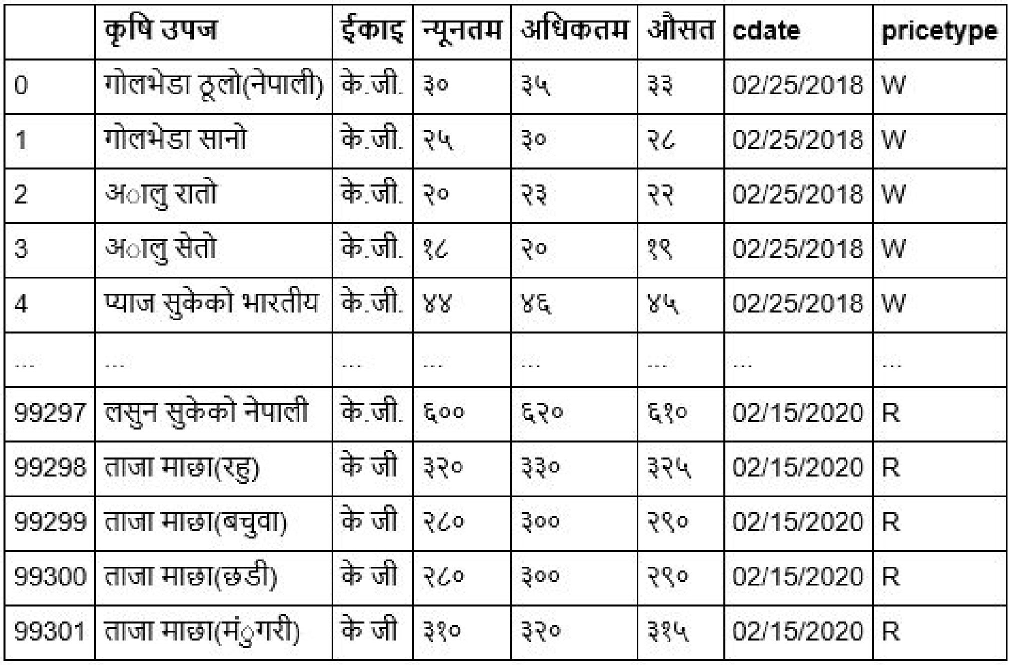
Linear Regression
Report on
Price Dataset

1.Data Analysis and Visualization

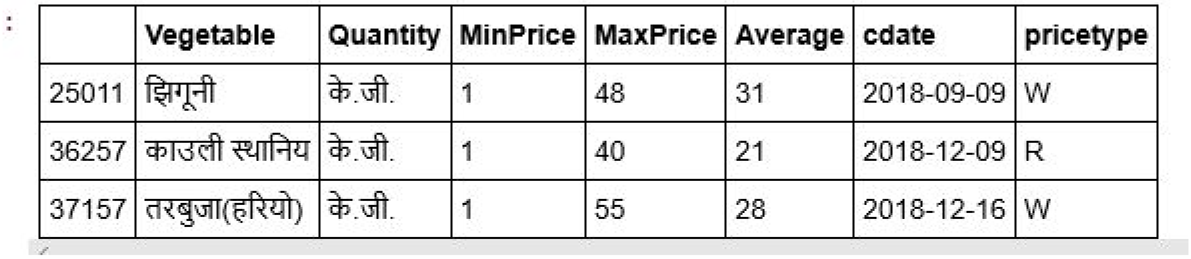
Given dataset looked like this:



	कृषि उपज	ईकाइ	न्यूनतम	अधिकतम	औसत	cdate	pricetype
0	गोलभेडा ठूलो(नेपाली)	के.जी.	३०	३५	३३	02/25/2018	W
1	गोलभेडा सानो	के.जी.	२५	३०	२८	02/25/2018	W
2	अ◌ालु रातो	के.जी.	२०	२३	२२	02/25/2018	W
3	अ◌ालु सेतो	के.जी.	१८	२०	१९	02/25/2018	W
4	प्याज सुकेको भारतीय	के.जी.	४४	४६	४५	02/25/2018	W
...	...	...	...	...	...	...	...
99297	लसुन सुकेको नेपाली	के.जी.	६००	६२०	६१०	02/15/2020	R
99298	ताजा माछा(रहु)	के जी	३२०	३३०	३२५	02/15/2020	R
99299	ताजा माछा(बचुवा)	के जी	२८०	३००	२९०	02/15/2020	R
99300	ताजा माछा(छडी)	के जी	२८०	३००	२९०	02/15/2020	R
99301	ताजा माछा(मं◌ुगरी)	के जी	३१०	३२०	३१५	02/15/2020	R

There were 120 unique vegetables in the given dataset. The minimum price was Rs.1 and the maximum price was Rs.1650.

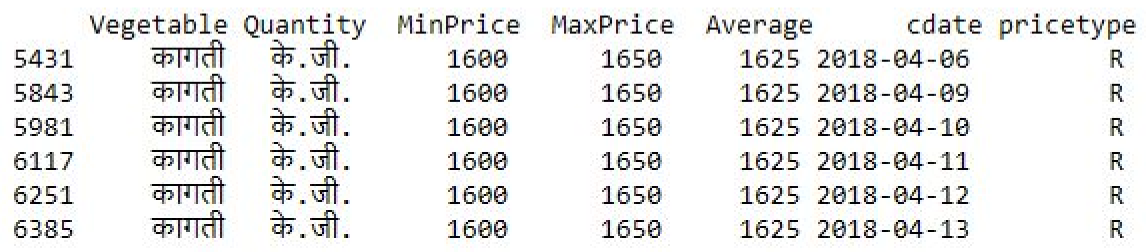
झिगूनी,काउली स्थानिय and तरबुजा(हरियो) had a minimum price than other vegetables at different dates.



:

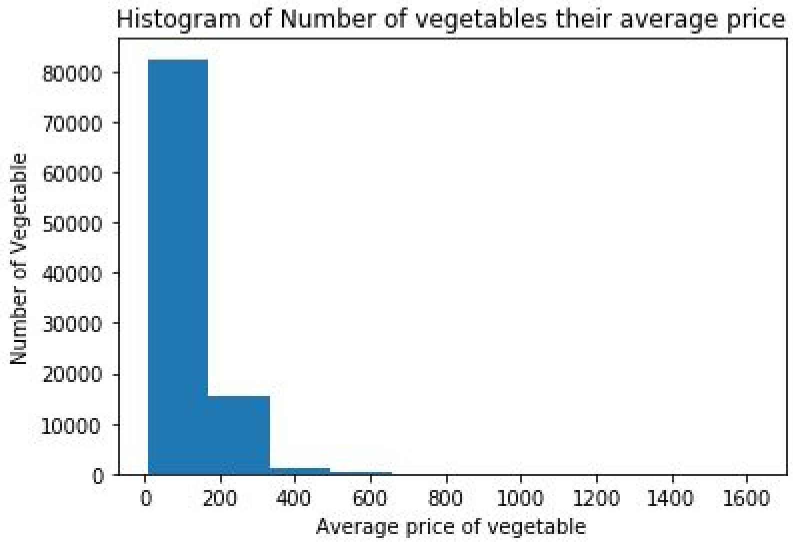
	Vegetable	Quantity	MinPrice	MaxPrice	Average	cdate	pricetype
25011	झिगूनी	के.जी.	1	48	31	2018-09-09	W
36257	काउली स्थानिय	के.जी.	1	40	21	2018-12-09	R
37157	तरबुजा(हरियो)	के.जी.	1	55	28	2018-12-16	W

कागती had a maximum price among other vegetables and maximum price in April.

