1.Problem Statement :

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

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/
- Then we are going to One Hot Encode the labels of the dataset.
- We split the Iris data into Train , Test and validation Set in 80:20 ratio and we normalized the features using below formula.
  Initial Hyperarameters :
  - Input nodes : 4
  - Output nodes : 3, activation for output layer : 'softmax'
  - first hidden layer Hidden nodes in: 64 , activation function: 'relu'
  - second hidden layer Hidden nodes in: 64 , activation function: 'relu'
  - Third hidden layer # of hidden nodes:64, activation function:'relu'
  - Output layer #nodes:1,
  - Loss function: categorical cross entropy

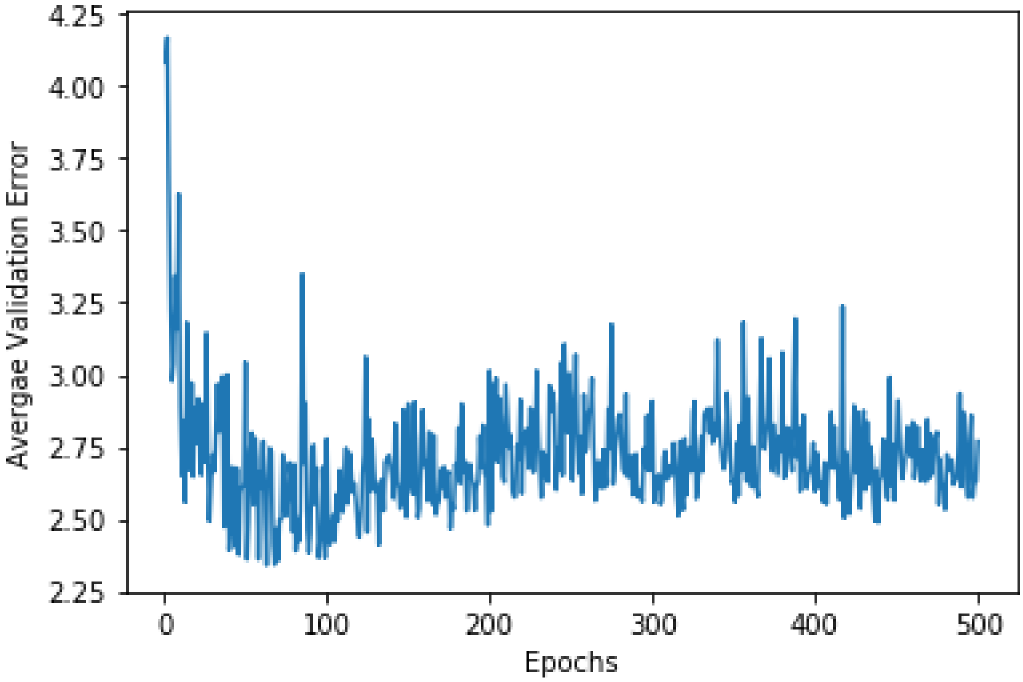
- We split the boston data in Train , Test and validation Set using train_test_split() method in 8:2 ratio.
- Then we normalized the features.
- We build various models using different Loss Functions, Optimization Techniques and regularization methods.
- Now we train our network with train data using 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 : 500

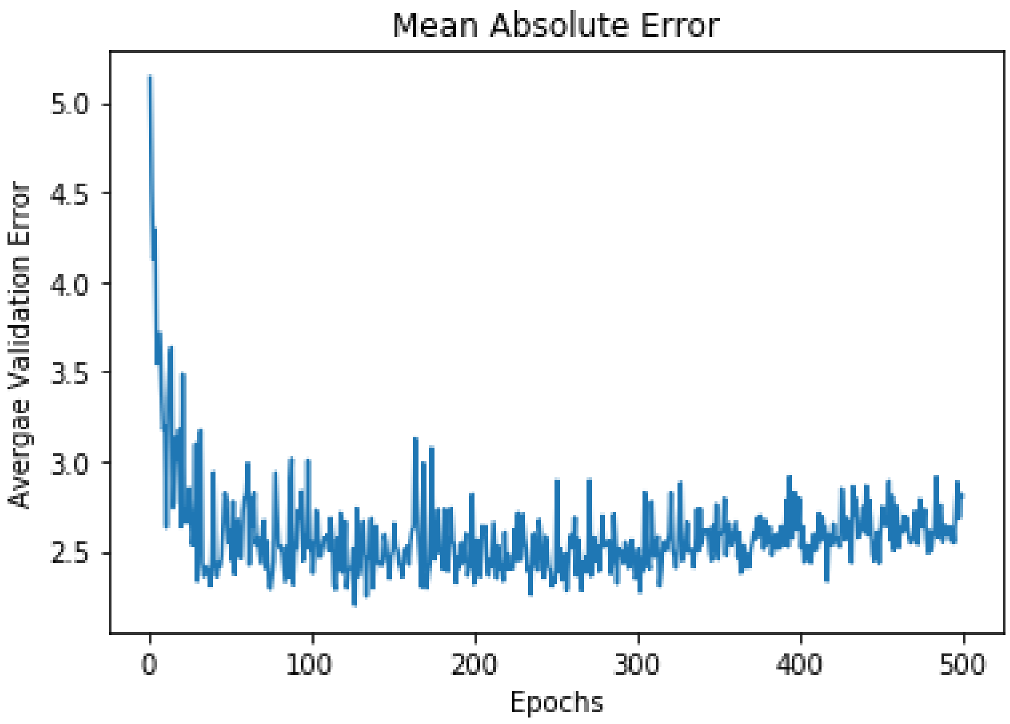
Evaluating Different Loss Function:

