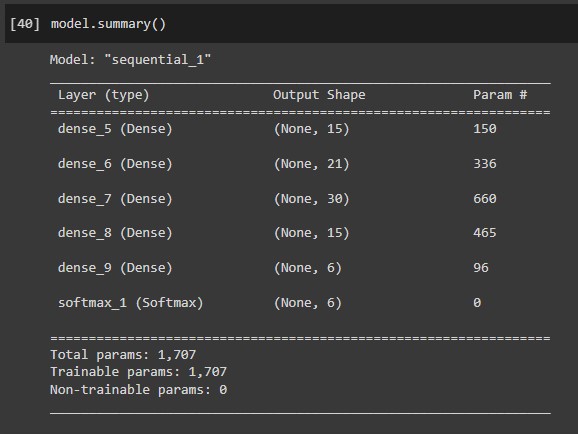
**Report Homework 2**

**A diagram of your network and the parameters used (such as size, no. of neurons, learning rate, etc.)**



Input layer: 6

1st Hidden layer: 15

2nd Hidden layer: 21

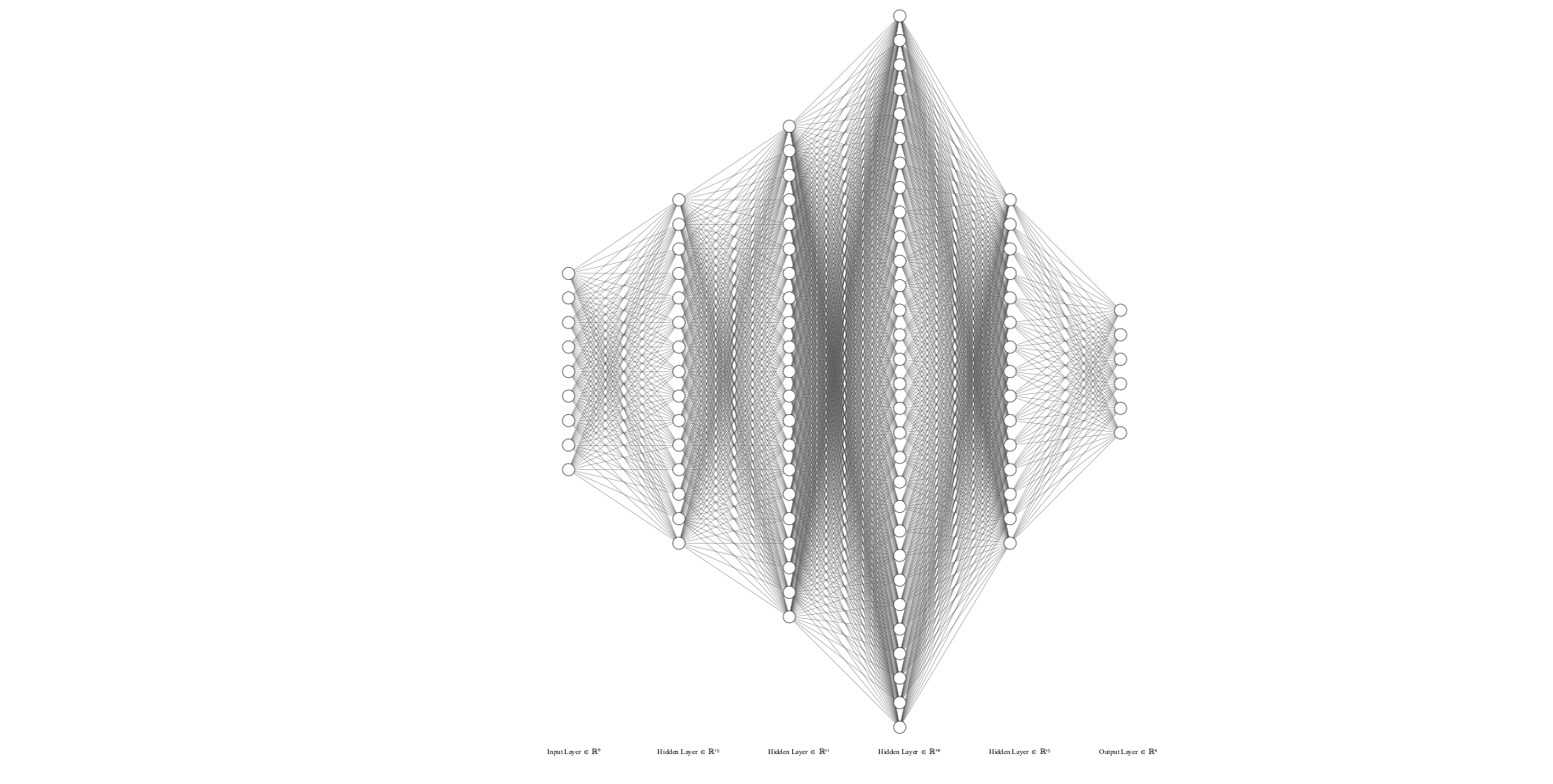
3rd Hidden layer: 30

4th Hidden layer: 15

5th Hidden layer: 6

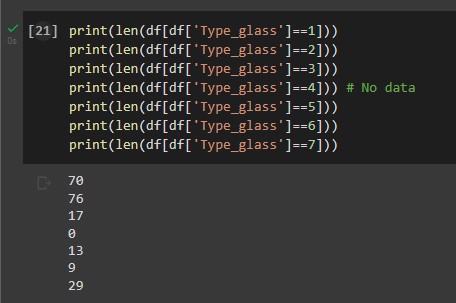
Then, Softmax function.

The neural network diagram next page is generated using <https://alexlenail.me/NN-SVG/>

