CS577-Assignment 3

Single Output Regression

Name : Sourav Yadav **Spring 2020**

AID: A20450418

1.Problem Statement :

Predication of Boston Housing Prices using different loss function and optimization techniques.

2.Proposed Solution :

We will design a neural network in Python with Keras and then train the network on train and validation data. Finally we will evaluate performance of the network on test data.

3.Implementation details :

- We first load the Boston Housing Data from the below link. Attribute Information is also given.
  https://archive.ics.uci.edu/ml/machine-learning-databases/housing/

Attribute Information:

1. CRIM per capita crime rate by town
2. ZN proportion of residential land zoned for lots over 25,000 sq.ft.
3. INDUS proportion of non-retail business acres per town
4. CHAS Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
5. NOX nitric oxides concentration (parts per 10 million)
6. RM average number of rooms per dwelling
7. AGE proportion of owner-occupied units built prior to 1940
8. DIS weighted distances to five Boston employment centres
9. RAD index of accessibility to radial highways
10. TAX full-value property-tax rate per $10,000
11. PTRATIO pupil-teacher ratio by town
12. B 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
13. LSTAT % lower status of the population
14. MEDV Median value of owner-occupied homes in $1000's

- We split the Iris data in Train , Test and validation Set using train_test_split() method in 8:2 ratio.
- Then we normalized the features.
- We designed neural network with 13 nodes in the input layers , two hidden layers followed by an output node with no activation function which will return predicated prices of the house.

- Activation used in the hidden layers is Relu and no Activation used in the output node .
- We designed various models using different Loss Functions, Optimization Techniques and regularization methods.
- Now we train our network with train data using K -fold cross validation and observed its performance . Results are discussed in below section.

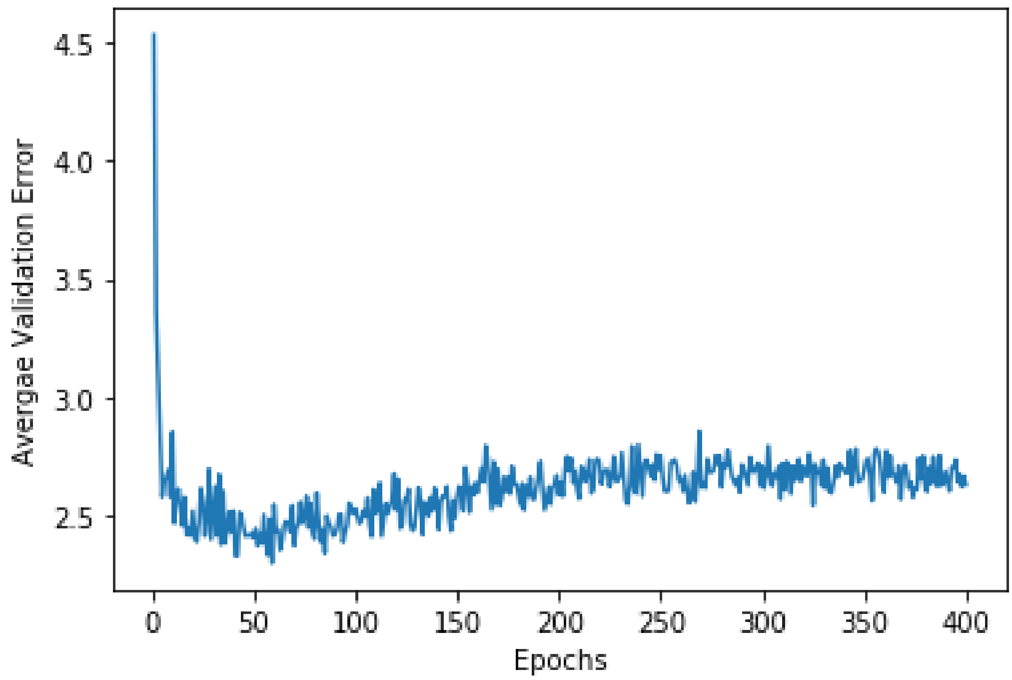
4. Results and discussion:

- Initial hyperparameters used during training.
  Learning Rate : 0.01
  Epochs : 400

Evaluating Different Loss Function:

Mean Square Error :

- After hyperparameter tuning , we got below results .

