Assignment #2 Report

Created	@June 10, 2022 1:17 AM

Question #1

Colab Notebook Link

https://colab.research.google.com/drive/1mEDigKUJ7ZxdexT43aPZI9khfzDr1G4X?usp=sharing

Report

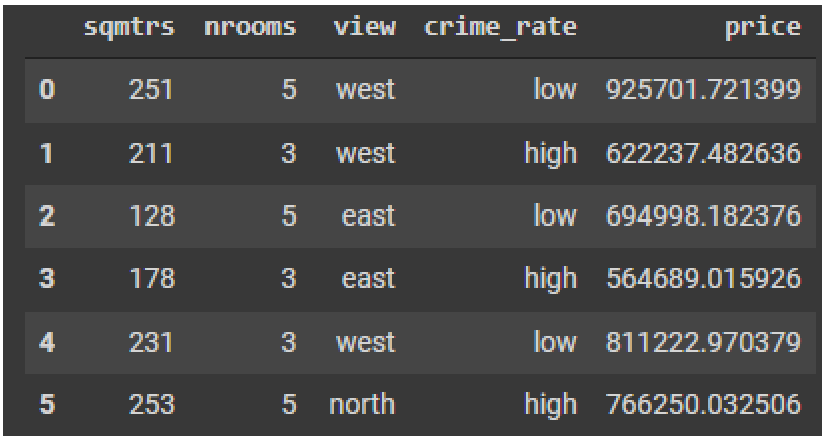
Problem Definition:

The aim of this project is to predict house prices based on the features of the houses. The features of the house includes the size of the house in square meters, number of rooms in the house, view of the house, and crime rate in the neighborhood which the house located. These independent variables will be used to predict the target variable, which is the house prices.

Preprocessing

Datasets - Train/ Validation/ Test

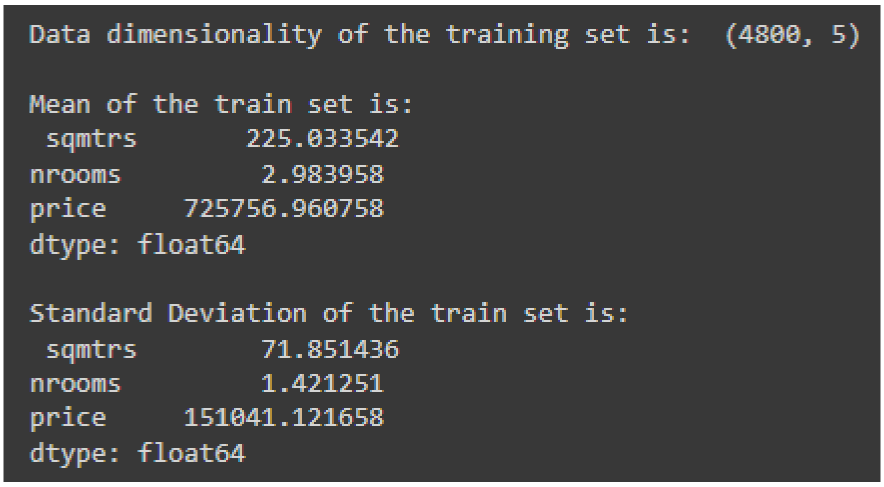
The column labels used in this projects dataset are ‘sqmtrs’, ‘nrooms’, ‘view’, ‘crime_rate’, and ‘price’. The size of the house represented as square meters under the ‘sqmtrs’ label, number of rooms in the house represented under the ‘nrooms’ label, view of the house represented under the ‘view’ label, crime_rate of the neighborhood represented under the ‘crime_rate’ label and price of the house represented under ‘price’ label. While ‘sqmtrs‘, nrooms’, and ‘price’ are numerical values, ‘view’ and ‘crime_rate’ are represented as categorical values (will be turned into numerical values in the preprocessing step).



	sqmtrs	nrooms	view	crime_rate	price
0	251	5	west	low	925701.721399
1	211	3	west	high	622237.482636
2	128	5	east	low	694998.182376
3	178	3	east	high	564689.015926
4	231	3	west	low	811222.970379
5	253	5	north	high	766250.032506