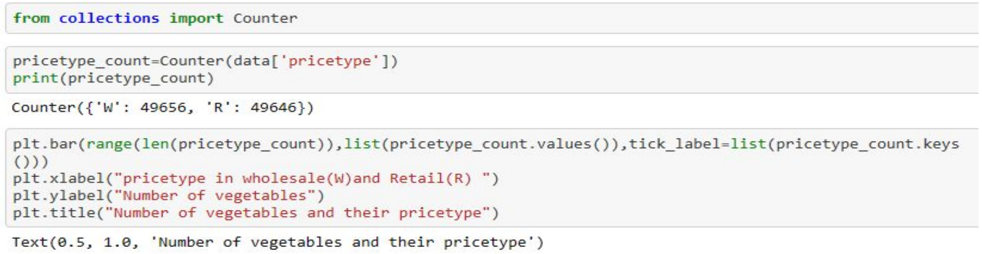


```
      Vegetable Quantity  MinPrice  MaxPrice  Average      cdate pricetype
5431      कागती    के.जी.      1600      1650     1625 2018-04-06         R
5843      कागती    के.जी.      1600      1650     1625 2018-04-09         R
5981      कागती    के.जी.      1600      1650     1625 2018-04-10         R
6117      कागती    के.जी.      1600      1650     1625 2018-04-11         R
6251      कागती    के.जी.      1600      1650     1625 2018-04-12         R
6385      कागती    के.जी.      1600      1650     1625 2018-04-13         R
```

In this dataset, maximum vegetables had an average price between 1 to 150.



In the dataset, the number of wholesale and retail was not the same. The difference between wholesale and retail price type was 10.



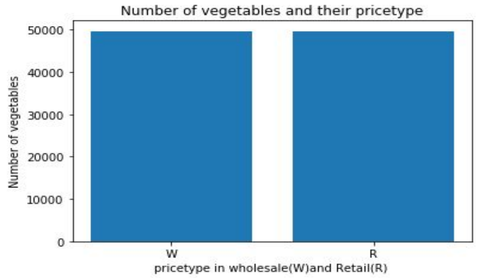
```
from collections import Counter
```

```
pricetype_count=Counter(data['pricetype'])
print(pricetype_count)
```

```
Counter({'W': 49656, 'R': 49646})
```

```
plt.bar(range(len(pricetype_count)),list(pricetype_count.values()),tick_label=list(pricetype_count.keys
()))
plt.xlabel("pricetype in wholesale(W)and Retail(R) ")
plt.ylabel("Number of vegetables")
plt.title("Number of vegetables and their pricetype")
```

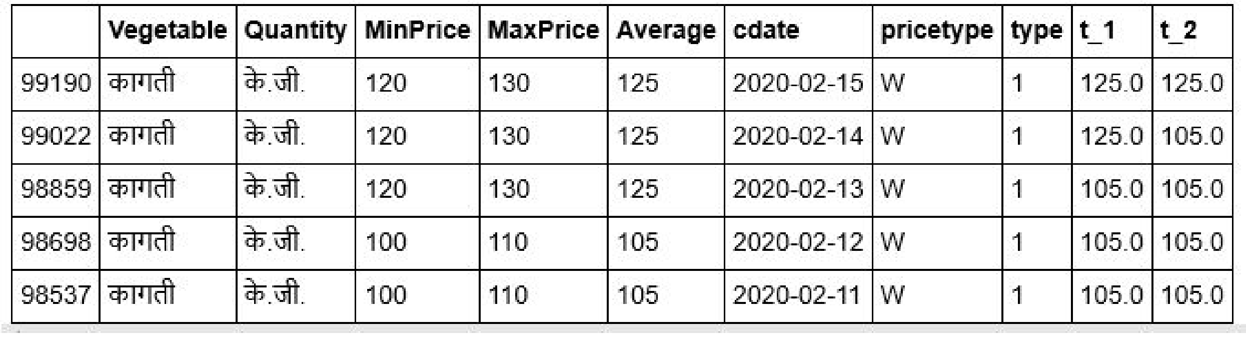
```
Text(0.5, 1.0, 'Number of vegetables and their pricetype')
```



2.Feature Extraction and Normalization

To predict tomorrow's vegetable price taken two features as the price of yesterday and the day before yesterday.