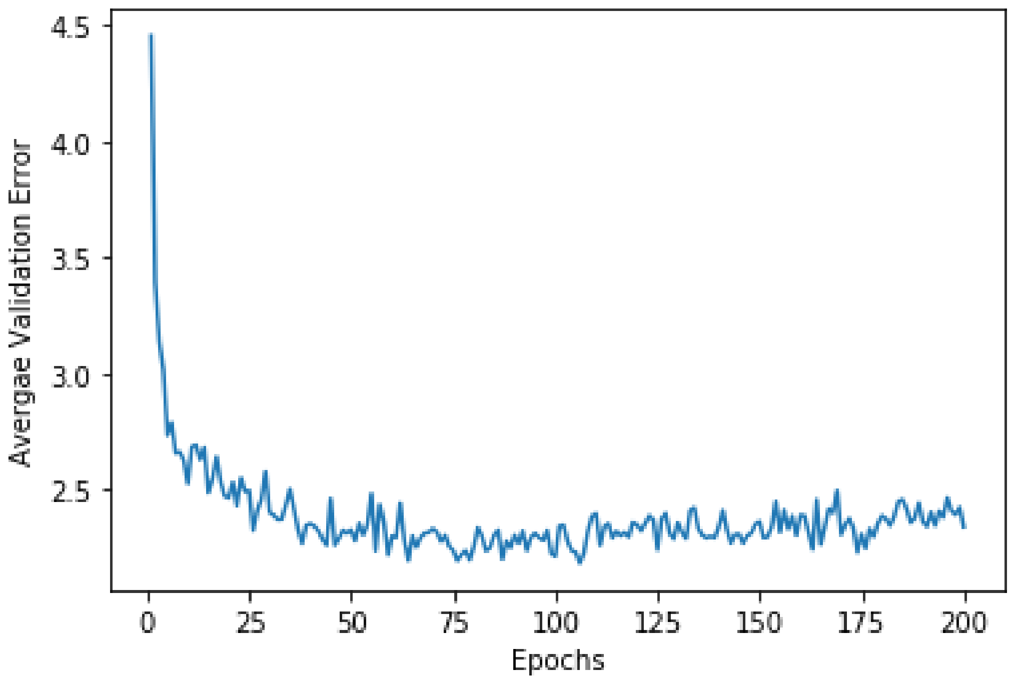


- From the above graph we see that our models validation loss starts to increase after 50 epochs and hence we choose our final number of epochs as 50.
- Finally we evaluate our network on test data and got below results.

```
After evaluating on test data:
Loss Value: 14.718111524394914
Mean Absolute error: 2.4721736907958984
```

Mean Absolute Error:

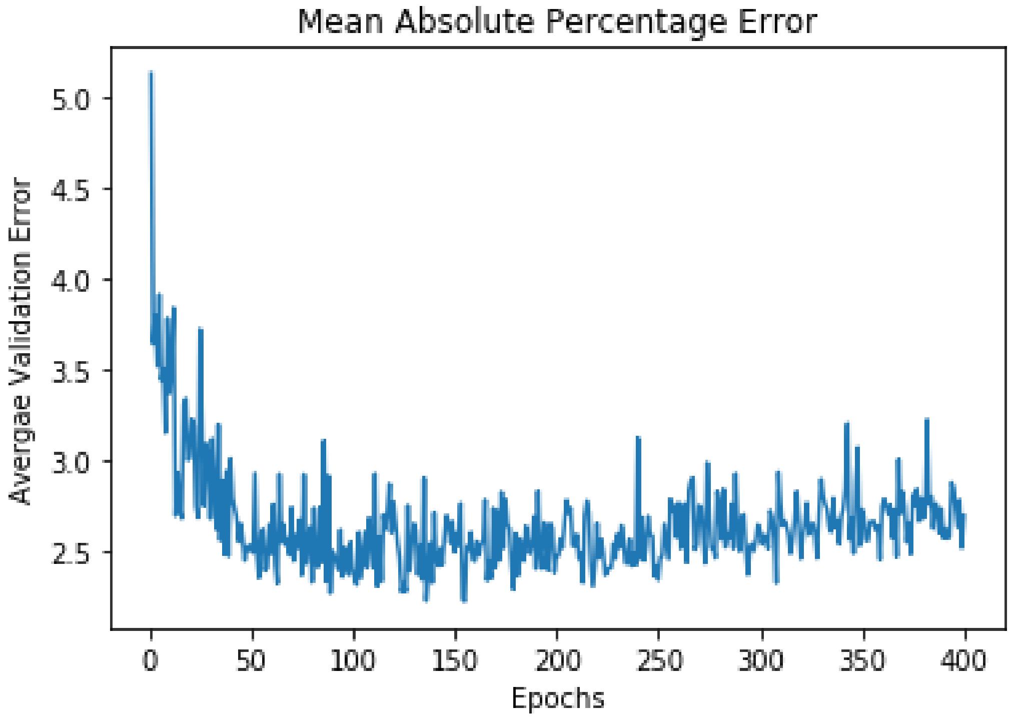
- After hyperparameter tuning , we got below results .



- From the above graph we see that our models validation loss starts to increase after 75 epochs and hence we choose our final number of epochs as 75.
- Finally we evaluate our network on test data and got below results.
-
```
Loss Value: 2.218156880023433
Mean Absolute error: 2.2181570529937744
```

Mean Absolute Percentage Error:

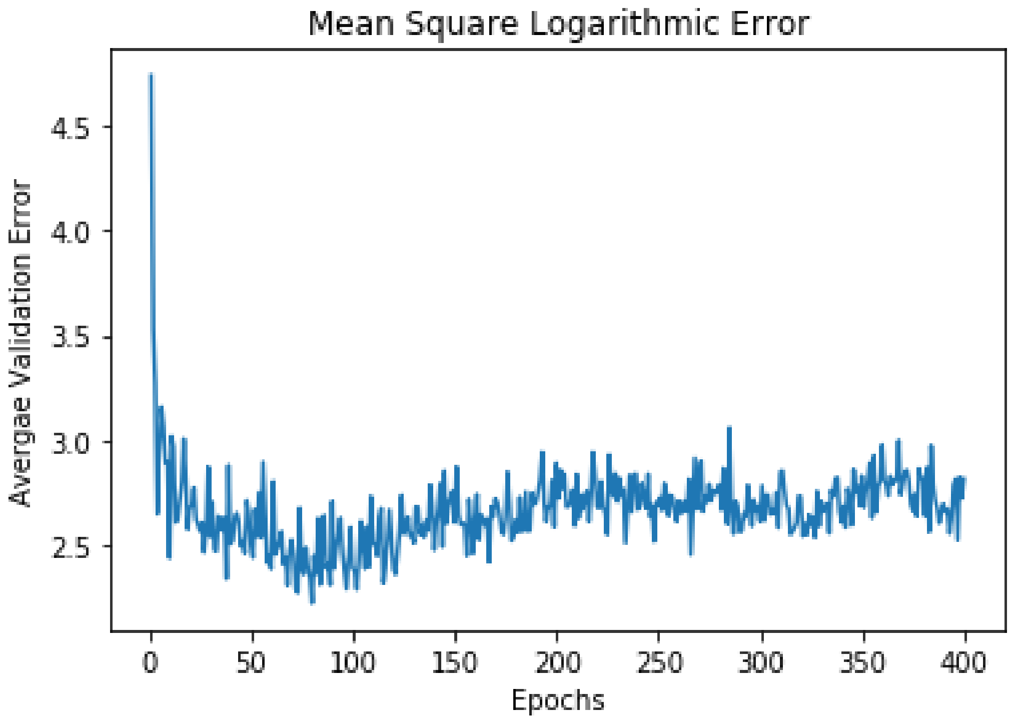
- After hyperparameter tuning , we got below results .



- From the above graph we see that our models validation loss starts to increase after 100 epochs and hence we choose our final number of epochs as 100.
- Finally we evaluate our network on test data and got below results.
-
```
Loss Value: 14.553347157497033
Mean Absolute error: 2.951164960861206
```

Mean Square Logarithmic Error:

- After hyperparameter tuning , we got below results .

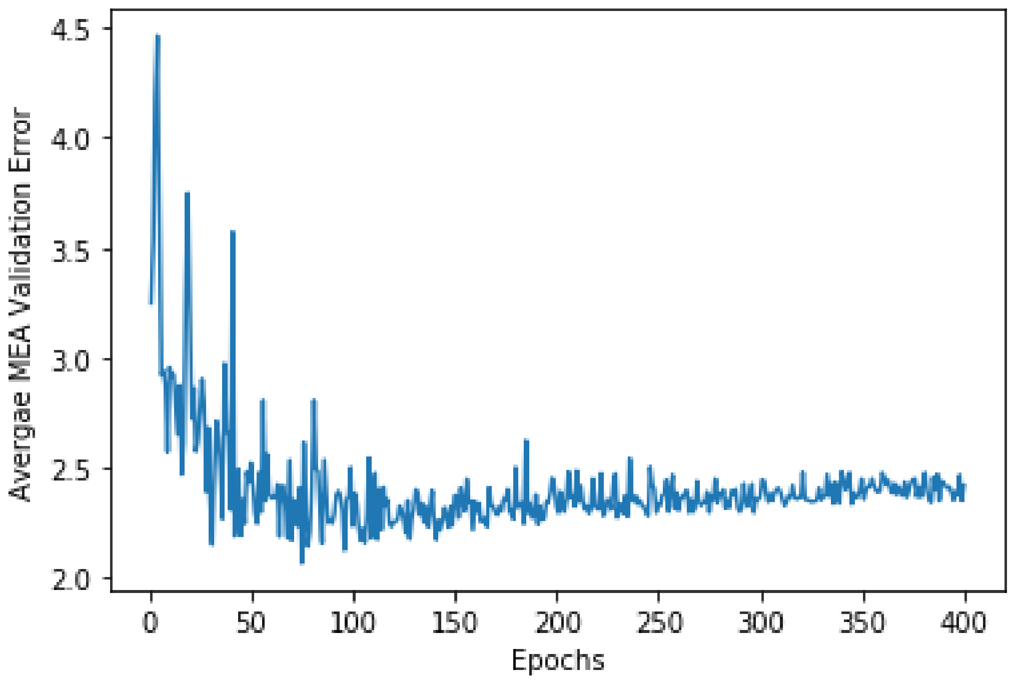


- From the above graph we see that our models validation loss starts to increase after 75 epochs and hence we choose our final number of epochs as 75.
- Finally we evaluate our network on test data and got below results.
-
```
After evaluating on test data:
Loss Value: 0.03959962674507908
Mean Absolute error: 2.89245343208313
```

Evaluating Different Optimizers :

SGD Optimizers :

