Digital VLSI Design Project

TEAM-SVMS

Sathya Sravya	20161121	Sidhant	20161051
Meghna	20161132	Sri Vyshnavi	20161235

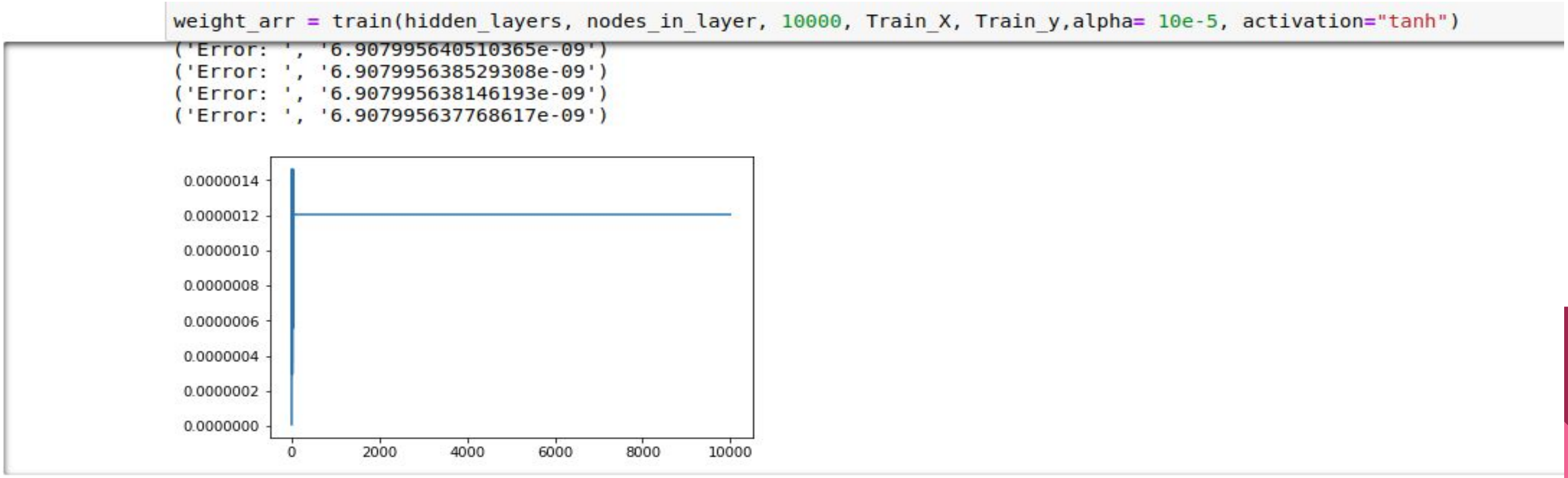
Aim

To train multiple models on the given Datasets and get accurate results

Neural Network without Deep Learning libraries

We have created a neural network using only numpy library

The loss is in the range 10^-9



```
weight_arr = train(hidden_layers, nodes_in_layer, 10000, Train_X, Train_y,alpha= 10e-5, activation="tanh")
```

```
('Error: ', '6.907995640510365e-09')
('Error: ', '6.907995638529308e-09')
('Error: ', '6.907995638146193e-09')
('Error: ', '6.907995637768617e-09')
```