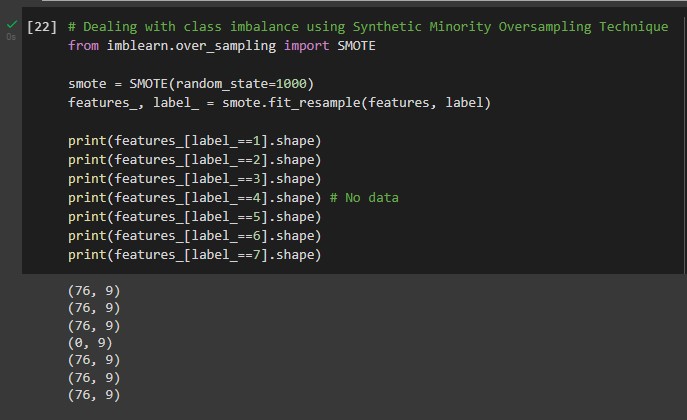


**Explanation how you normalize the data and perform class balancing (by code or by hand)**

A class balancing should be performed before normalizing data.

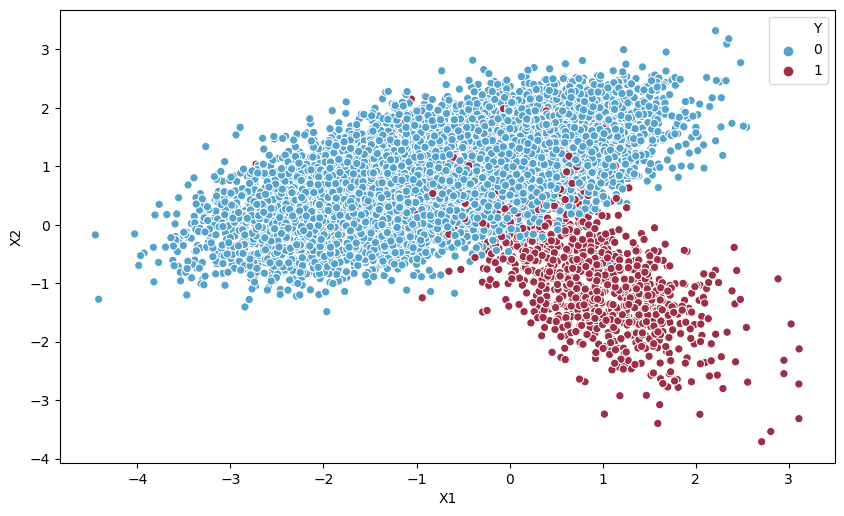


First, I investigated on how many samples in each type of label. It is clear that there are class imbalance, the first and second label are a lot more than the rest (there was no data for label 4).