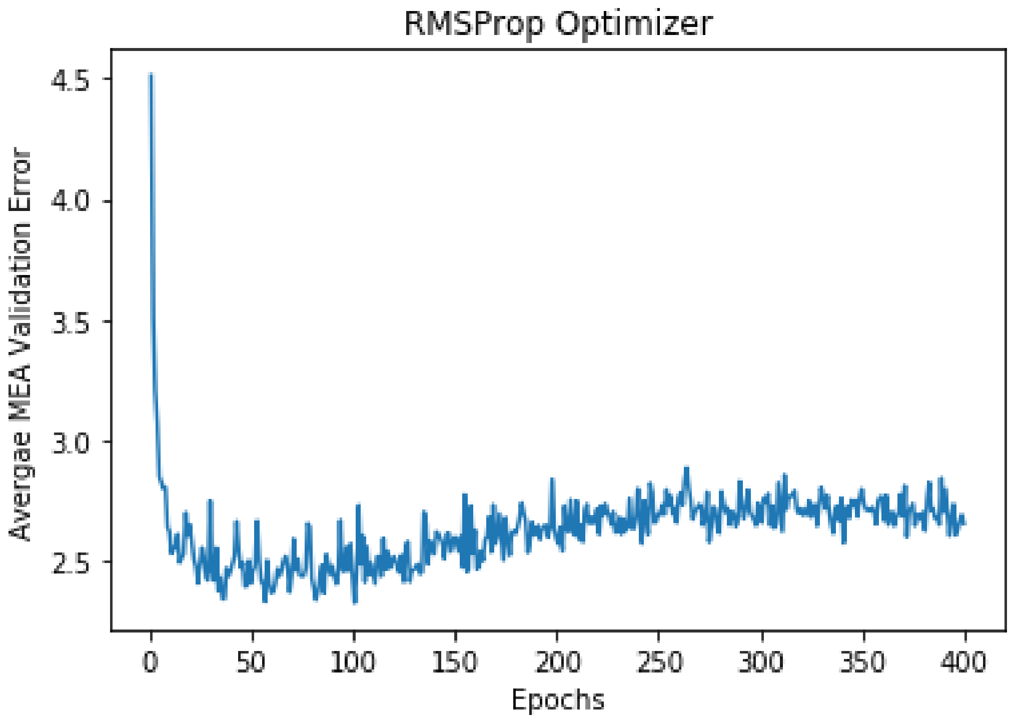
- After hyperparameter tuning , we got below results .



- From the above graph we see that our models validation loss starts to increase after 75 epochs and hence we choose our final number of epochs as 75.
- Finally we evaluate our network on test data and got below results.
-
```
After evaluating on test data:
MSE Loss Value: 15.531299740660424
Mean Absolute error: 2.6186249256134033
```

RMS Prop:

- After hyperparameter tuning , we got below results .

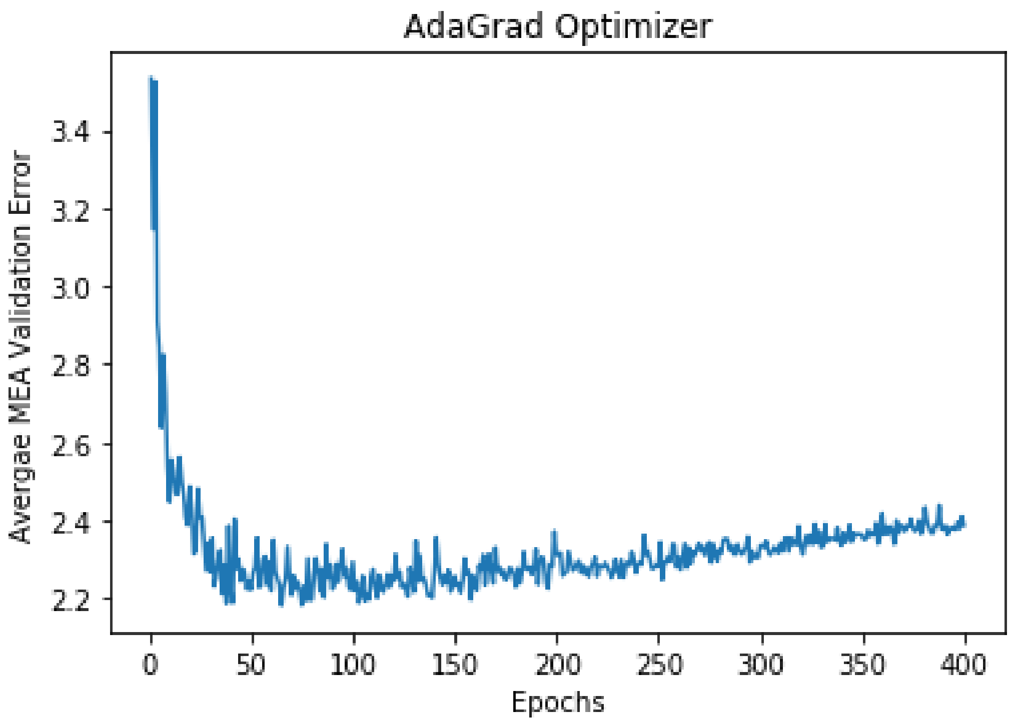


- From the above graph we see that our models validation loss starts to increase after 50 epochs and hence we choose our final number of epochs as 50.
- Finally we evaluate our network on test data and got below results.

```
After evaluating on test data:
MSE Loss Value: 15.854372361127067
Mean Absolute error: 2.3838040828704834
```

AdaGrad Optimizer :