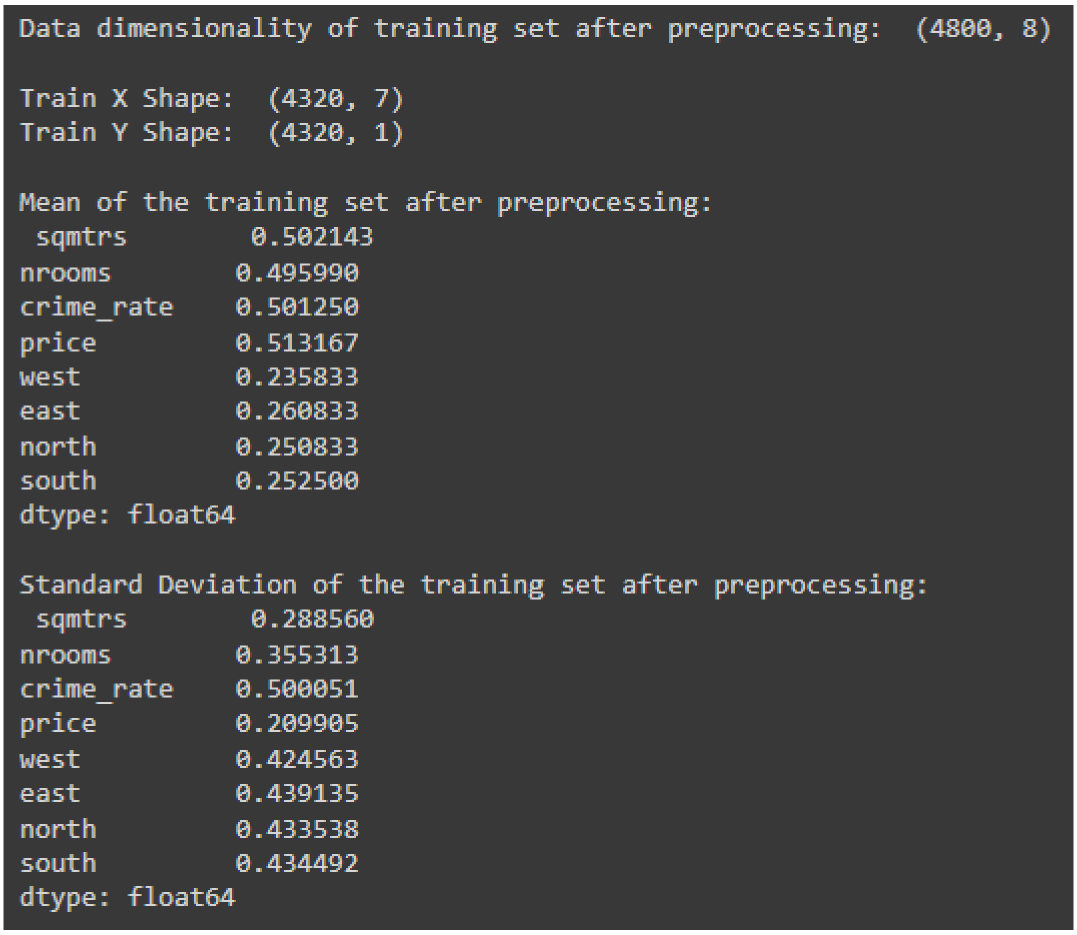
- **Train Set**

  Initially training set consists of 4800 rows and 5 columns. The mean and the standard deviation of the numerical values in the training set is:



```
Data dimensionality of the training set is:  (4800, 5)

Mean of the train set is:
 sqmtrs       225.033542
nrooms          2.983958
price      725756.960758
dtype: float64

Standard Deviation of the train set is:
 sqmtrs        71.851436
nrooms          1.421251
price      151041.121658
dtype: float64
```

NOTE: After preprocessing shape and mean/standard deviation of the training set becomes like the below due to additional columns and the MinMaxScaler operation. More detail about the preprocessing will be provided in the "Preprocessing" section of the report:

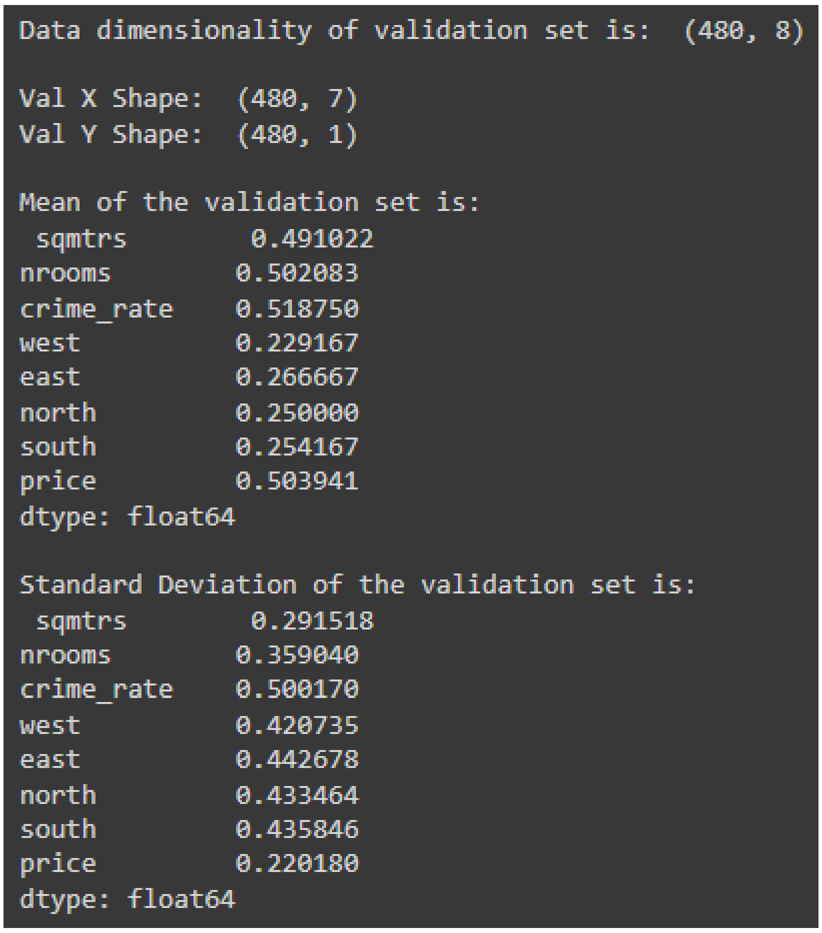


```
Data dimensionality of training set after preprocessing:  (4800, 8)

Train X Shape:  (4320, 7)
Train Y Shape:  (4320, 1)

Mean of the training set after preprocessing:
 sqmtrs        0.502143
nrooms        0.495990
crime_rate    0.501250
price         0.513167
west          0.235833
east          0.260833
north         0.250833
south         0.252500
dtype: float64

Standard Deviation of the training set after preprocessing:
 sqmtrs        0.288560
nrooms        0.355313
crime_rate    0.500051
price         0.209905
west          0.424563
east          0.439135
north         0.433538
south         0.434492
dtype: float64
```

- **Validation Set**

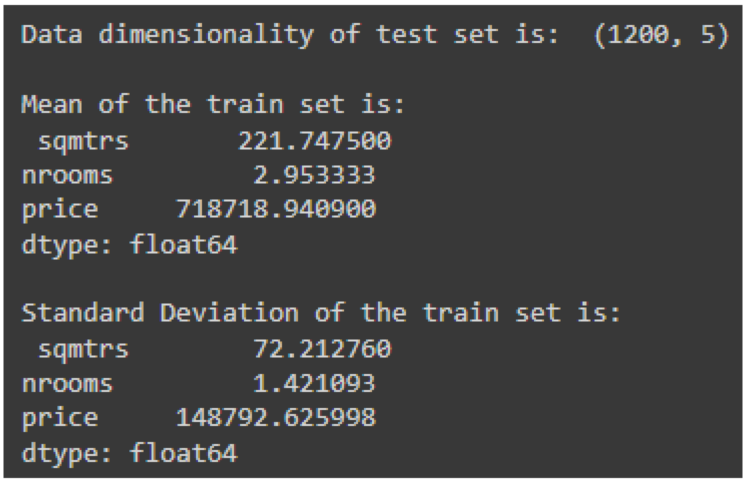
  Validation set is created out of training set using train_test_split with test_size = 0.1 and random_state = 42. After this operation size of the training set reduced to 4320 rows and size of the validation set becomes 480 rows.