Mean Square Error :



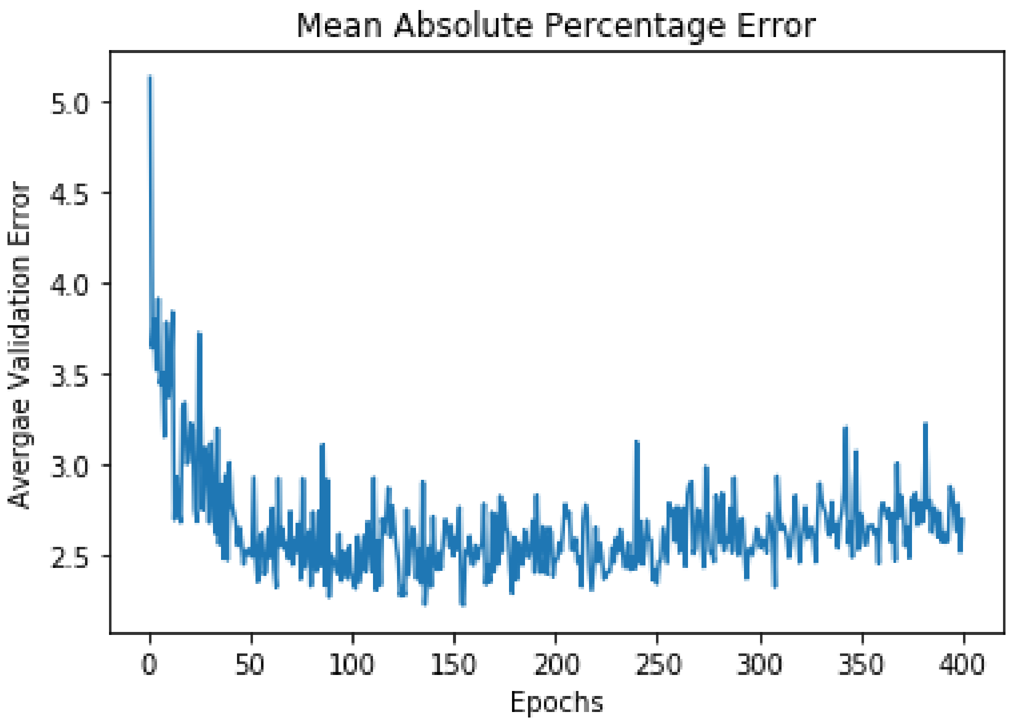
- From the above graph we see that our models validation loss starts to increase after 80 epochs and hence we choose our final number of epochs as 80.
- Finally we evaluate our network on test data and got below results.
- After evaluating on test data:
    - Loss Value: 12.806650423536114
    - Mean Absolute error: 2.5503060817718506

Mean Absolute Error:



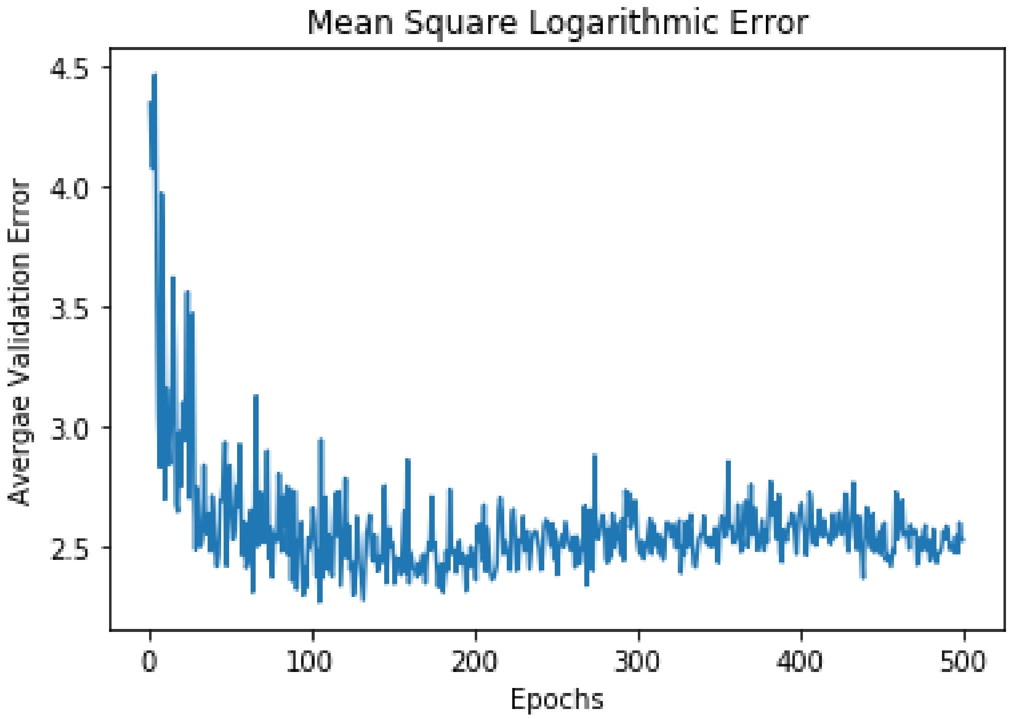
- From the above graph we see that our validation set overfits after 120 epochs, so we chose no. of epochs (hyperparameter) as 120.
- Finally, we evaluate our network on test data and got below results.
- After evaluating on test data:
  - Loss Value: 2.9244151302412447
  - Mean Absolute error: 2.924415111541748