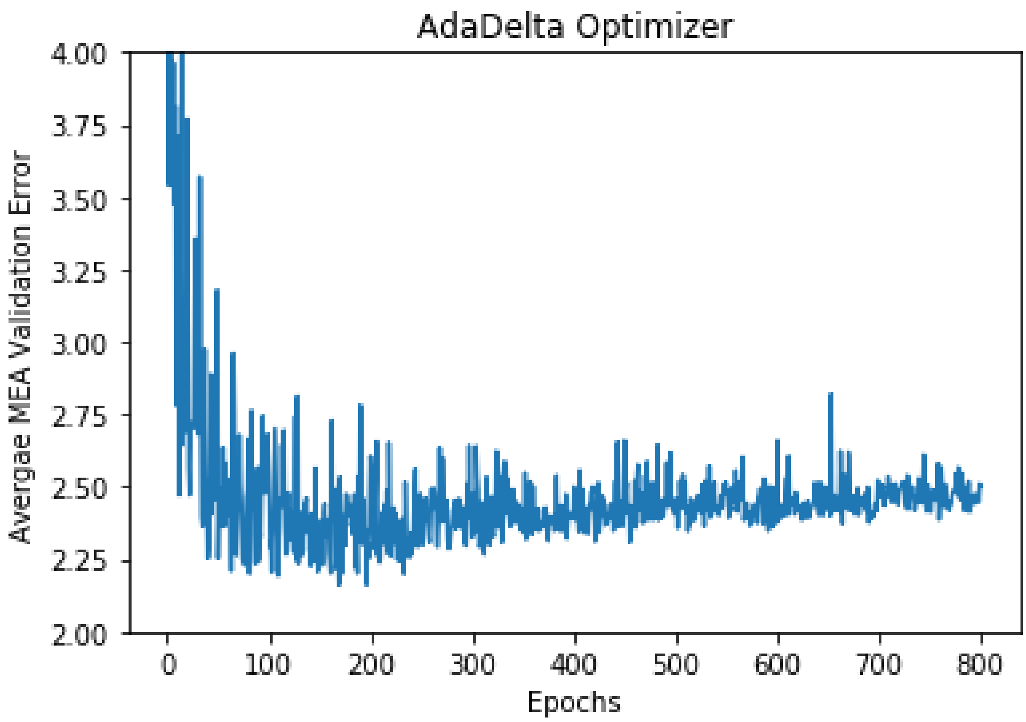
- After hyperparameter tuning , we got below results .



- From the above graph we see that our models validation loss starts to increase after 75 epochs and hence we choose our final number of epochs as 75.
- Finally we evaluate our network on test data and got below results.
-
```
After evaluating on test data:
MSE Loss Value: 12.34331807903215
Mean Absolute error: 2.585848331451416
```

AdaDelta :

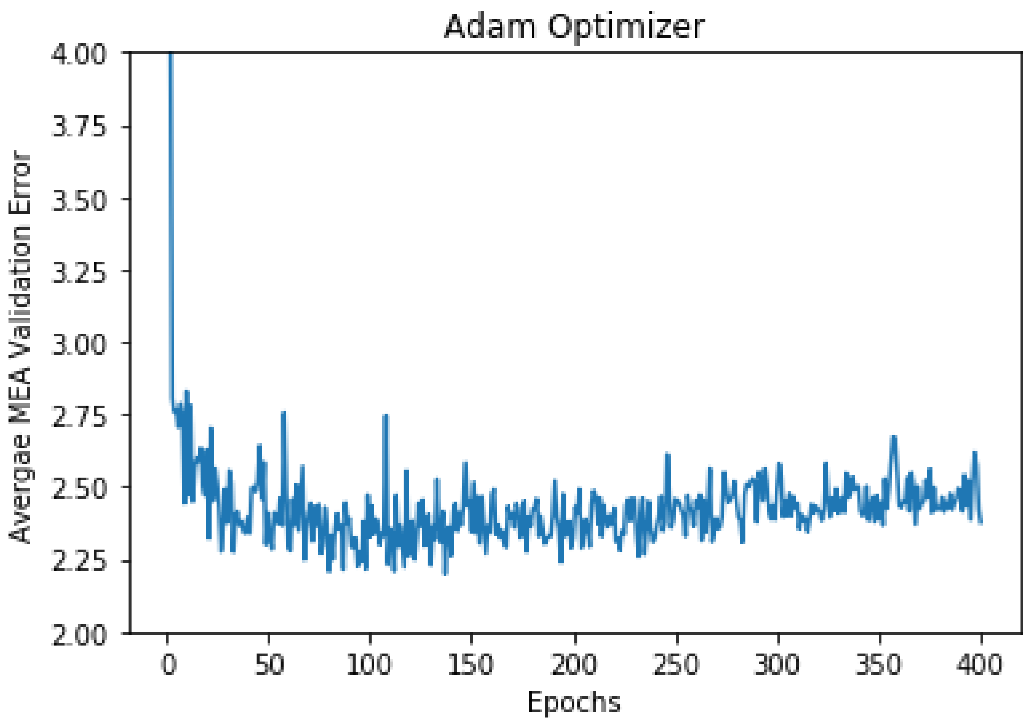
- After hyperparameter tuning , we got below results .