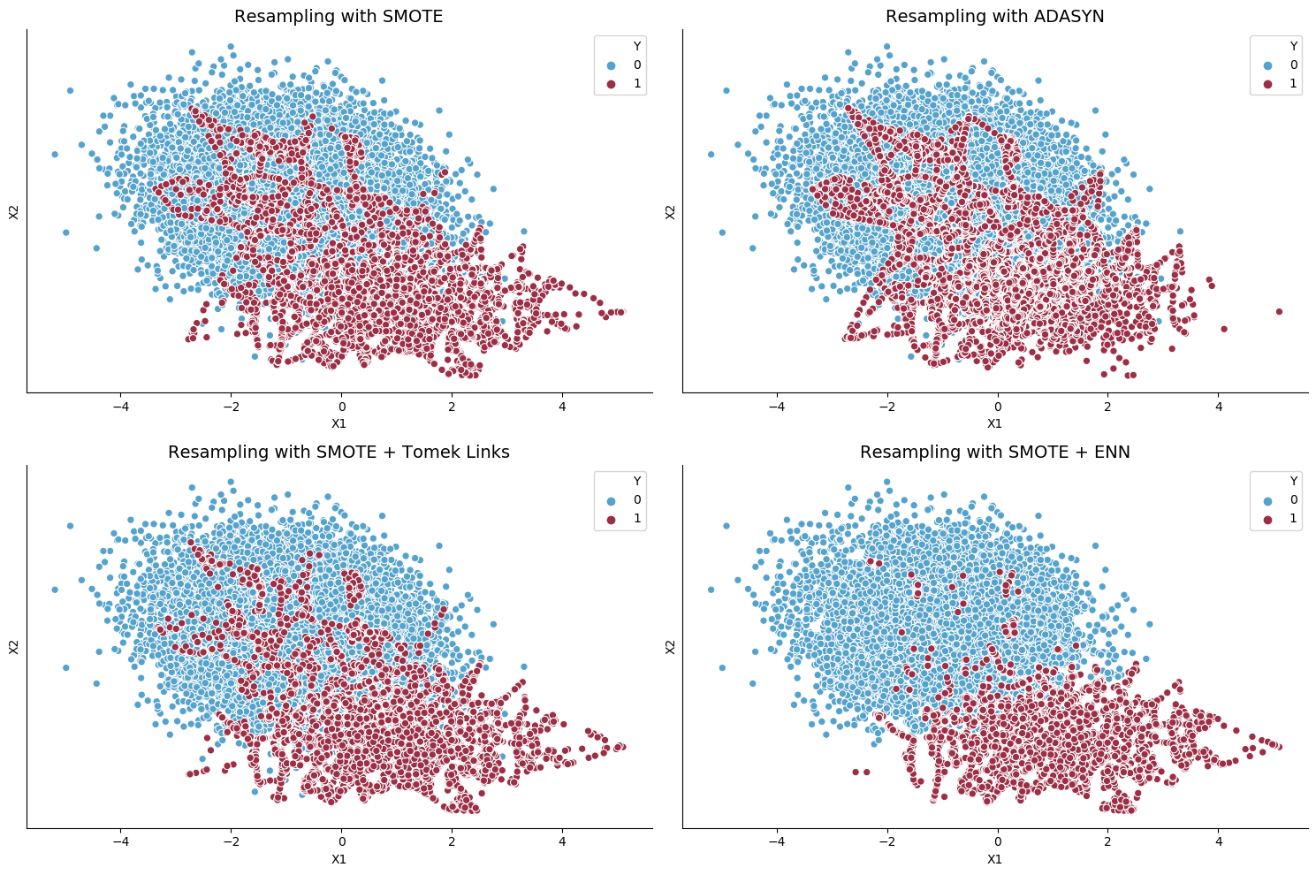


Working with imbalanced datasets presents a challenge because most machine learning techniques ignore the minority class, which results in poor performance even though this performance is typically the most crucial.