Keras Implementation of Neural Network

This is a sequential model that is made up of 3 layers

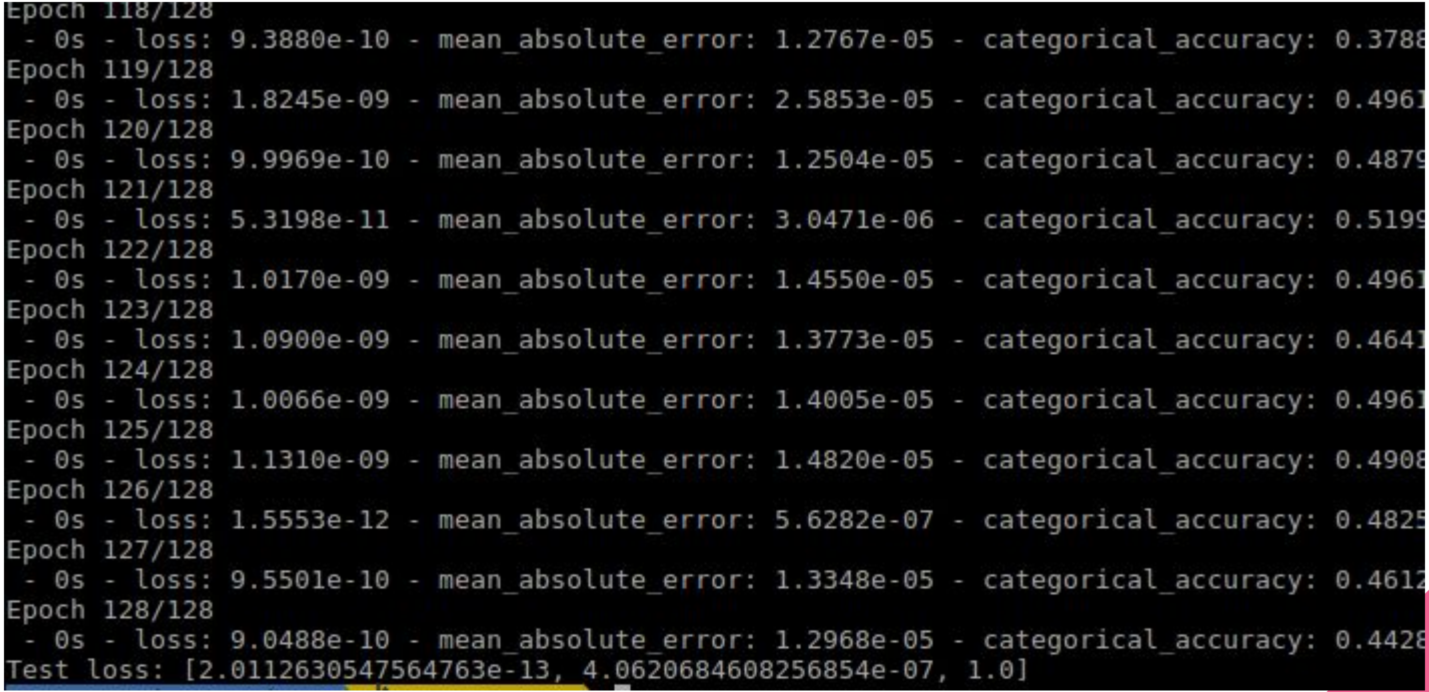
The Activation Function used in the Neural - Network is Tanh. Usage of other Activation functions causes the loss to go to Positive infinity

Ran 128 epochs

Loss is mean_squared_error

Optimizer is Adamax

Keras Implementation of Neural Network



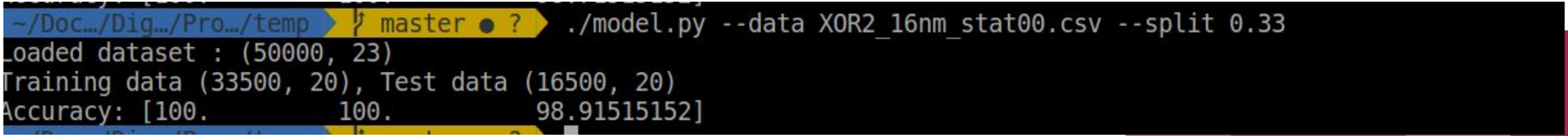
```
Epoch 118/128
 - 0s - loss: 9.3880e-10 - mean_absolute_error: 1.2767e-05 - categorical_accuracy: 0.3788
Epoch 119/128
 - 0s - loss: 1.8245e-09 - mean_absolute_error: 2.5853e-05 - categorical_accuracy: 0.4961
Epoch 120/128
 - 0s - loss: 9.9969e-10 - mean_absolute_error: 1.2504e-05 - categorical_accuracy: 0.4879
Epoch 121/128
 - 0s - loss: 5.3198e-11 - mean_absolute_error: 3.0471e-06 - categorical_accuracy: 0.5199
Epoch 122/128
 - 0s - loss: 1.0170e-09 - mean_absolute_error: 1.4550e-05 - categorical_accuracy: 0.4961
Epoch 123/128
 - 0s - loss: 1.0900e-09 - mean_absolute_error: 1.3773e-05 - categorical_accuracy: 0.4641
Epoch 124/128
 - 0s - loss: 1.0066e-09 - mean_absolute_error: 1.4005e-05 - categorical_accuracy: 0.4961
Epoch 125/128
 - 0s - loss: 1.1310e-09 - mean_absolute_error: 1.4820e-05 - categorical_accuracy: 0.4908
Epoch 126/128
 - 0s - loss: 1.5553e-12 - mean_absolute_error: 5.6282e-07 - categorical_accuracy: 0.4825
Epoch 127/128
 - 0s - loss: 9.5501e-10 - mean_absolute_error: 1.3348e-05 - categorical_accuracy: 0.4612
Epoch 128/128
 - 0s - loss: 9.0488e-10 - mean_absolute_error: 1.2968e-05 - categorical_accuracy: 0.4428
Test loss: [2.0112630547564763e-13, 4.0620684608256854e-07, 1.0]
```