- From the above graph we see that our models validation loss starts to increase after 150 epochs and hence we choose our final number of epochs as 150.
- Finally we evaluate our network on test data and got below results.
-
```
After evaluating on test data:
MSE Loss Value: 11.85307648602654
Mean Absolute error: 2.2152202129364014
```

Adam :

- After hyperparameter tuning , we got below results .

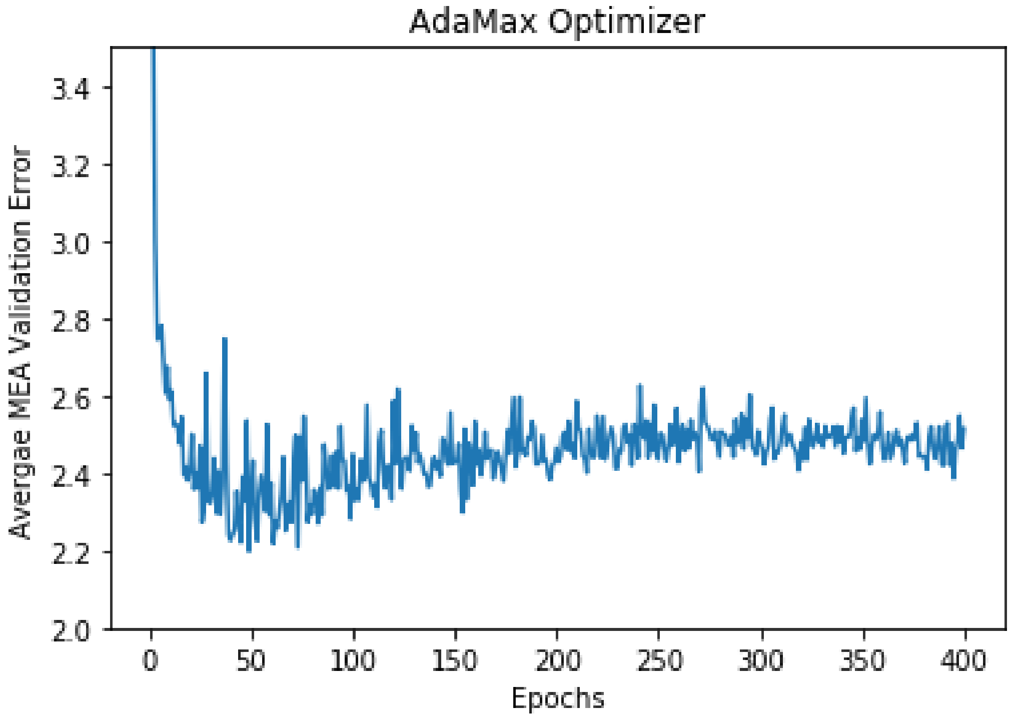


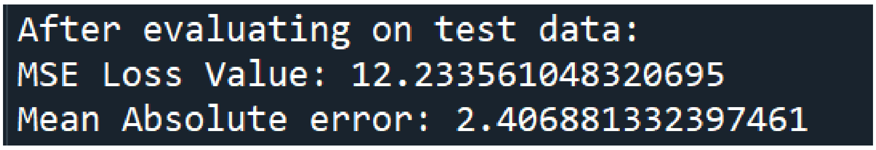
- From the above graph we see that our models validation loss starts to increase after 50 epochs and hence we choose our final number of epochs as 50.
- Finally we evaluate our network on test data and got below results.

```
After evaluating on test data:
MSE Loss Value: 15.678947747922411
Mean Absolute error: 2.5582523345947266
```

AdaMax:

- After hyperparameter tuning , we got below results .