Oversampling the minority class is one way to deal with unbalanced datasets. Duplicating examples from the minority class is the simplest method, but these examples don't provide any new insight into the model. Instead, new examples can be created by synthesizing the existing ones. The Synthetic Minority Oversampling Technique, or SMOTE for short, is a type of data augmentation for the minority class.

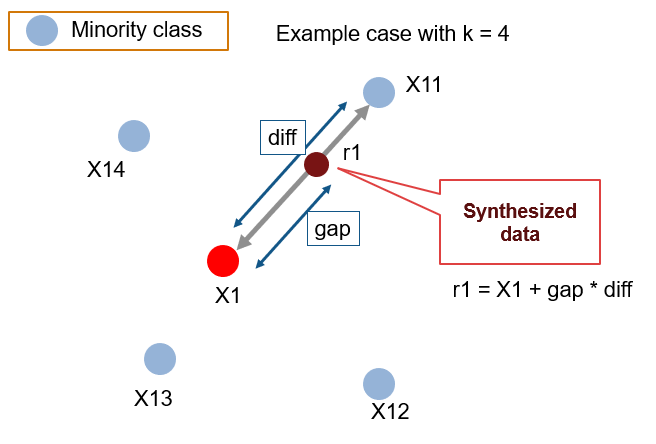


Example of majority (0) and minority (1) class

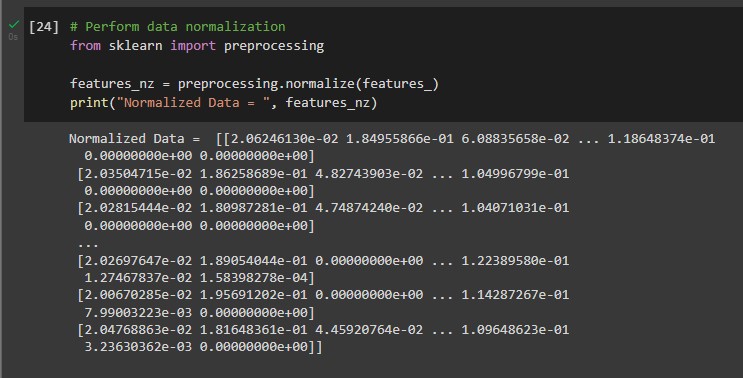


After applying SMOTE resampling

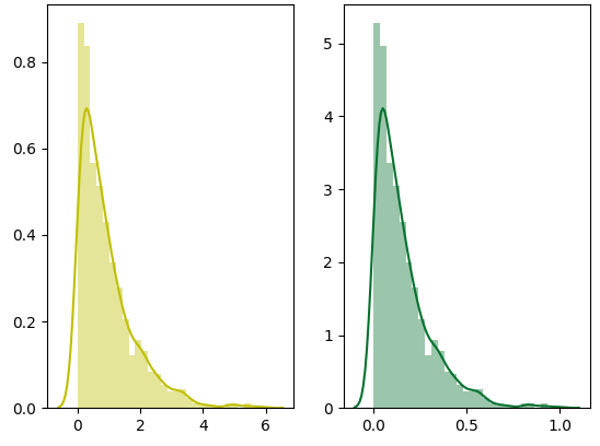
The total number of oversampling observations, N, is first established. The binary class distribution is typically chosen to be 1:1. But, if necessary, that could be turned down. The iteration then begins by first picking a random member of a positive class. The KNNs for that instance are then obtained (there are 5 by default). Finally, N is picked from among these K instances to interpolate new synthetic instances. To do that, the distance between the feature vector and its neighbors is calculated using any distance metric. The previous feature vector is now multiplied by this difference plus any random value within the range of [0, 1]. Below is a visual representation of this:



Thus, I applied SMOTE resample on the features and labels, it will resample samples for each label to remove the class imbalance problem.