Scikit Implementation of models

In the First model we used **Linear Regression**

This had the maximum accuracy of 100% 100% 98%

Considering the data’s sparsity it is intuitive the classic Linear Regression will work the best

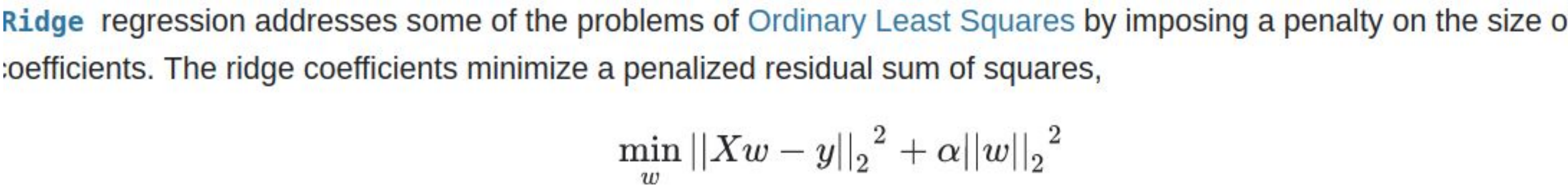


```
~/Doc…/Dig…/Pro…/temp  master ● ?  ./model.py --data XOR2_16nm_stat00.csv --split 0.33
Loaded dataset : (50000, 23)
Training data (33500, 20), Test data (16500, 20)
Accuracy: [100.         100.         98.91515152]
```

Scikit Implementation of models

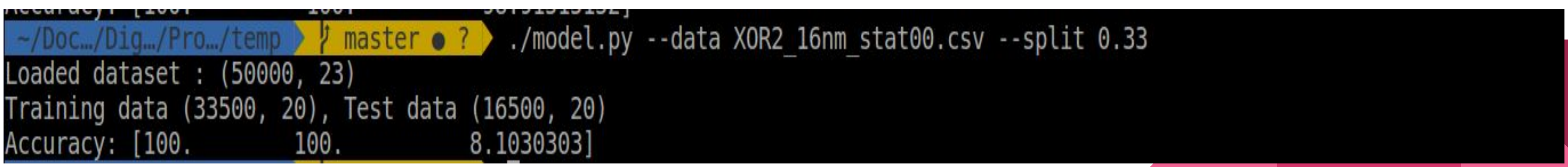
Ridge Regression

Since the data is sparse we could not get good results on Leakage



Ridge regression addresses some of the problems of Ordinary Least Squares by imposing a penalty on the size o
:oefficients. The ridge coefficients minimize a penalized residual sum of squares,