Final Dataset looked like this:

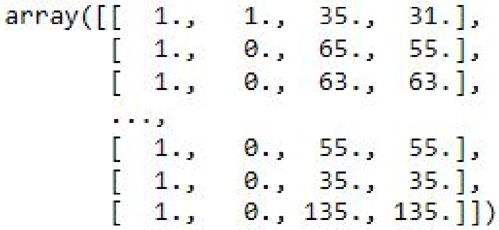


	Vegetable	Quantity	MinPrice	MaxPrice	Average	cdate	pricetype	type	t_1	t_2
99190	कागती	के.जी.	120	130	125	2020-02-15	W	1	125.0	125.0
99022	कागती	के.जी.	120	130	125	2020-02-14	W	1	125.0	105.0
98859	कागती	के.जी.	120	130	125	2020-02-13	W	1	105.0	105.0
98698	कागती	के.जी.	100	110	105	2020-02-12	W	1	105.0	105.0
98537	कागती	के.जी.	100	110	105	2020-02-11	W	1	105.0	105.0

The train dataset was split into x_train and y_train and the test dataset was split into x_test and y_test and validation set was split into x_val and y_val.

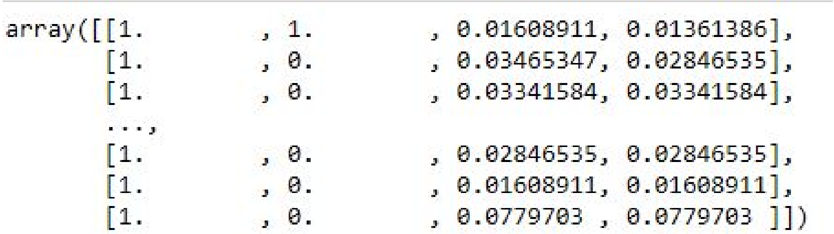
Min_Max Normalization was used for normalizing data.

Data before normalization



```
array([[  1.,   1.,  35.,  31.],
       [  1.,   0.,  65.,  55.],
       [  1.,   0.,  63.,  63.],
       ...,
       [  1.,   0.,  55.,  55.],
       [  1.,   0.,  35.,  35.],
       [  1.,   0., 135., 135.]])
```