Mean Absolute Percentage Error:



- From the above graph we see that our validation set overfits after 100 epochs, so we chose no. of epochs (hyperparameter) as 100.
- Finally we evaluate our network on test data and got below results.
- After evaluating on test data:
    - Loss Value: 13.705399045757218
    - Mean Absolute error: 2.7445971965789795

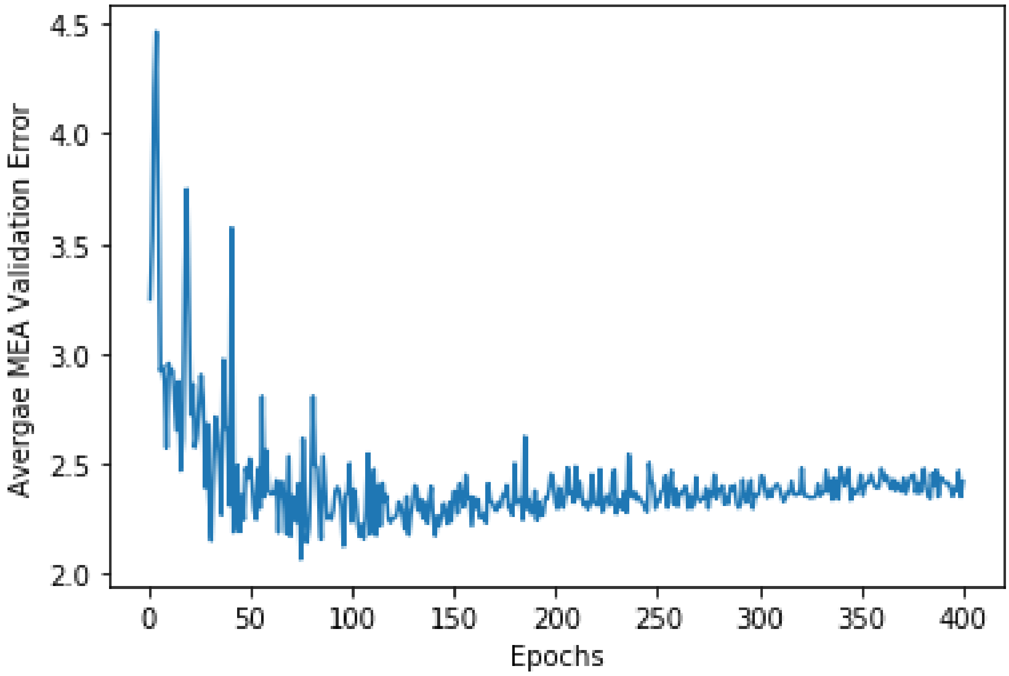
Mean Square Logarithmic Error:



- From the above graph we see that our validation set overfits after 200 epochs, so we chose no. of epochs (hyperparameter) as 200.
- Finally we evaluate our network on test data and got below results.
    - Loss Value: 0.03188016893816929
    - Mean Absolute error: 2.4127893447875977

