March 20, 2018

**Lab Assignment 4**

Sanchit Singhal

INF 385T – Introduction to Machine Learning with Danna Gurari

Spring 2018

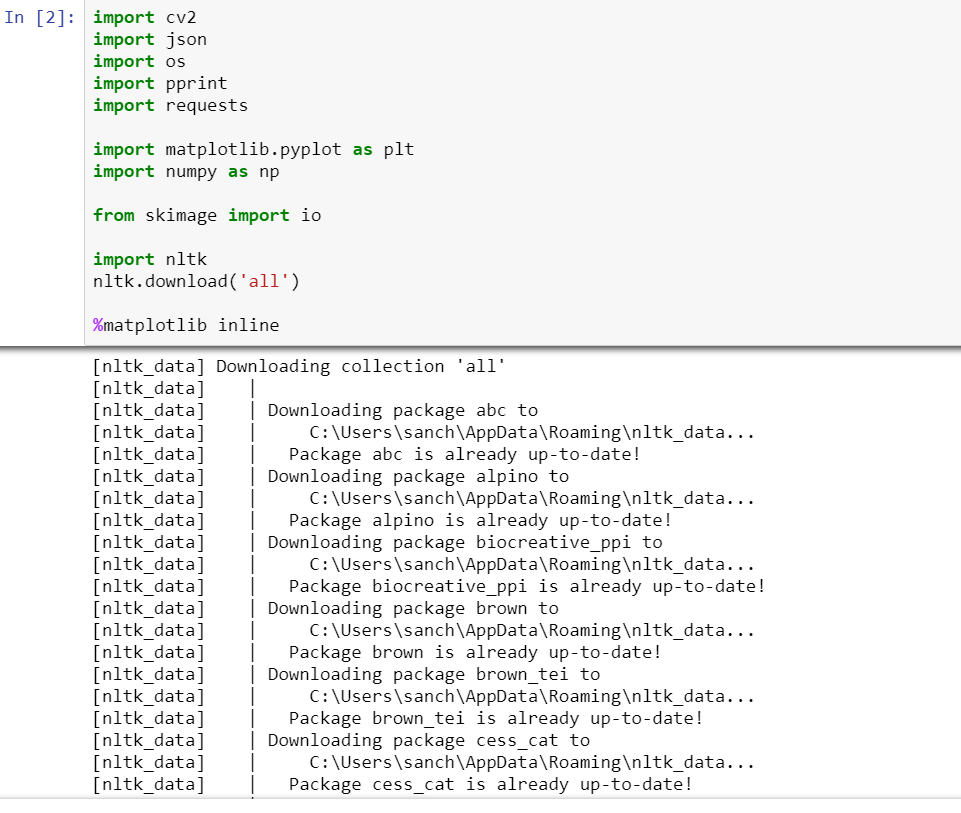
School of Information

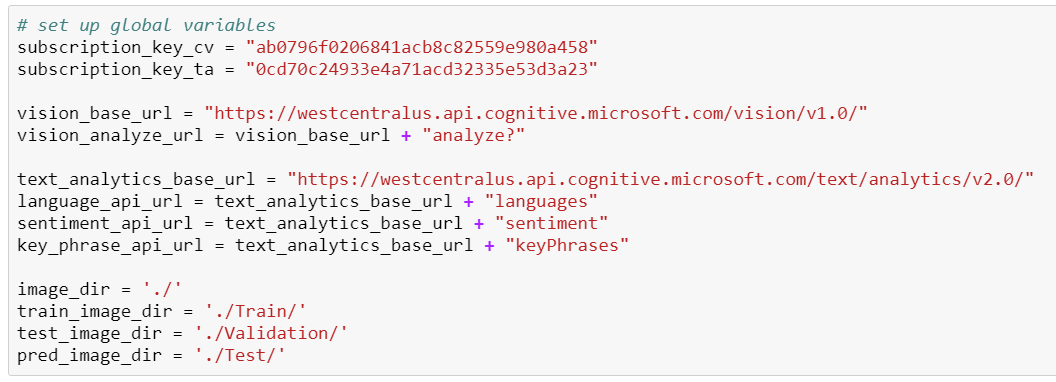
**Hyperlink to code:** [**https://introtoml-sanchit1276.notebooks.azure.com/nb/notebooks/IntroToML/LabAssignment4.ipynb**](https://introtoml-sanchit1276.notebooks.azure.com/nb/notebooks/IntroToML/LabAssignment4.ipynb)