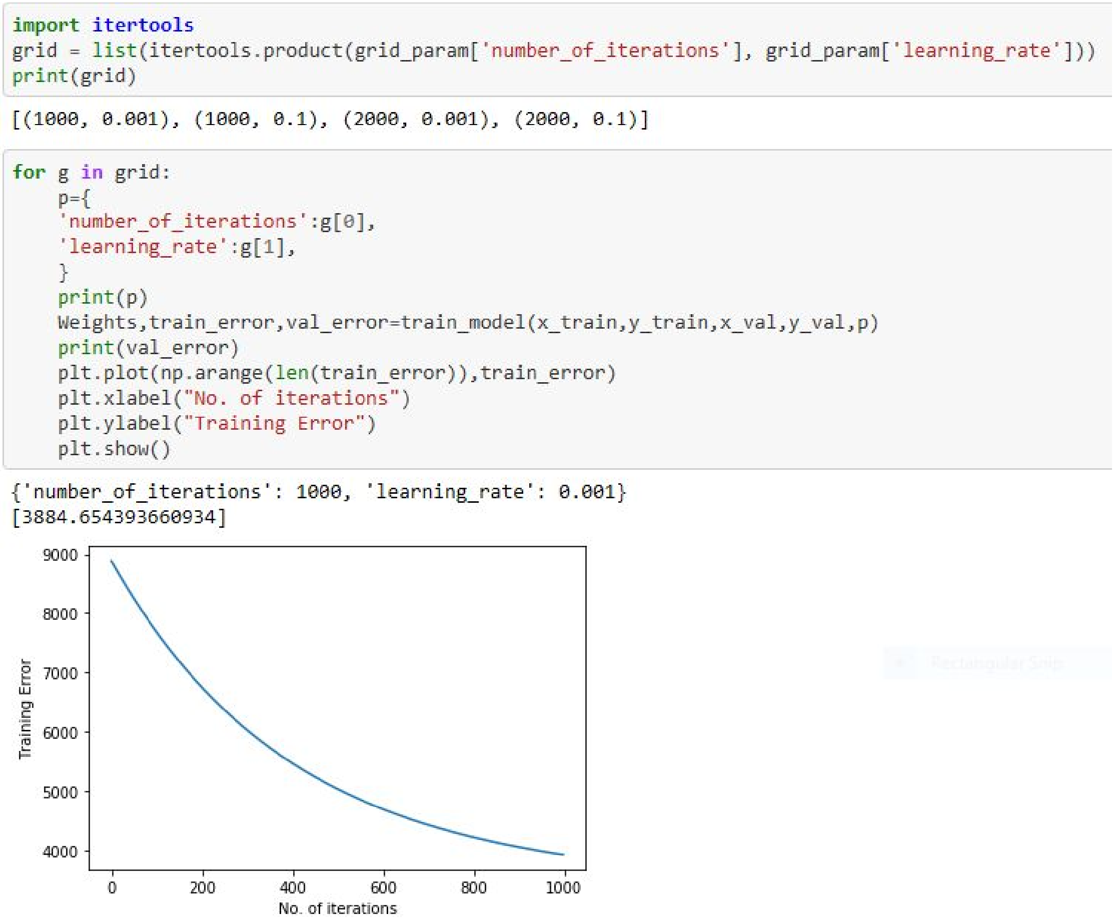
Data after applied Min_Max normalization



```
array([[1.        , 1.        , 0.01608911, 0.01361386],
       [1.        , 0.        , 0.03465347, 0.02846535],
       [1.        , 0.        , 0.03341584, 0.03341584],
       ...,
       [1.        , 0.        , 0.02846535, 0.02846535],
       [1.        , 0.        , 0.01608911, 0.01608911],
       [1.        , 0.        , 0.0779703 , 0.0779703 ]])
```

3.Grid search parameter with val loss

Mean Square Error was used as a model and **number_of_iterations** and **Learning_rate** used for grid search parameters.



```
import itertools
grid = list(itertools.product(grid_param['number_of_iterations'], grid_param['learning_rate']))
print(grid)
```

```
[(1000, 0.001), (1000, 0.1), (2000, 0.001), (2000, 0.1)]
```

```
for g in grid:
    p={
    'number_of_iterations':g[0],
    'learning_rate':g[1],
    }
    print(p)
    Weights,train_error,val_error=train_model(x_train,y_train,x_val,y_val,p)
    print(val_error)
    plt.plot(np.arange(len(train_error)),train_error)
    plt.xlabel("No. of iterations")
    plt.ylabel("Training Error")
    plt.show()
```

```
{'number_of_iterations': 1000, 'learning_rate': 0.001}
[3884.654393660934]
```

4.Train loss plot in best model

Best Hyperparameter was found in a number _of_ iteration **2000** and learning _rate **0.1** with training loss **539.86**

5.Model Evaluation using R2 Score

R-squared is a statistical measure of how close the data are to the fitted regression line. It is also known as the coefficient of determination, or the coefficient of multiple determination for multiple regression.

R-squared = Explained variation / Total variation

By evaluating the model using R2 Score got R2 Score **0.832** it means that model fits data very well.

6.Code Link

https://github.com/rachanakafle/Linear-and-Logistic-Regression/blob/master/LinearRegression.ipynb