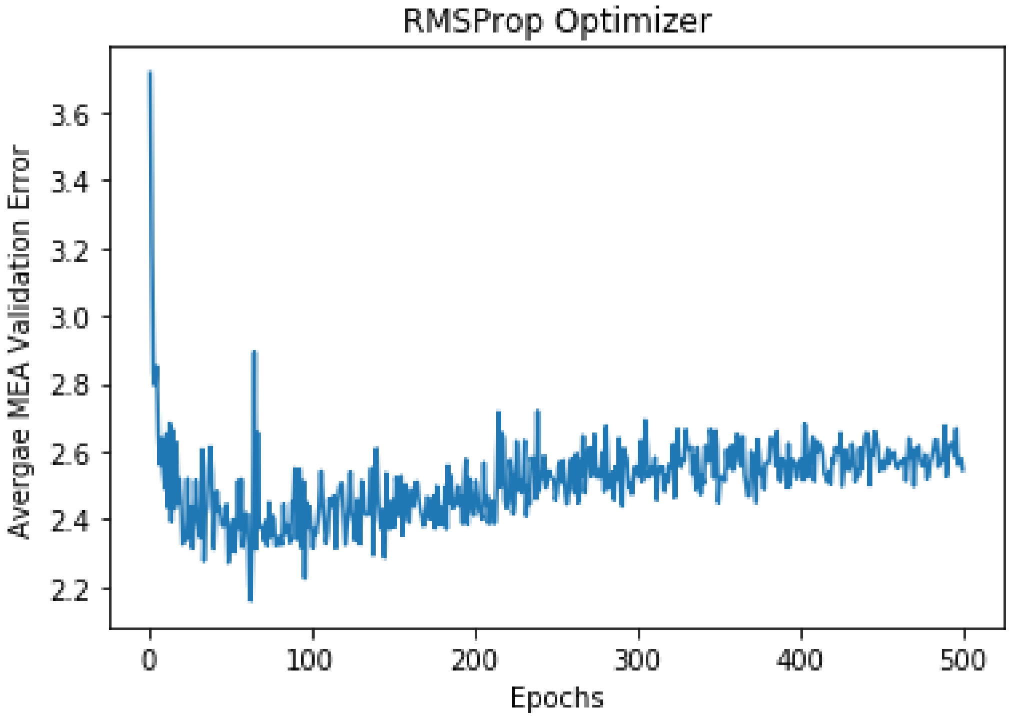
Evaluating Different Optimizers :

SGD Optimizers :



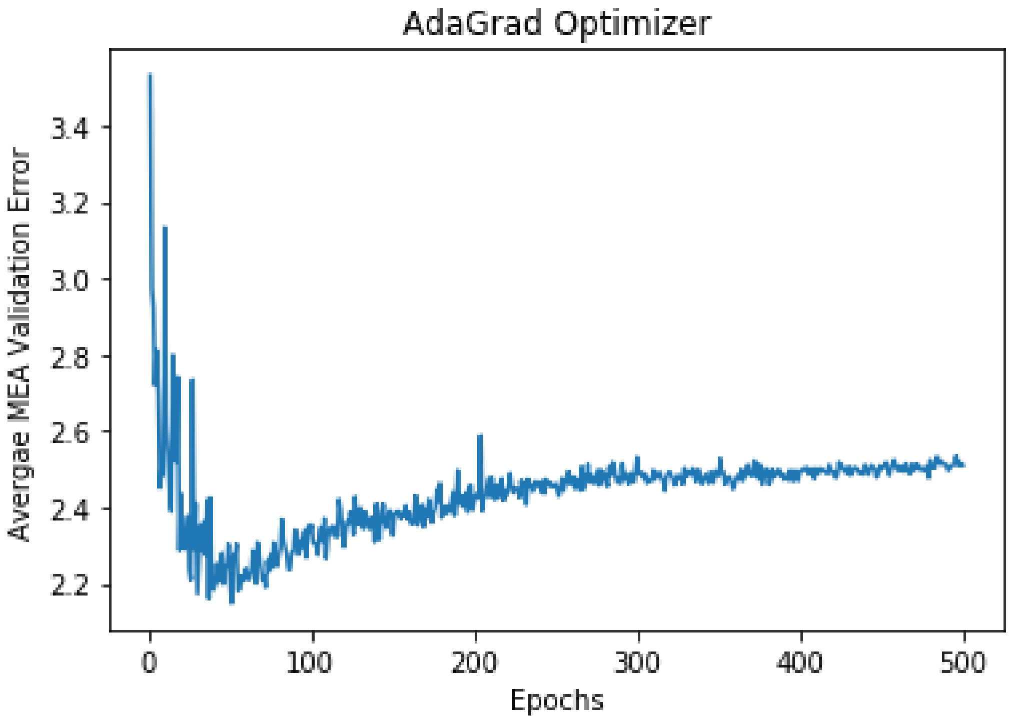
- From the above graph we see that our validation set overfits after 100 epochs, so we chose no. of epochs (hyperparameter) as 100.
- Finally we evaluate our network on test data and got below results.
  - MSE Loss Value: 13.357263004078584
  - Mean Absolute error: 2.5273001194000244

RMS Prop:



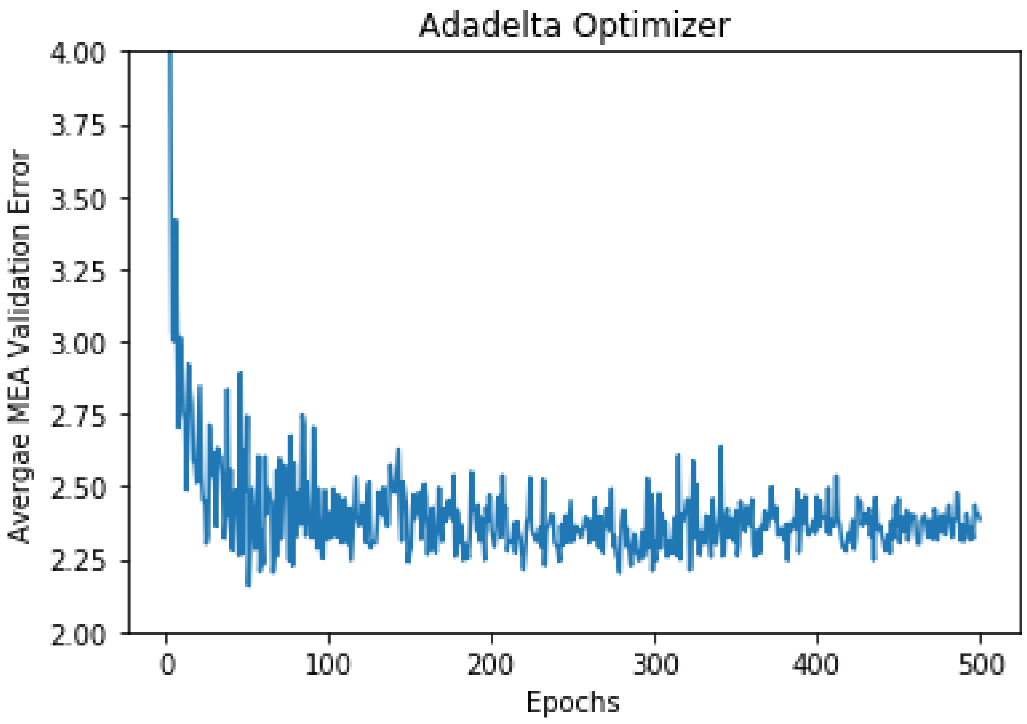
- From the above graph we see that our validation set overfits after 50 epochs, so we chose no. of epochs (hyperparameter) as 50.
- Finally we evaluate our network on test data and got below results.
    - MSE Loss Value: 12.951670927159926
    - Mean Absolute error: 2.5793259143829346

AdaGrad Optimizer :



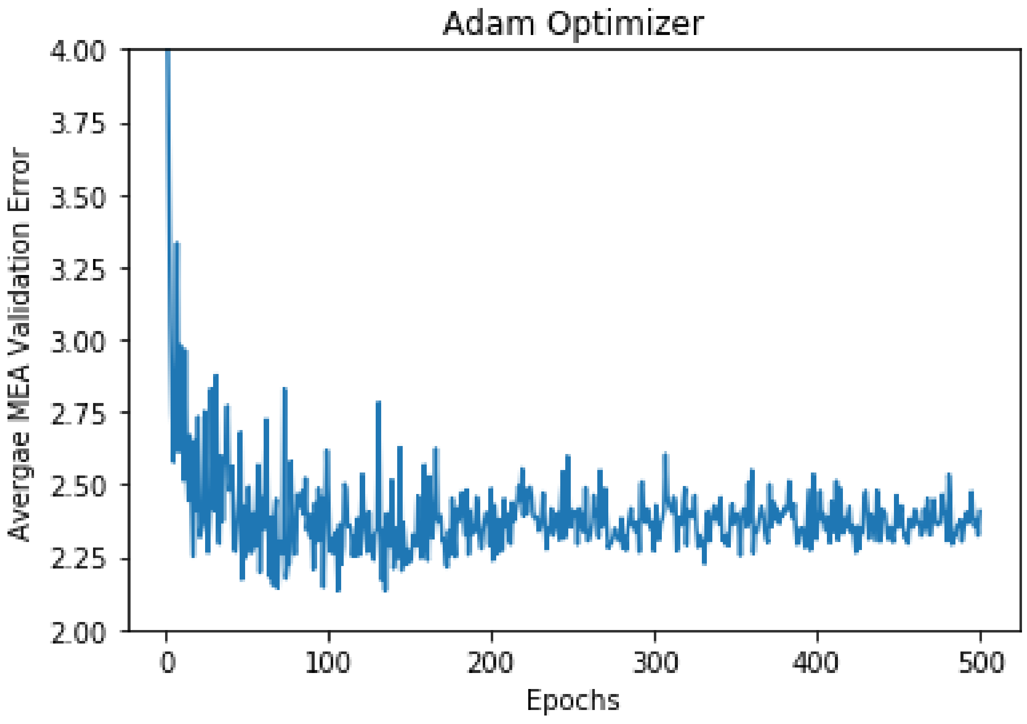
- From the above graph we see that our validation set overfits after 50 epochs, so we chose no. of epochs (hyperparameter) as 50.
- Finally we evaluate our network on test data and got below results.
    - MSE Loss Value: 14.262457604501762
    - Mean Absolute error: 2.4033048152923584