  NOTE: The validation set created after preprocessing step; therefore, due to the additional columns and MinMaxScaler operation, its shape and the mean/standard deviation values are different.

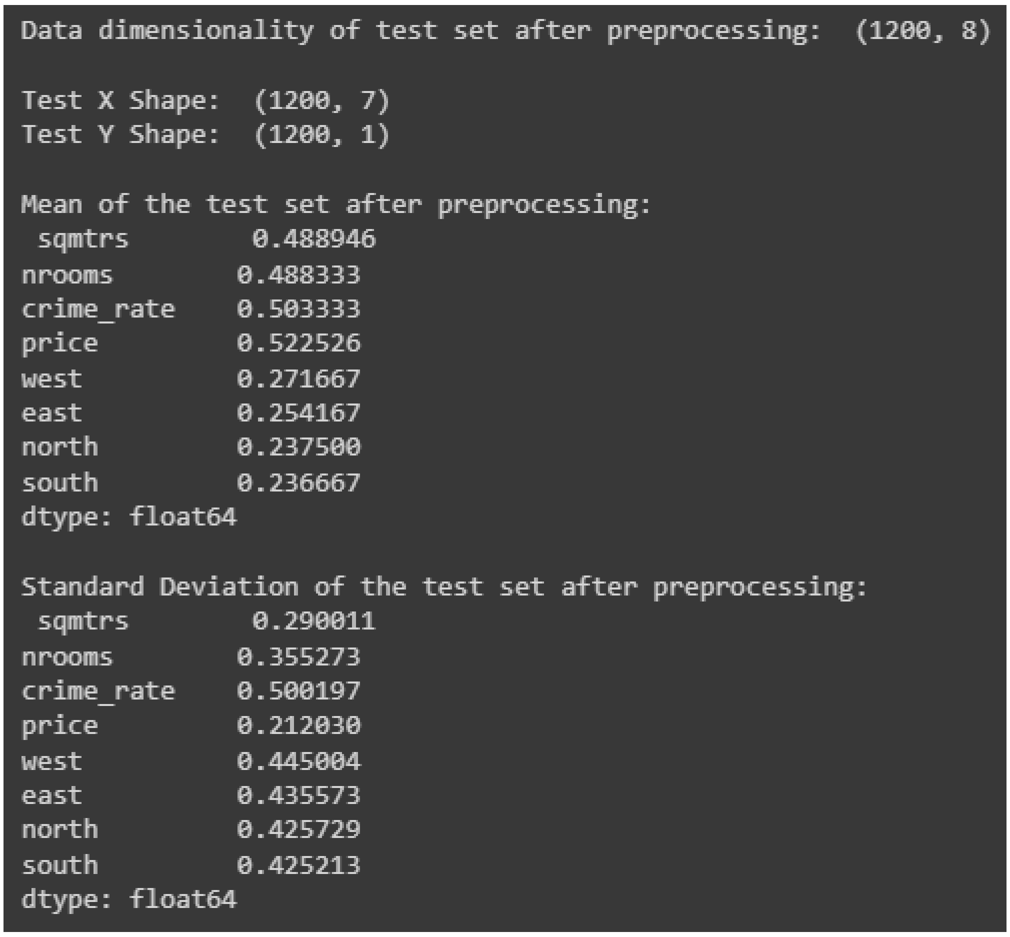

```
Data dimensionality of validation set is:  (480, 8)

Val X Shape:  (480, 7)
Val Y Shape:  (480, 1)

Mean of the validation set is:
 sqmtrs        0.491022
nrooms        0.502083
crime_rate    0.518750
west          0.229167
east          0.266667
north         0.250000
south         0.254167
price         0.503941
dtype: float64

Standard Deviation of the validation set is:
 sqmtrs        0.291518
nrooms        0.359040
crime_rate    0.500170
west          0.420735
east          0.442678
north         0.433464
south         0.435846
price         0.220180
dtype: float64
```

- **Test Set**

  Initially test set consists of 1200 rows and 5 columns. The mean and the standard deviation of the numerical values in the test set is:


```
Data dimensionality of test set is:  (1200, 5)

Mean of the train set is:
 sqmtrs       221.747500
nrooms         2.953333
price     718718.940900
dtype: float64

Standard Deviation of the train set is:
 sqmtrs        72.212760
nrooms         1.421093
price     148792.625998
dtype: float64
```

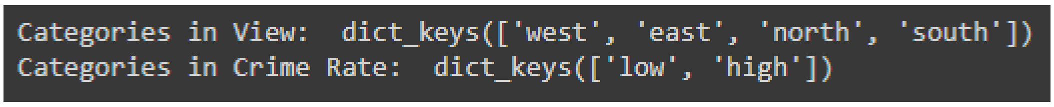
NOTE: After preprocessing shape and mean/standard deviation of the test set becomes like the below due to additional columns and the MinMaxScaler operation.



```
Data dimensionality of test set after preprocessing:  (1200, 8)

Test X Shape:  (1200, 7)
Test Y Shape:  (1200, 1)

Mean of the test set after preprocessing:
 sqmtrs        0.488946
nrooms        0.488333
crime_rate    0.503333
price         0.522526
west          0.271667
east          0.254167
north         0.237500
south         0.236667
dtype: float64

Standard Deviation of the test set after preprocessing:
 sqmtrs        0.290011
nrooms        0.355273
crime_rate    0.500197
price         0.212030
west          0.445004
east          0.435573
north         0.425729
south         0.425213
dtype: float64
```