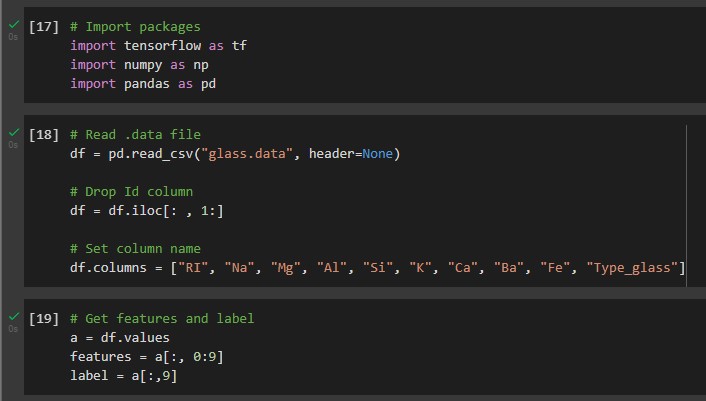


After that, I used Scikit-learn package’s command to apply normalization to the balanced features. Machine learning uses data normalization to reduce the sensitivity of model training to feature scale. This enables our model to converge to better weights, which results in a model that is more accurate.



Example, Left: Original data, Right: Normalized data

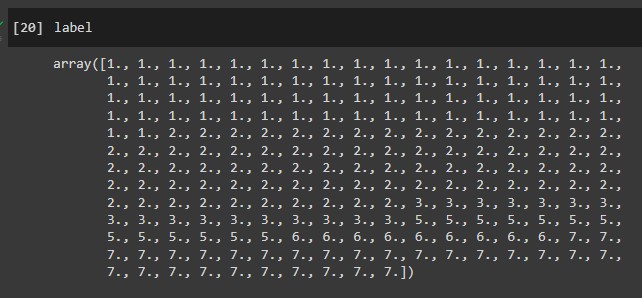
**Printed the source code and a short explanation of each part**



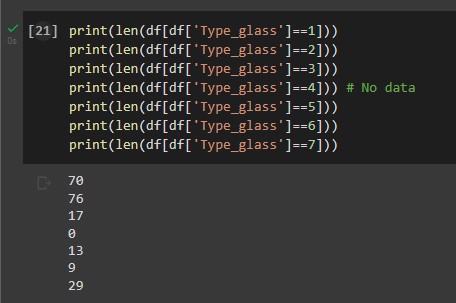
First, I imported the main packages, including TensorFlow, to work on neural networks. NumPy, to work on arrays, and Pandas to work on Data Frames.

Then, I read the .data file, that is already in the CSV format, dropped the Id column because it is not necessary for the neural network training. Then give column name for each column.

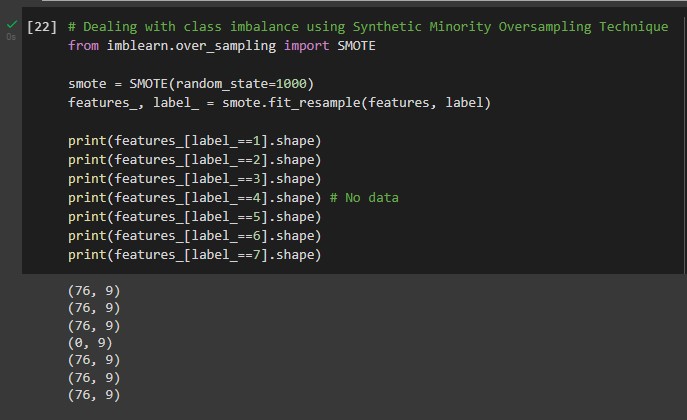
After that, I apply code to get features and label, first 9 columns are the features (0:9 = column 0-8), and the last column is label (type of glass).



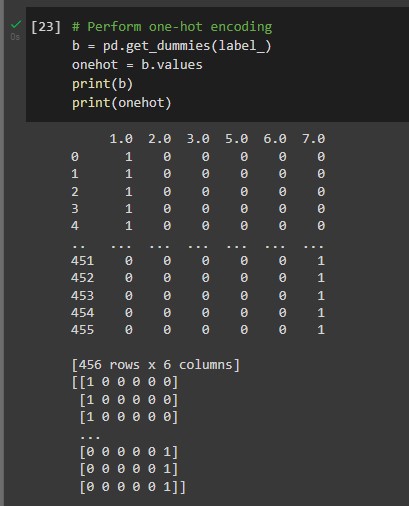
There is no sample for label 4, as the instruction said.