AdaDelta :



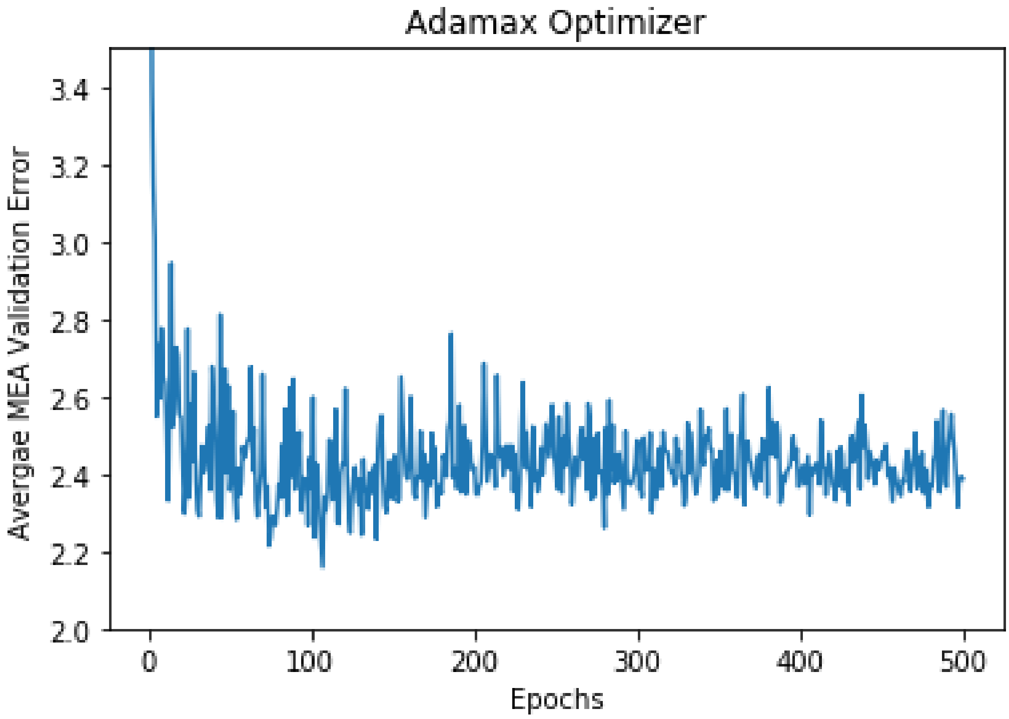
- From the above graph we see that our validation set overfits after 80 epochs, so we chose no. of epochs (hyperparameter) as 80.
- Finally we evaluate our network on test data and got below results.
  - MSE Loss Value: 14.830144321217256
  - Mean Absolute error: 2.80137038230896

Adam :



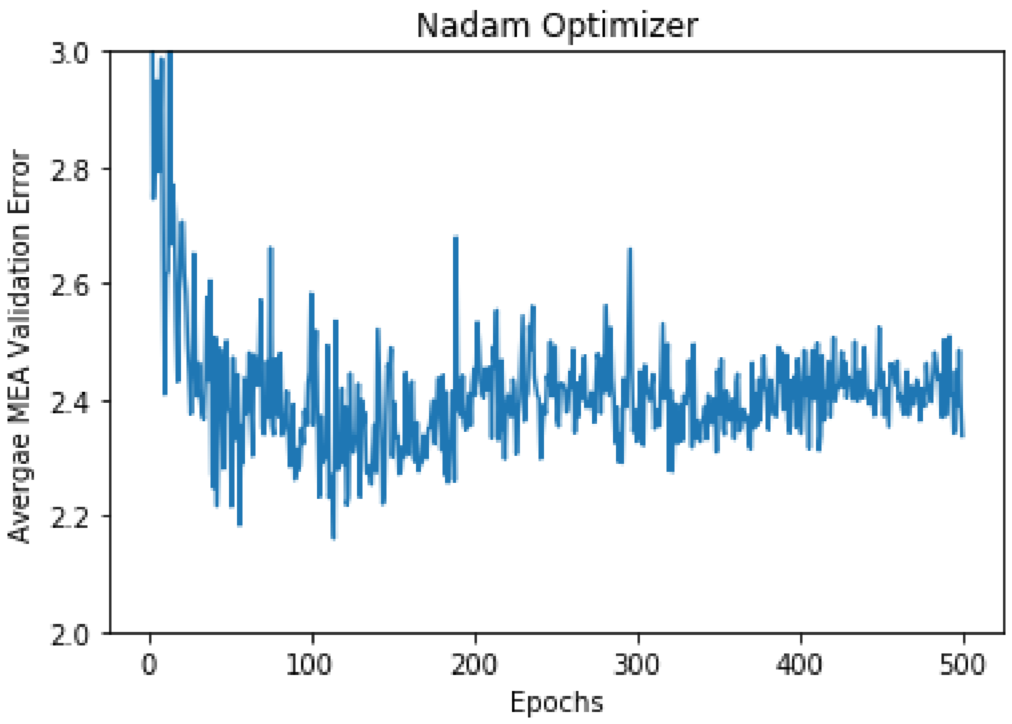
- From the above graph we see that our models validation loss starts to increase after 100 epochs and hence we choose our final number of epochs as 100.
- Finally we evaluate our network on test data and got below results.
  - MSE Loss Value: 15.179154713948568
  - Mean Absolute error: 2.706160068511963