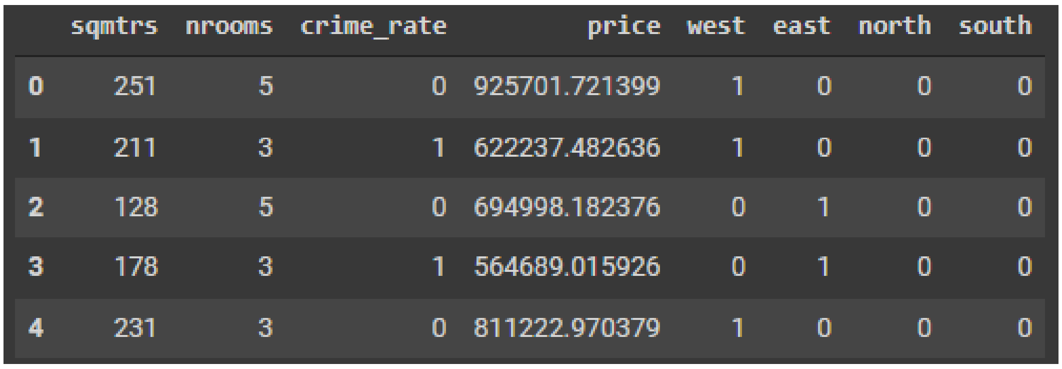
Preprocessing

First unique values in the 'view' and 'crime_rate' obtained by using a dictionary. The unique values are as the following:



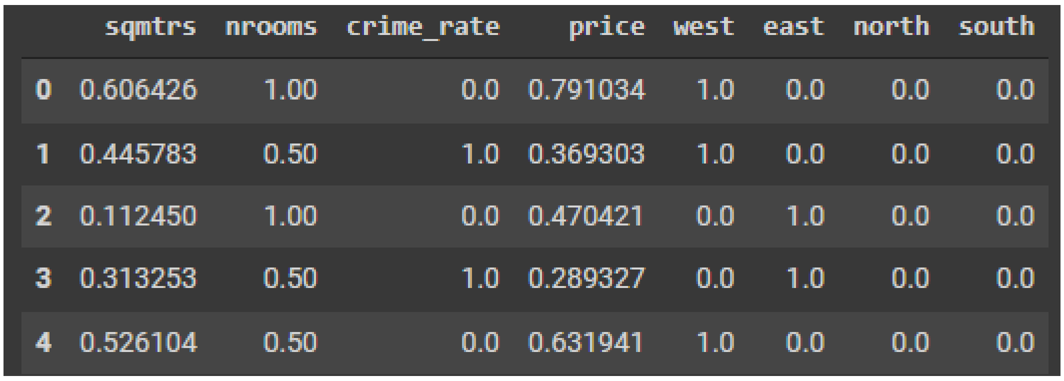
```
Categories in View:  dict_keys(['west', 'east', 'north', 'south'])
Categories in Crime Rate:  dict_keys(['low', 'high'])
```

In the preprocessing step, categorical values such as 'view' and 'crime_rate' transformed into numerical values. In the sake of obtaining better results, views are divided into separate columns. In 'crime_rates' the value 'low' mapped to 0 and the value 'high' mapped to 1.



	sqmtrs	nrooms	crime_rate	price	west	east	north	south
0	251	5	0	925701.721399	1	0	0	0
1	211	3	1	622237.482636	1	0	0	0
2	128	5	0	694998.182376	0	1	0	0
3	178	3	1	564689.015926	0	1	0	0
4	231	3	0	811222.970379	1	0	0	0

After every value in the dataset transformed into numerical values, MinMaxScaler applied in order to scale the values between [0 - 1].



	sqmtrs	nrooms	crime_rate	price	west	east	north	south
0	0.606426	1.00	0.0	0.791034	1.0	0.0	0.0	0.0
1	0.445783	0.50	1.0	0.369303	1.0	0.0	0.0	0.0
2	0.112450	1.00	0.0	0.470421	0.0	1.0	0.0	0.0
3	0.313253	0.50	1.0	0.289327	0.0	1.0	0.0	0.0
4	0.526104	0.50	0.0	0.631941	1.0	0.0	0.0	0.0

Before applying train_test_split to create validation set, train_df (dataframe of the train set) divided into X_train and y_train dataframes. Since it is aimed to predict the price of the houses, X_train consists of 'sqmtrs', 'nrooms', 'west', 'east', 'north', 'south', 'crime_rate’ labels and y_train consists of the ’price’ label.

After this X-y separation, validation set created from the X_train and y_train with the ratio of 0.1.

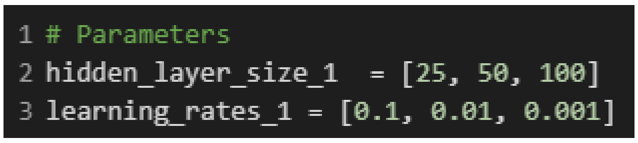
NOTE: Same operations also applied to the test set in the later stages, in order to standardize the datasets.

Train the Models

- **One Layered Neural Network Model**

  In order to easily create models with different hidden layer size and learning rate, a “one-layered model generation” function is written. The function “One_Hidden_Layer_Model” takes ‘hidden_layer_size’ and ‘_learning_rate’ as its parameters, to generate different models faster and easier.

  “Hidden Layer Sizes” and “Learning Rates” that are tested on models:



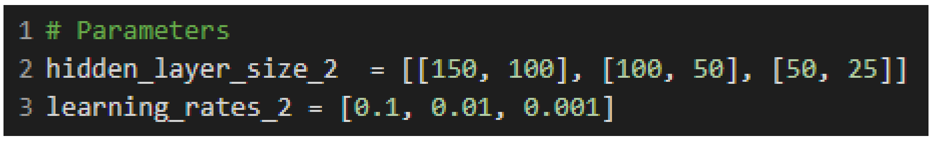
```
# Parameters
hidden_layer_size_1  = [25, 50, 100]
learning_rates_1 = [0.1, 0.01, 0.001]
```

In total, 9 one layered neural network model is generated and tested.