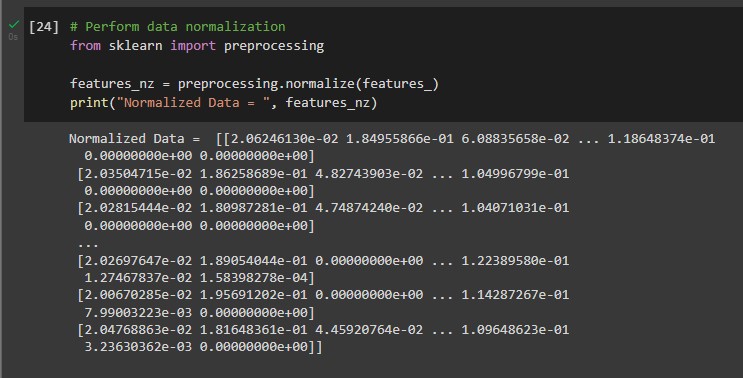
I check how many samples for each label by printing out len of df according to the label number. You can see that class balancing need to be performed.



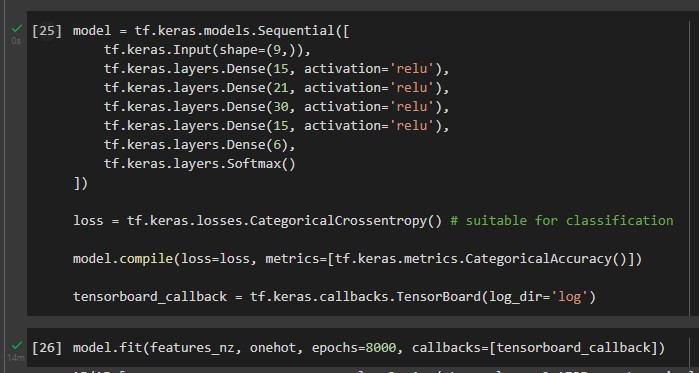
After that, I performed class balancing using SMOTE techniques, as mentioned earlier.



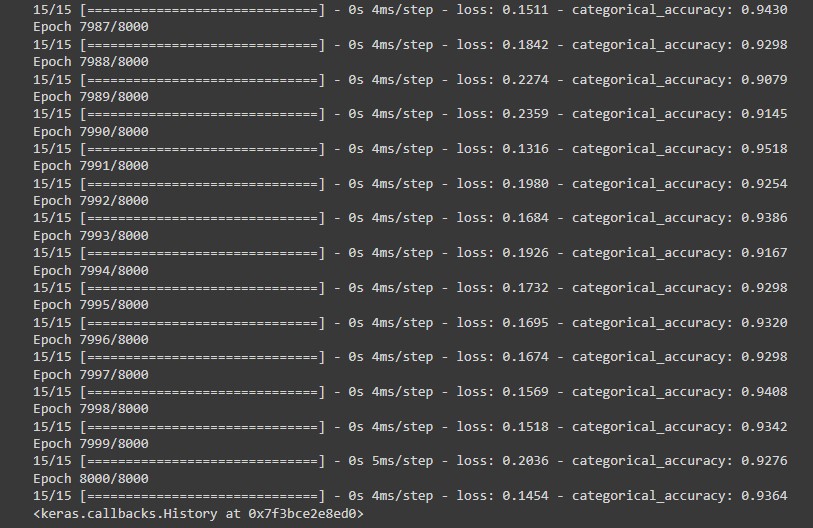
After we get the balanced features and labels, apply Pandas’ command to perform one-hot encoding on new label.



