `Real estate train.csv` is for `multivar_linreg()` and `Real estate test.csv` is for the other functions. However, we will be using other data sets and csv files to check if your work is correct.
3. By running the provided test cases, you should get the following results:



```
expected weights: [ 4.95350000e+00 -2.69600000e-01 -4.50000000e-03  1.11480000e+00
  2.30797600e+02 -1.35932000e+01 -1.40396784e+04]
your answer: [ 4.95350000e+00 -2.69600000e-01 -4.50000000e-03  1.11480000e+00
  2.30797600e+02 -1.35932000e+01 -1.40396768e+04]
CORRECT

expected predictions: [41.0483 33.3038 39.6696 45.0673 46.8449 37.8108 49.7323 26.6559 32.0756
 14.4788 50.0397 46.9619 44.7727 53.9234]
your answer: [41.0483 33.3038 39.6696 45.0673 46.8449 37.8108 49.7323 26.6559 32.0756
 14.4788 50.0397 46.9619 44.7727 53.9234]
CORRECT

expected MAE: 5.4668
your answer: 5.4668
CORRECT

expected MRE: 0.1518
your answer: 0.1518
CORRECT
```

4. After completing these functions, comment out the test cases (or delete them) or else the grading script will pick it up and mark your program as incorrect. Ensure that you have commented out or deleted ALL print statements. You risk losing points if your file prints anything.
5. Convert your `multi_lin_reg.py` file to a `.txt` file. Submit your `multi_lin_reg.py` file and your `.txt` file on BeachBoard. Do NOT submit it in compressed folder.

Some helpful functions (refer to part 1 for other helpful functions)

Function name	**What it does**
`np.round(array, num)`	Rounds all elements in `array` to `num` decimal places Example: `np.round([0.1234, 0.6545], 3) => [0.123, 0.655]`

`df_name.shape`	Gets the dimension of the data frame `df.shape => (num_rows, num_columns)`
`np.append(x, y)`	Appends y to the end of x. See documentation here

Grading rubric:

To achieve any points, your submission must have the following. Anything missing from this list will result in an automatic zero. NO EXCEPTIONS!

- Submit everything: py file, txt file, and pdf file
- Program has no errors (infinite loops, syntax errors, logical errors, etc.) that terminates the program

Please note that if you change the function headers or if you do not return the proper outputs according to the function requirements, you risk losing all points for those test cases.

Points	Requirement
5	Submission is correct. All two files are part of submission (py file and txt file)- All or nothing
15	Implemented `multivar_linreg()` correctly (three other cases not including Real estate)
12	Implemented `predict()` correctly (three other cases not including Real estate)
6	Implemented `MAE()` correctly (three other cases not including Real estate)
6	Implemented `MRE()` correctly ((three other cases not including Real estate)
5	Passes original test cases (test cases on python file have been commented out too)- all or nothing
TOTAL: 49	