- **Two Layered Neural Network Model**

  In order to easily create models with different hidden layer size and learning rate, a “two-layered model generation” function is written. The function “Two_Hidden_Layer_Model” takes ‘hidden_layer_first’, ‘hidden_layer_second’, and ‘_learning_rate’ as its parameters, to generate different models faster and easier.

  “Hidden Layer Sizes” and “Learning Rates” that are tested on models:

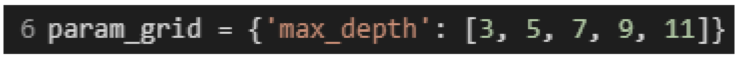


```
# Parameters
hidden_layer_size_2  = [[150, 100], [100, 50], [50, 25]]
learning_rates_2 = [0.1, 0.01, 0.001]
```

  In total, 9 two layered neural network model is generated and tested.

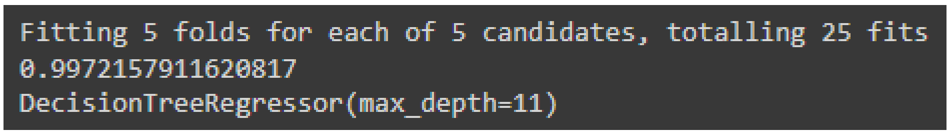
- **Decision Tree**

  In decision tree, GridSearchCV applied to obtain the optimal “max_depth” of the tree. Tested max_depths are:



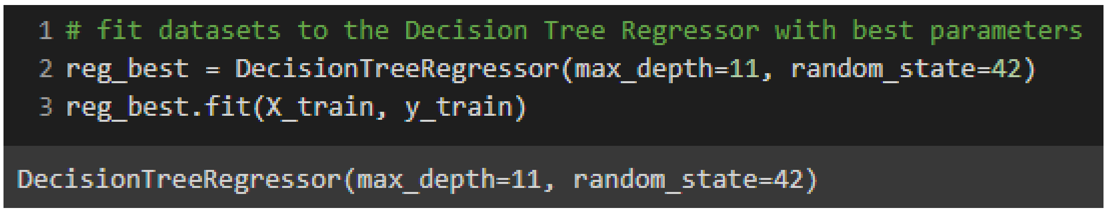
```
param_grid = {'max_depth': [3, 5, 7, 9, 11]}
```

  As the result of the GridSearchCV, best parameter obtained for the depth of the Decision Tree Regressor is max_depth = 11.



```
Fitting 5 folds for each of 5 candidates, totalling 25 fits
0.9972157911620817
DecisionTreeRegressor(max_depth=11)
```

  Therefore, Decision Tree Regressor model is applied to the dataset with max_depth = 11 and random_state = 42.



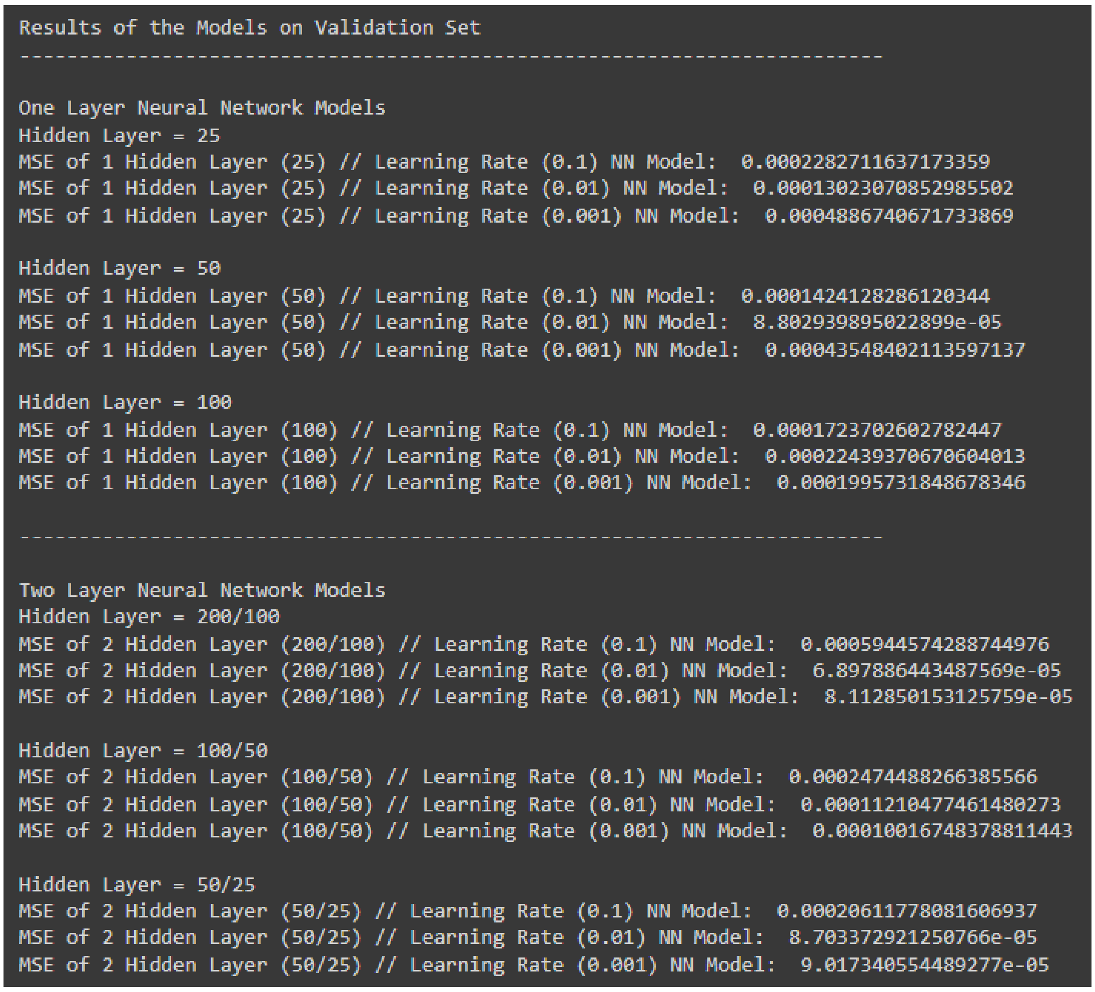
```
# fit datasets to the Decision Tree Regressor with best parameters
reg_best = DecisionTreeRegressor(max_depth=11, random_state=42)
reg_best.fit(X_train, y_train)
```

```
DecisionTreeRegressor(max_depth=11, random_state=42)
```