AdaMax:



- From the above graph we see that our validation set overfits after 100 epochs, so we chose no. of epochs (hyperparameter) as 100.
- Finally we evaluate our network on test data and got below results.
  - MSE Loss Value: 15.128381317737055
  - Mean Absolute error: 2.538341760635376

Nadam :

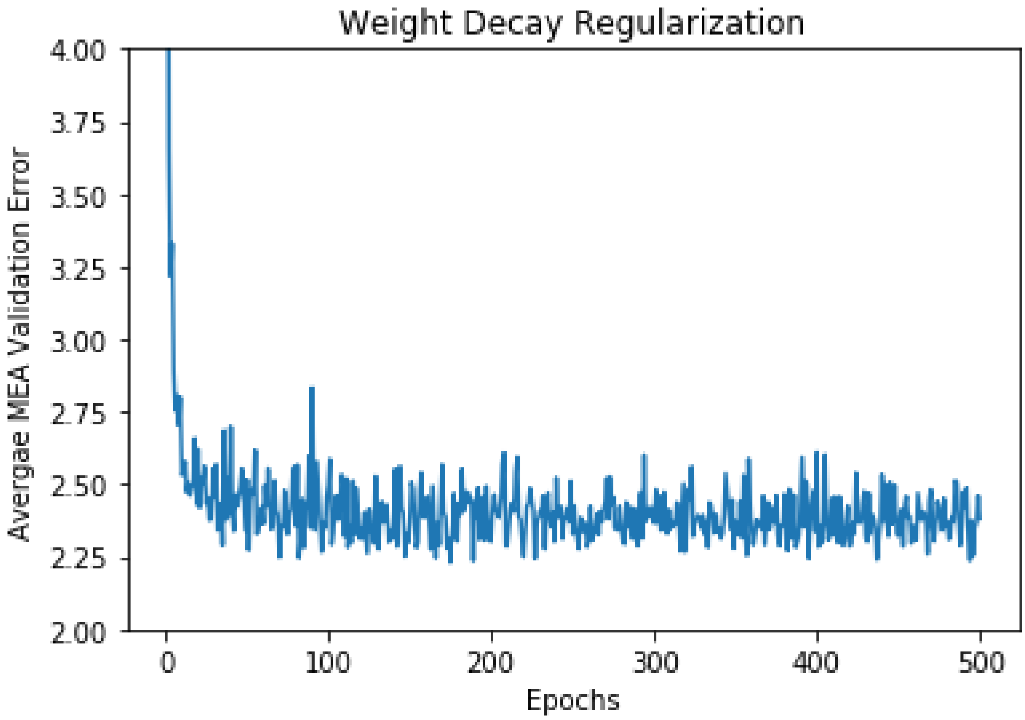


- From the above graph we see that our validation set overfits after 120 epochs, so we chose no. of epochs (hyperparameter) as 100.
- Finally we evaluate our network on test data and got below results.
  - MSE Loss Value: 9.845044491337795
  - Mean Absolute error: 2.290827989578247

By comparing all of the above Optimizers , we can see that Nadam Optimizers gives the best results (least mean squared error among all).

Evaluating Different Regularization measures :

Weight Decay with RMS Prop Optimizer model evaluated above:



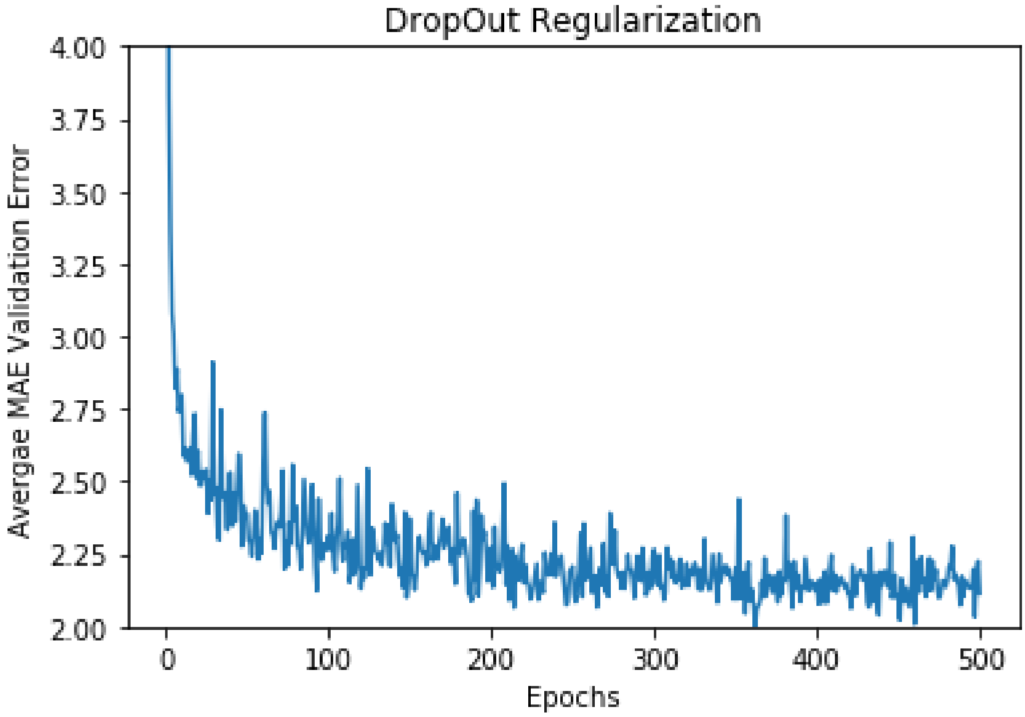
From the above graph we can see the using regularization our model generalizes well compared to RMS Prop Optimizer model without regularisation.

After evaluating on test data:

MSE Loss Value: 13.629344790589576

Mean Absolute error: 2.662123680114746

Drop Out with RMS Prop Optimizer model evaluated above:



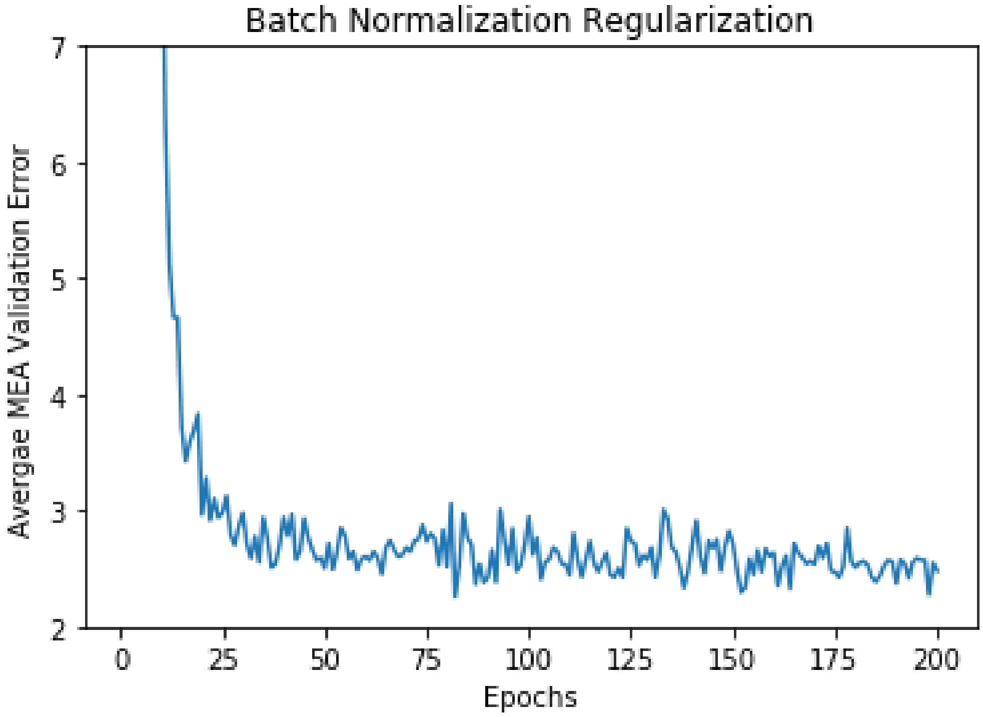
From the above graph we can see the using regularization our model generalizes well compared to RMS Prop Optimizer model without regularisation. We can see that validation error doesn’t increase after few epochs as compared to previous model which shows better generalisation.

After evaluating on test data:

MSE Loss Value: 11.140097468507056

Mean Absolute error: 2.312666893005371

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.

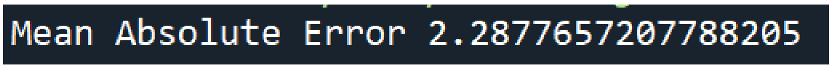
First we use Adam Classifier to predict values and then use RMSProp Classifier to predict values. Then we can take average of the difference between predicated values and true values to get mean absolute error of the ensemble classifier.

MSE Loss Value: 17.366640277937346

Mean Absolute error: 2.765009641647339

Mean Absolute Error 2.3894632068334842

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



```
Mean Absolute Error 2.2877657207788205
```