**IMPLEMENTATION REPORT OF PROGRAMMING CHALLANGE**

**Libraries used:-**

* Numpy
* Sklearn
* Pandas
* Matplotlib

**Project Link:-**[**https://github.com/VikasOjha666/AdaActNN**](https://github.com/VikasOjha666/AdaActNN)

**Implementation details:-**

I have conducted the experiment on Iris dataset which is in csv file format. The whole neural network i.e. both the forward pass and the backward pass has been implemented with numpy. The pandas has been used for loading the data. Various utilities function of sklearn has been used for splitting, one hot encoding and calculating various metrics like accuracy, precision,etc.

I choose the Xavier’s method for initializing the weights which samples from a normal distribution from 0 to standard deviation given by where fanin is input dimension and fanout is output dimension of the weight array. The K vector was initialized with random small numbers.

As per the paper different function like g(x) termed as g\_function was implemented to act as our Ada act function.

Sotmax equation was implemented.

The forward and backward functions were implemented as per the description in the paper.Various functions of numpy were used for this. The np.dot() function was used for matrix multiplication,np.average() was used for taking column average where axis=0 means along the column axis. The function np.mean was used for calculating the average over the whole matrix.

After the implementation of neural network is done.The data is loaded using pandas and the fisrt 100 samples from the iris dataset is selected as it contains two classes upto that point.After the 100 sample the dataset contains the data of third class.As we are working on two class setting we are interested in data much data only.

After that the label from string is changed to 0 and 1 respectively. And the data is split into training and test sets. The training set contains 75% of the data while test set contains 25%.

In next step we use the Sklearn StandardScaler class to perform the normalization of the data.

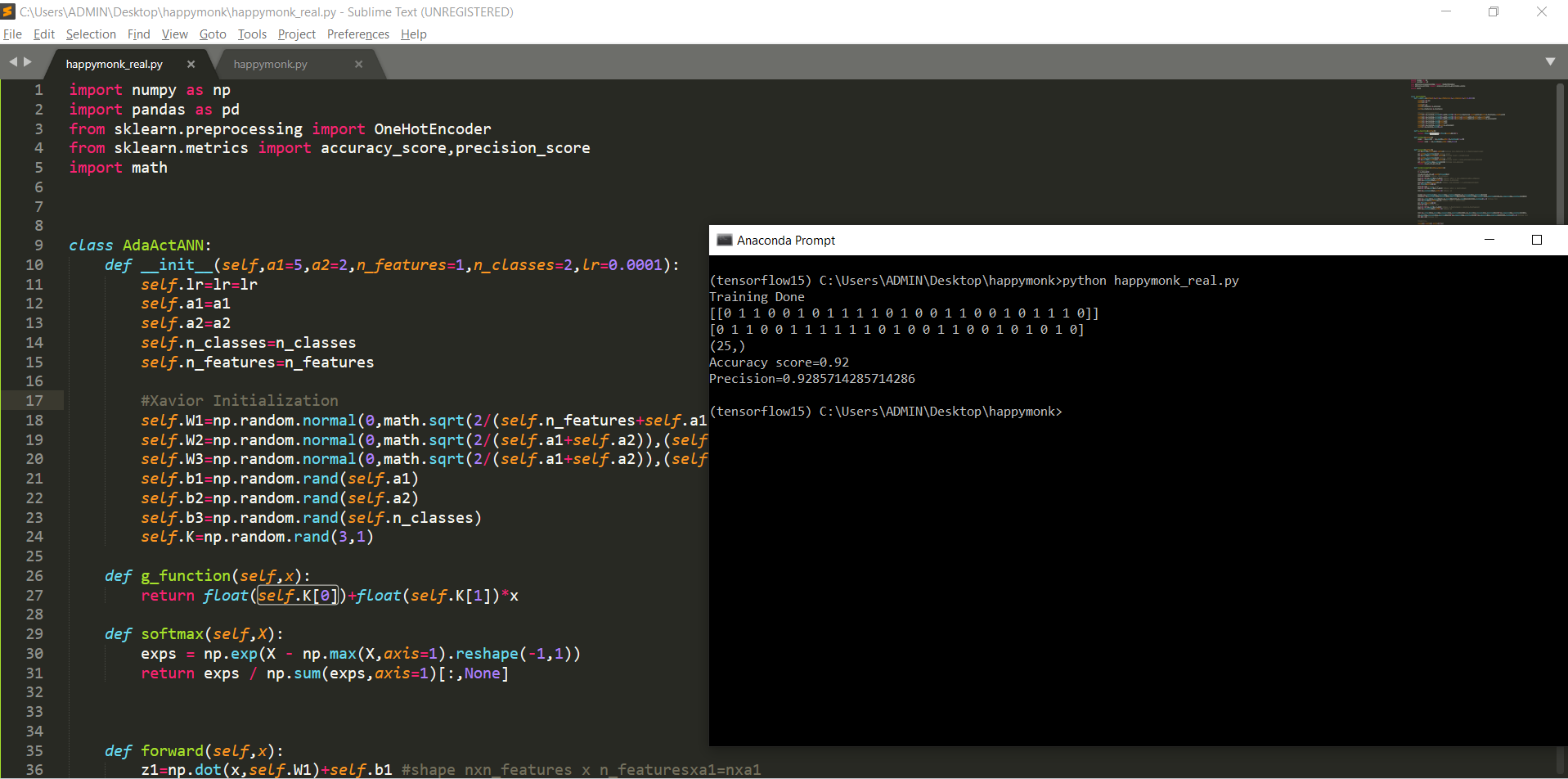
After performing the normalization on the data, the labels are one hot encoded.

After that we create the instance of the above defined Network class and is trained using a for loop for 1000 epochs.

After the training the prediction is stored in a array and then we use sklearn.metrics package to calculate the accuracy and precision.

The implemented network not always gives the optimum result due to the stochastic nature of algorithm. There is more room for improvement in code as well.

But in the given timeframe this is one of the best result I have got:-



Here the accuracy is around 0.92 and precision is around 0.9285 on test data which is not bad.

But this result may vary across various runs.

The Epochs vs loss plots over 1000 epochs:-