Results / Observations

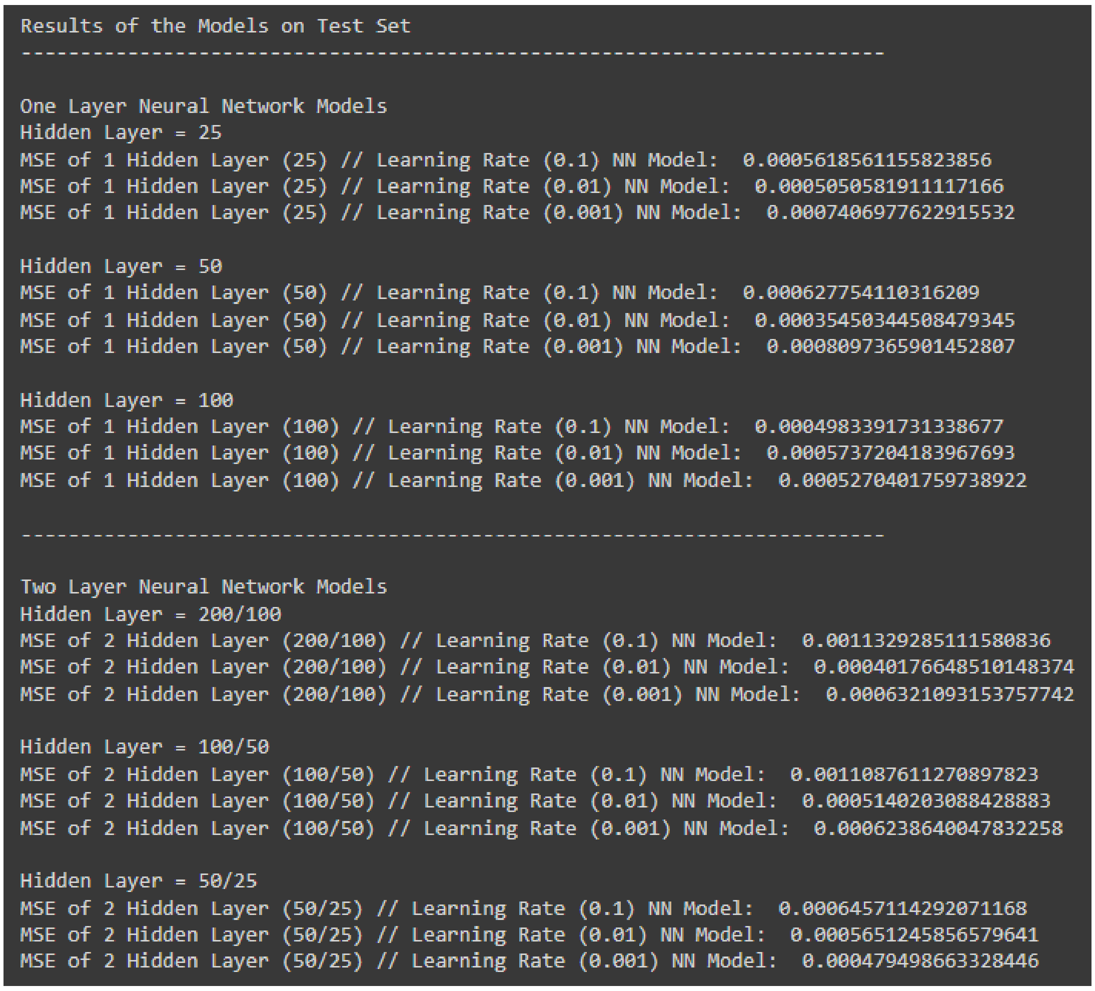
Learning rate gets lower execution gets faster

- **Neural Network Models**
  - **Tested on Validation Set**



```
Results of the Models on Validation Set
------------------------------------------------------------------------

One Layer Neural Network Models
Hidden Layer = 25
MSE of 1 Hidden Layer (25) // Learning Rate (0.1) NN Model:  0.0002282711637173359
MSE of 1 Hidden Layer (25) // Learning Rate (0.01) NN Model:  0.00013023070852985502
MSE of 1 Hidden Layer (25) // Learning Rate (0.001) NN Model:  0.0004886740671733869

Hidden Layer = 50
MSE of 1 Hidden Layer (50) // Learning Rate (0.1) NN Model:  0.00014241282861 20344
MSE of 1 Hidden Layer (50) // Learning Rate (0.01) NN Model:  8.802939895022899e-05
MSE of 1 Hidden Layer (50) // Learning Rate (0.001) NN Model:  0.00043548402113597137

Hidden Layer = 100
MSE of 1 Hidden Layer (100) // Learning Rate (0.1) NN Model:  0.0001723702602782447
MSE of 1 Hidden Layer (100) // Learning Rate (0.01) NN Model:  0.00022439370670604013
MSE of 1 Hidden Layer (100) // Learning Rate (0.001) NN Model:  0.0001995731848678346

------------------------------------------------------------------------

Two Layer Neural Network Models
Hidden Layer = 200/100
MSE of 2 Hidden Layer (200/100) // Learning Rate (0.1) NN Model:  0.0005944574288744976
MSE of 2 Hidden Layer (200/100) // Learning Rate (0.01) NN Model:  6.897886443487569e-05
MSE of 2 Hidden Layer (200/100) // Learning Rate (0.001) NN Model:  8.112850153125759e-05

Hidden Layer = 100/50
MSE of 2 Hidden Layer (100/50) // Learning Rate (0.1) NN Model:  0.0002474488266385566
MSE of 2 Hidden Layer (100/50) // Learning Rate (0.01) NN Model:  0.00011210477461480273
MSE of 2 Hidden Layer (100/50) // Learning Rate (0.001) NN Model:  0.00010016748378811443

Hidden Layer = 50/25
MSE of 2 Hidden Layer (50/25) // Learning Rate (0.1) NN Model:  0.00020611778081606937
MSE of 2 Hidden Layer (50/25) // Learning Rate (0.01) NN Model:  8.703372921250766e-05
MSE of 2 Hidden Layer (50/25) // Learning Rate (0.001) NN Model:  9.017340554489277e-05
```