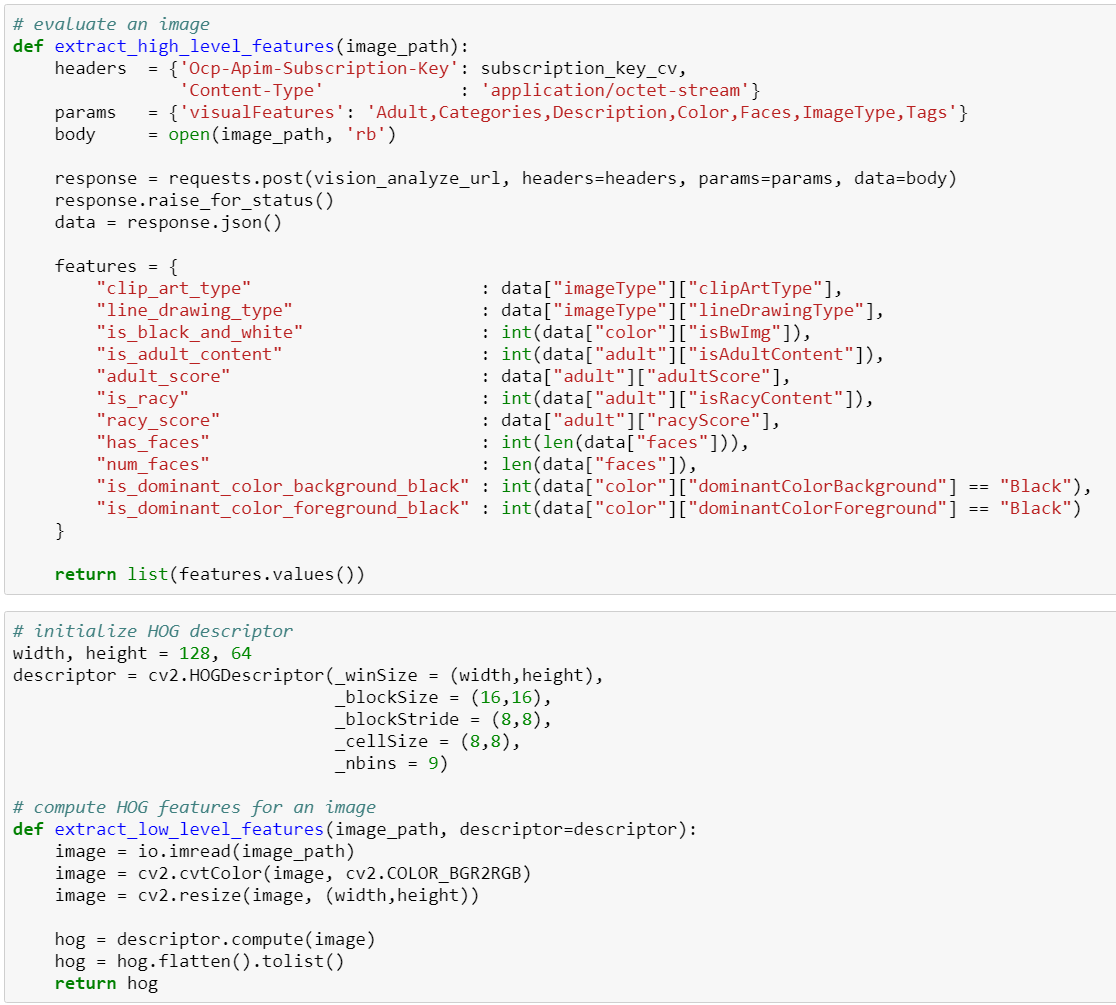
1. **Classification Using Hand-Crafted Features**
2. **Downloaded dataset**