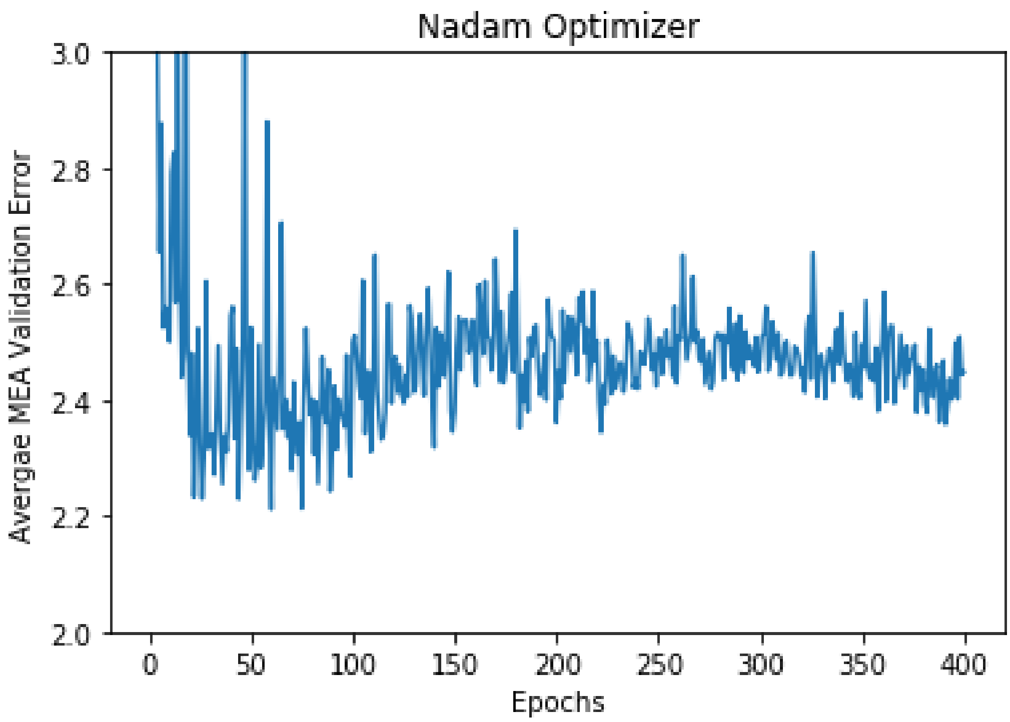


- From the above graph we see that our models validation loss starts to increase after 50 epochs and hence we choose our final number of epochs as 50.
- Finally we evaluate our network on test data and got below results.
- 

```
After evaluating on test data:
MSE Loss Value: 12.233561048320695
Mean Absolute error: 2.406881332397461
```

Nadam :

- After hyperparameter tuning , we got below results .

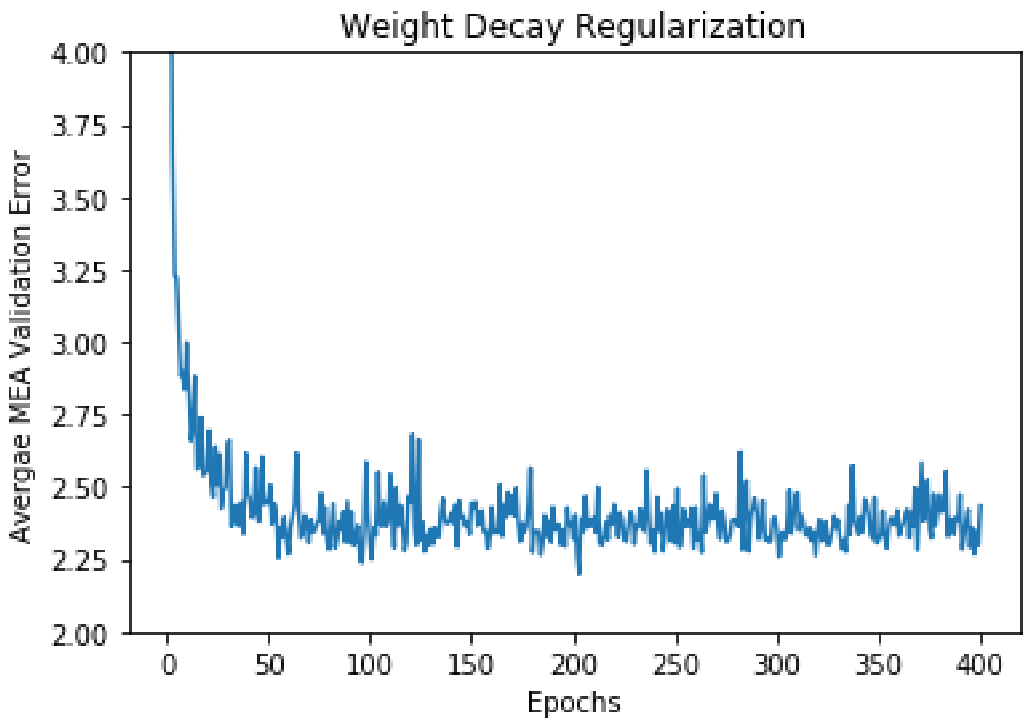


- From the above graph we see that our models validation loss starts to increase after 50 epochs and hence we choose our final number of epochs as 50.
- Finally we evaluate our network on test data and got below results.
-
```
After evaluating on test data:
MSE Loss Value: 19.94035982618145
Mean Absolute error: 3.392071008682251
```

By comparing all of the above Optimizers , we can see that Adam Optimizers gives the best results by converging in around 50 epochs.

Evaluating Different Regularization measures :