  - **Tested on Test Set:**



```
Results of the Models on Test Set
--------------------------------------------------------------------------

One Layer Neural Network Models
Hidden Layer = 25
MSE of 1 Hidden Layer (25) // Learning Rate (0.1) NN Model:  0.0005618561155823856
MSE of 1 Hidden Layer (25) // Learning Rate (0.01) NN Model:  0.0005050581911117166
MSE of 1 Hidden Layer (25) // Learning Rate (0.001) NN Model:  0.0007406977622915532

Hidden Layer = 50
MSE of 1 Hidden Layer (50) // Learning Rate (0.1) NN Model:  0.000627754110316209
MSE of 1 Hidden Layer (50) // Learning Rate (0.01) NN Model:  0.00035450344508479345
MSE of 1 Hidden Layer (50) // Learning Rate (0.001) NN Model:  0.0008097365901452807

Hidden Layer = 100
MSE of 1 Hidden Layer (100) // Learning Rate (0.1) NN Model:  0.0004983391731338677
MSE of 1 Hidden Layer (100) // Learning Rate (0.01) NN Model:  0.0005737204183967693
MSE of 1 Hidden Layer (100) // Learning Rate (0.001) NN Model:  0.0005270401759738922

--------------------------------------------------------------------------

Two Layer Neural Network Models
Hidden Layer = 200/100
MSE of 2 Hidden Layer (200/100) // Learning Rate (0.1) NN Model:  0.0011329285111580836
MSE of 2 Hidden Layer (200/100) // Learning Rate (0.01) NN Model:  0.00040176648510148374
MSE of 2 Hidden Layer (200/100) // Learning Rate (0.001) NN Model:  0.0006321093153757742

Hidden Layer = 100/50
MSE of 2 Hidden Layer (100/50) // Learning Rate (0.1) NN Model:  0.0011087611270897823
MSE of 2 Hidden Layer (100/50) // Learning Rate (0.01) NN Model:  0.0005140203088428883
MSE of 2 Hidden Layer (100/50) // Learning Rate (0.001) NN Model:  0.0006238640047832258

Hidden Layer = 50/25
MSE of 2 Hidden Layer (50/25) // Learning Rate (0.1) NN Model:  0.0006457114292071168
MSE of 2 Hidden Layer (50/25) // Learning Rate (0.01) NN Model:  0.0005651245856579641
MSE of 2 Hidden Layer (50/25) // Learning Rate (0.001) NN Model:  0.000479498663328446
```

- **Observations:**
  - **One Layer Neural Network Models:**
    - For Hidden Layer Size = 25:

      [Test on Validation Set] Learning Rate 0.01 provided the best result for the model with Hidden Layer Size = 50.

      [Test on Test Set] Learning Rate 0.1 and 0.01 provided close mean squared errors for the model with Hidden Layer Size = 25. But, with a small difference Learning Rate 0.01 provides the best result.
    - For Hidden Layer Size = 50:

      [Test on Validation Set] Learning Rate 0.1 provided the best result for the model with Hidden Layer Size = 100.

      [Test on Test Set] Learning Rate 0.01 provided the best result for the model with Hidden Layer Size = 50.

- For Hidden Layer Size = 100:

  [Test on Validation Set] Learning Rate 0.1 provided the best result for the model with Hidden Layer Size = 100.

  [Test on Test Set] Learning Rate 0.1 provided the best result for the model with Hidden Layer Size = 100.

- **Two Layer Neural Network Models:**
  - For Hidden Layer Size = 200/100:

    [Test on Validation Set] Learning Rate 0.1 provided the best result for the model with Hidden Layer Sizes = 200/100.

    [Test on Test Set] Learning Rate 0.01 provided the best result for the model with Hidden Layer Sizes = 200/100.

  - For Hidden Layer Size = 100/50:

    [Test on Validation Set] Learning Rate 0.01 provided the best result for the model with Hidden Layer Sizes = 100/50.

    [Test on Test Set] Learning Rate 0.01 provided the best result for the model with Hidden Layer Sizes = 100/50.

  - For Hidden Layer Size = 50/25:

    [Test on Validation Set] Learning Rate 0.1 provided the best result for the model with Hidden Layer Sizes = 50/25.

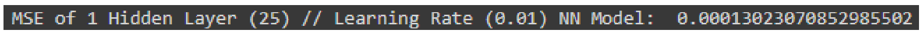
    [Test on Test Set] Learning Rate 0.001 provided the best result for the model with Hidden Layer Sizes = 50/25.

- **Overall Observation About Neural Network Models**
  1. Overall learning rates between [0.1 - 0.01] provided better results for the models.
  2. As the hidden layer size increases, models require larger learning rates, in order to provide better results
  3. Best results provided by the model with Hidden Layer Size
  4. Best results on the validation set obtained with the One Hidden Layer approach using a value of Hidden Layer Size = 25 and

Learning Rate = 0.01 as parameters.

- Validation Set / Best Scoring Model:

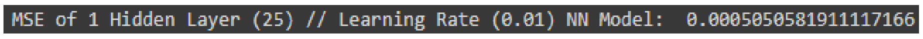
  The result of this model on the validation data is 0.0001% mean squared error.



```
MSE of 1 Hidden Layer (25) // Learning Rate (0.01) NN Model:  0.00013023070852985502
```

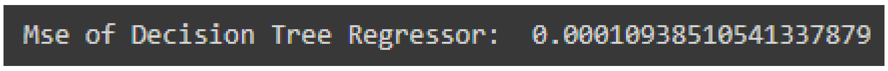
- Test Set / Best Scoring Model:

  The result of this model on the test data is 0.0003% mean squared error.