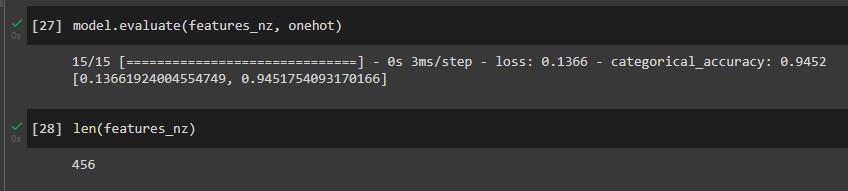
After that, apply normalization using scikit-learn command as mentioned earlier.



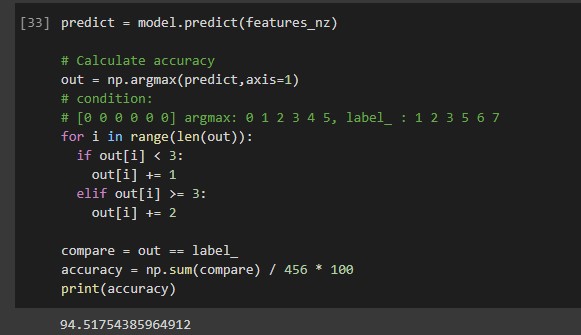
Now it’s time to build neural networks. I specified the hidden layers as shown. There are 9 features, so the input shape is 9. I use ReLU activation function for every hidden layer, except for the last one. Then apply the Softmax function. The loss function is Categorical Cross Entropy, it is the one suitable for classification problem. For accuracy metric, I use Categorical Accuracy, it calculates how often predictions match one-hot labels. Then, fit the model with 8000 epochs.



Wait until it finishes training procedure.



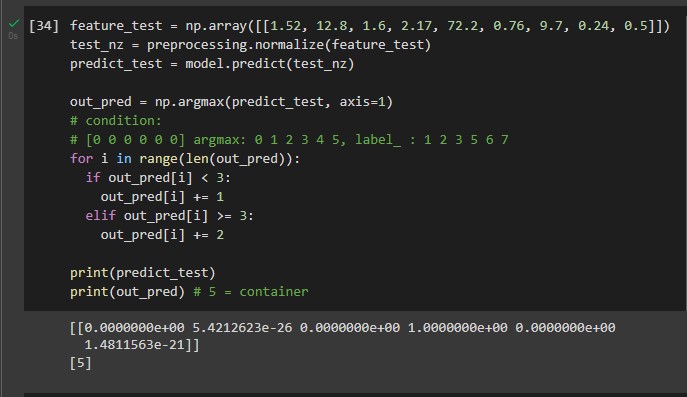
Evaluate model loss and accuracy. Also, I use command len(features\_nz) to use for accuracy calculation further.



Now, try to predict on training set. The problems are

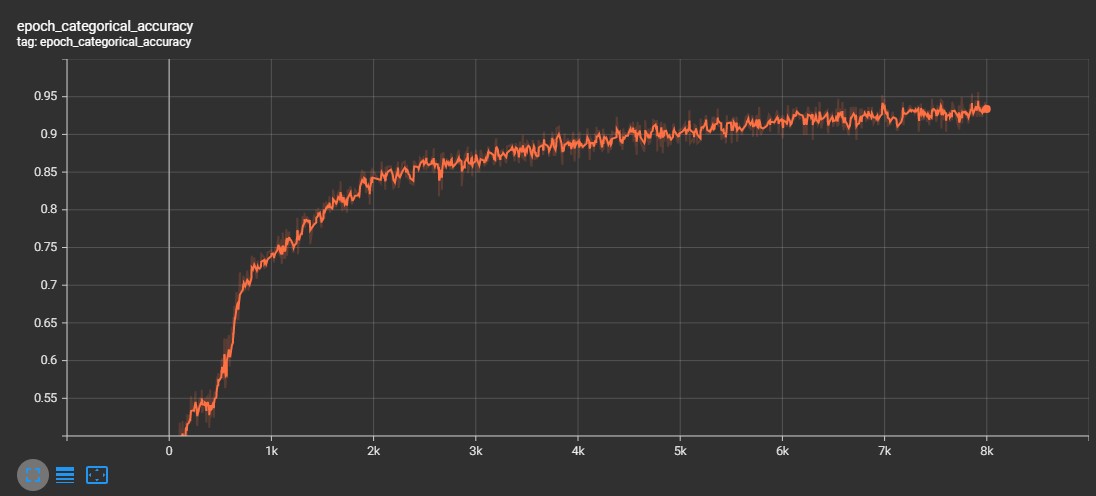
1. Label did not start with 0
2. There was no ‘4’ in the label