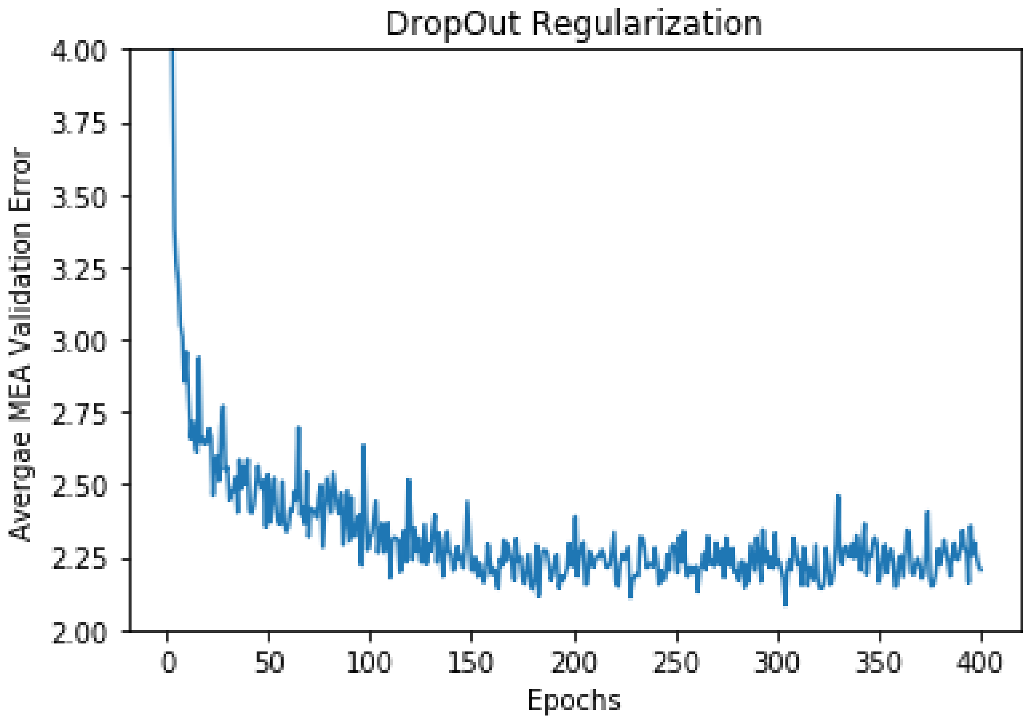
Weight Decay with RMS Prop Optimizer model evaluated above:



From the above graph we can see the using regularization our model generalizes well compared to RMS Prop Optimizer model without regularisation. We can see that validation error doesn't increase after few epochs as compared to previous model which shows better generalisation.

```
After evaluating on test data:
MSE Loss Value: 15.958480236577053
Mean Absolute error: 2.790804147720337
```

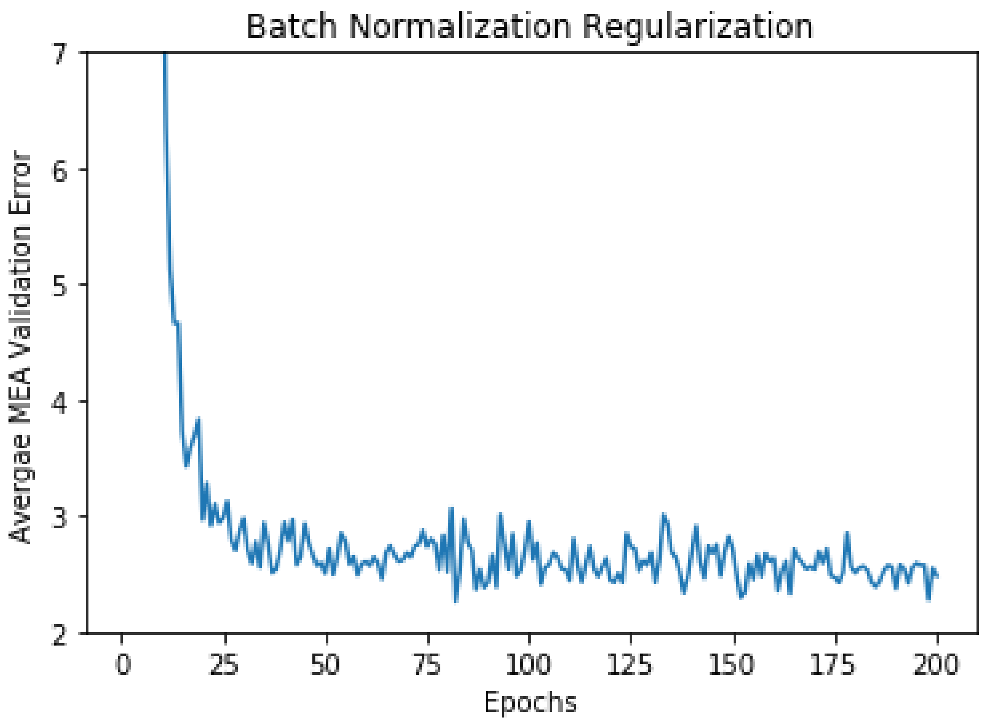
Drop Out with RMS Prop Optimizer model evaluated above:



From the above graph we can see the using regularization our model generalizes well compared to RMS Prop Optimizer model without regularisation. We can see that validation error doesn't increase after few epochs as compared to previous model which shows better generalisation.

```
After evaluating on test data:
MSE Loss Value: 10.934232823988971
Mean Absolute error: 2.2663652896881104
```

Batch Normalization with RMS Prop Optimizer model evaluated above:



From the above graph we can see the using regularization our model generalizes well compared to RMS Prop Optimizer model without regularisation. We can see that validation error doesn't increase after few epochs as compared to previous model which shows better generalisation.

```
After evaluating on test data:
MSE Loss Value: 13.927823908188763
Mean Absolute error: 2.577984571456909
```

Ensemble Classifier using two models : Adam Classifier and RMSProp Classifier:

First we use Adam Classifier to predict values and then use RMSProp Classifier to predict values. Then we can take average of the difference between predicated values and true values to get mean absolute error of the ensemble classifier.

```
Mean Absolute Error 2.2877657207788205
```