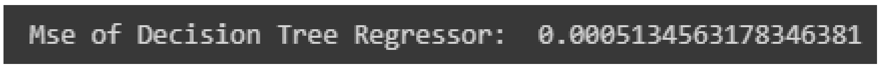
```
MSE of 1 Hidden Layer (25) // Learning Rate (0.01) NN Model:  0.0005050581911117166
```

- **Decision Tree (max_depth = 11)**
  - **Tested on Validation Set**



```
Mse of Decision Tree Regressor:  0.00010938510541337879
```

  - **Tested on Test Set**



```
Mse of Decision Tree Regressor:  0.0005134563178346381
```

  - **Observation**

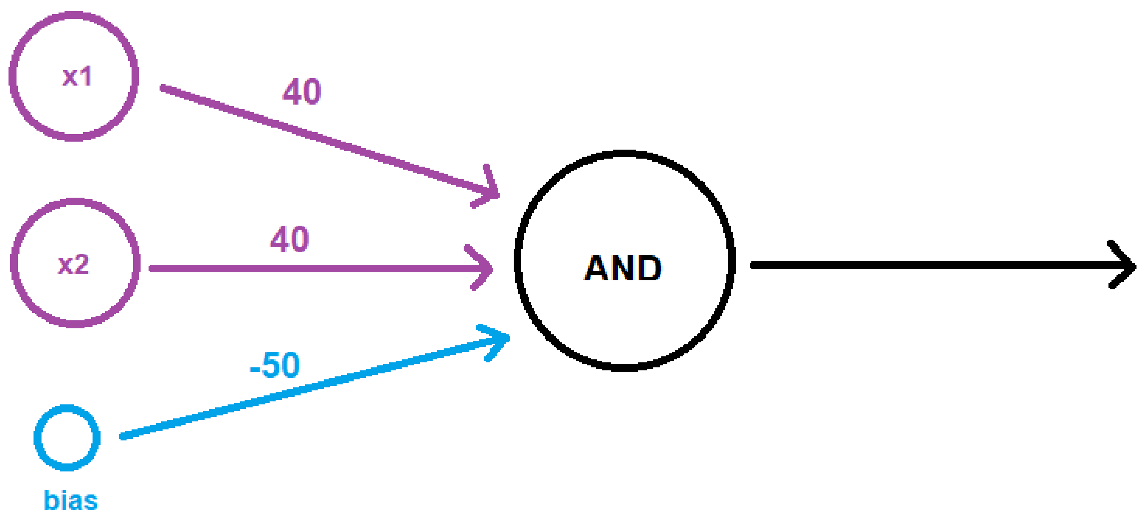
    Decision Tree Regressor Model also obtained very good result with a very small mean squared error on this regression task.

Question #2

Considering a weight vector w = [bias, w1 w2]. What could be the weights (bias, w1 and w2) of a neuron that implements the Boolean AND function of its two inputs? (AND: Output is 1 if and only if both inputs are 1.)

x_1	x_2	$x_1 \wedge x_2$	$w_1 = 40,\ w_2 = 40,\ bias = -50$	Output of the Neuron
1	1	1	$1 * 40 + 1 * 40 - 50 = 30$	Positive
1	0	0	$1 * 40 + 0 * 40 - 50 = -10$	Negative
0	1	0	$0 * 40 + 1 * 40 - 50 = -10$	Negative
0	0	0	$0 * 40 + 0 * 40 - 50 = -50$	Negative

Therefore, the weights and bias could be: $w_1 = 40,\ w_2 = 40,\ bias = -50$



Question #3

Consider a neuron using the threshold activation function,

- 3 inputs: $x_1,\ x_2,\ x_3$
- Weight vector w = [1, 3 , -1] and bias b= -1.

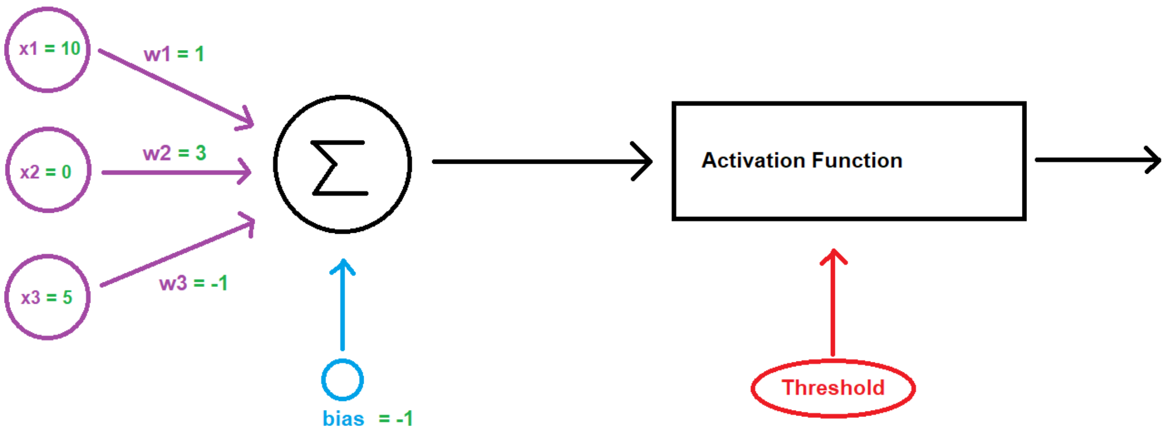
What would be the net input for an input of x = [10, 0, 5]?

What would be the output for the same input?

Net Input = $\sum(weight\ inputs)\ +\ bias$

Net Input = $1 * 10 + 3 * 0 + (-1) * 5 + (-1)$

= $10 + 0 + (-5) + (-1) = 4$



if Net Input ≥ 0 → Threshold(Net Input) = 1

if Net Input < 0 → Threshold(Net Input) = 0

Since Net Input = 4 > 0 → Threshold = 1

f(x) = 1 if x > 0

f(x) = 0 if x < 0

<u>Output = f(4) = 1</u>