But the ‘out’ still treats as [0 0 0 0 0 1] for example. So, I manually rearrange the values output. For instance, if the ‘out’ output is ‘5’, it means that, the prediction is something like [0 0.01 0 0 0 0.99], the highest probability is 0.99, so if applied np.argmax command, it will return ‘5’ (since its 0-5). But the ‘5’ returned doesn’t mean it is the label ‘5’. In fact, it is labeled ‘7’. Thus, if out is less than 3 (0, 1, 2), I add 1 to it (becomes 1 2 3). But if its more than or equal to 3 (3, 4, 5), I add 2 to it (becomes 5, 6, 7). After that, we compare the true/false with the training data, sum the true (1), divide it with the number of dataset (456) and multiply by 100, we get the accuracy number (Nearly the same as using Keras’ metrics).

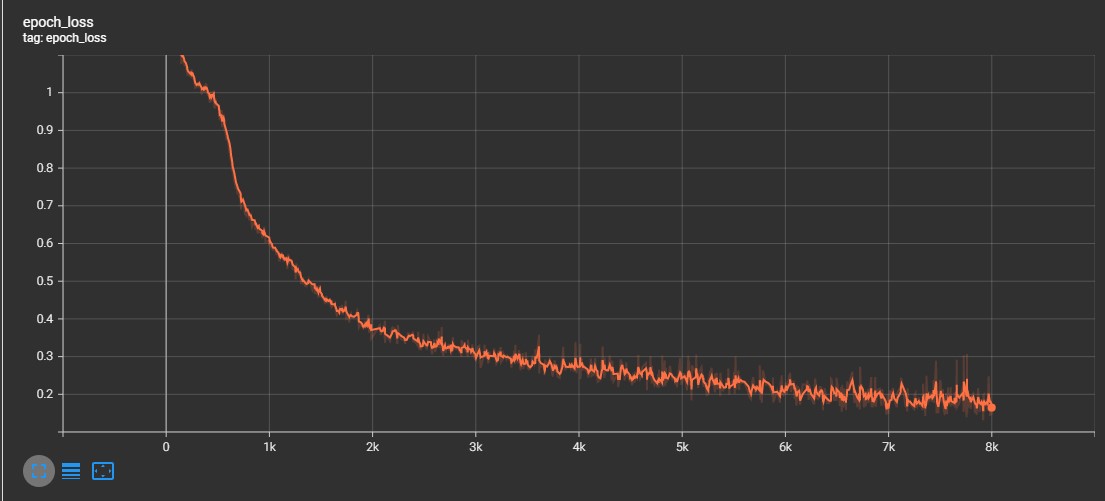


Then, it’s time to test the given features (1.52, 12.8, 1.6, 2.17, 72.2, 0.76, 9.7, 0.24, 0.5). Set it as NumPy array. But we have to normalize it first, since our training dataset is also normalized. Then, do the same procedure and print out the ‘out\_pred’ to know which one has the highest probability (argmax). It seems that it shows ‘5’ (5 = container). If we print out predict\_test, it will show the softmax function probability. You can see that label ‘5’ has the highest probability.

**Graph Results (2 graphs)**



The training data categorical accuracy graph (8000 epoch)



The training data loss graph (8000 epoch)

**Answer which types of glass is the sample and show the probability value**

Text

Description automatically generated

It seems that it shows ‘5’ (5 = containers). If we print out predict\_test, it will show the softmax function probability. You can see that label ‘5’ has the highest probability. The probability value is nearly 1.

-- 1 building\_windows\_float\_processed -- 2 building\_windows\_non\_float\_processed

-- 3 vehicle\_windows\_float\_processed

-- 4 vehicle\_windows\_non\_float\_processed (none in this database)

-- 5 containers

-- 6 tableware -- 7 headlamps