$$\min_{w} ||Xw - y||_2{}^2 + \alpha ||w||_2{}^2$$

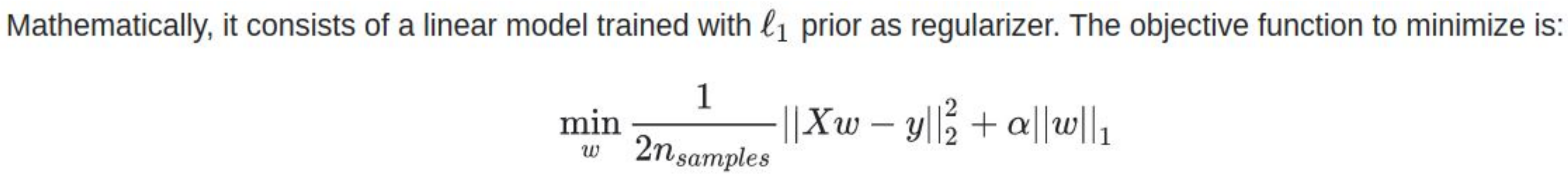


```
~/Doc…/Dig…/Pro…/temp  master ● ?  ./model.py --data XOR2_16nm_stat00.csv --split 0.33
Loaded dataset : (50000, 23)
Training data (33500, 20), Test data (16500, 20)
Accuracy: [100.        100.         8.1030303]
```

Scikit Implementation of models

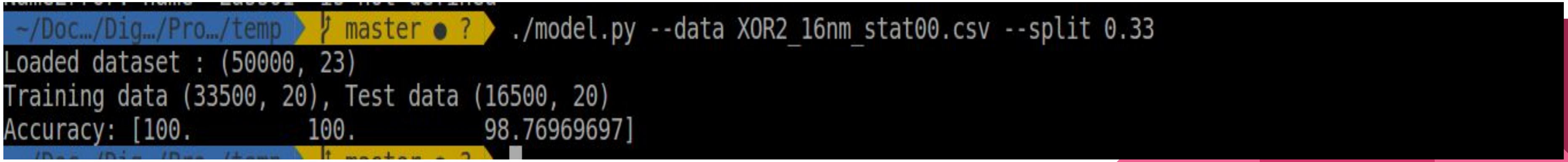
Lasso Regression

This model also had high accuracy



Mathematically, it consists of a linear model trained with ℓ_1 prior as regularizer. The objective function to minimize is:

$$\min_{w} \frac{1}{2n_{samples}} ||Xw - y||_2^2 + \alpha ||w||_1$$



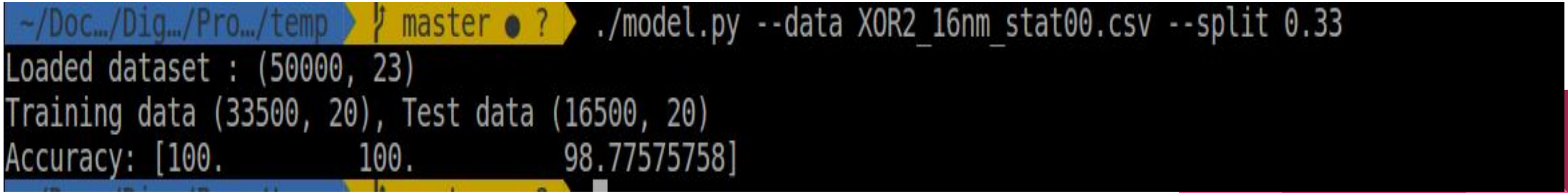
```
NameError: name 'Lasso' is not defined
 ~/Doc…/Dig…/Pro…/temp  master ● ?  ./model.py --data XOR2_16nm_stat00.csv --split 0.33
Loaded dataset : (50000, 23)
Training data (33500, 20), Test data (16500, 20)
Accuracy: [100.         100.          98.76969697]
 ~/Doc…/Dig…/Pro…/temp  master ● ? 
```

Scikit Implementation of models

LassoLars

is piecewise linear as a function of the norm of its coefficients.

The Results are pretty good on this model as well



```
~/Doc…/Dig…/Pro…/temp  master ● ?  ./model.py --data XOR2_16nm_stat00.csv --split 0.33
Loaded dataset : (50000, 23)
Training data (33500, 20), Test data (16500, 20)
Accuracy: [100.        100.         98.77575758]
```

Neural Network with dropout

Including dropout layers,to reduce overfitting,make the system work on any dataset given,this implementation is done, results were obtained.

Neural Network with SGD optimizer

Using Stochastic gradient descent optimiser and mentioned , along with gradient clipping and momentum inclusion, results were obtained.

Conclusion

- With an average accuracy of around 99.xx% we have implemented hspice like models using machine learning algorithms
- The objective or the criteria of error being less than 1% is attained.