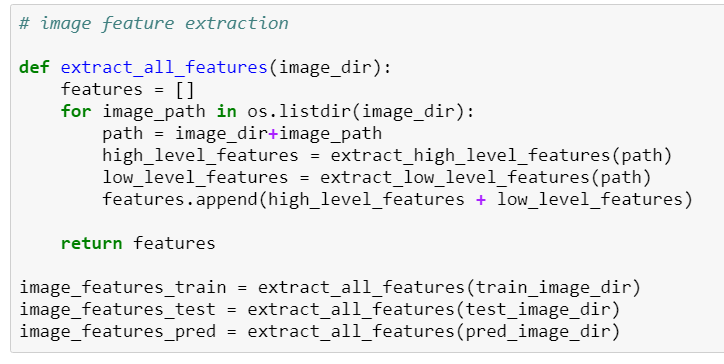
I am only using 101 training samples and 31 validation samples because Jupyter Notebooks is timing out with a data rate error when trying to load the full dataset.

1. **Feature Extraction**

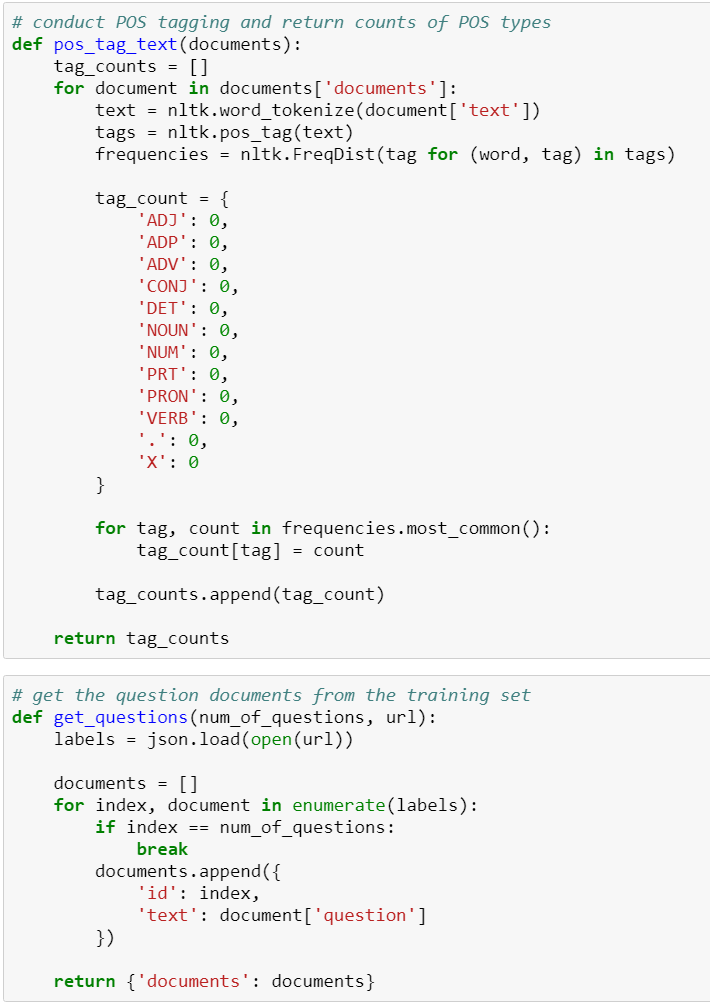












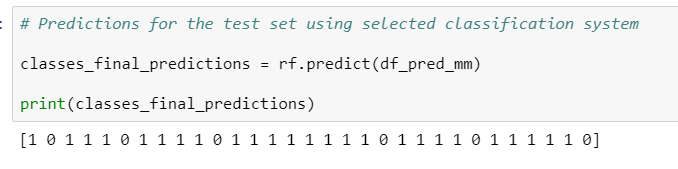


1. **Training/Validating models for selection**





1. **Test Predictions with chosen method**



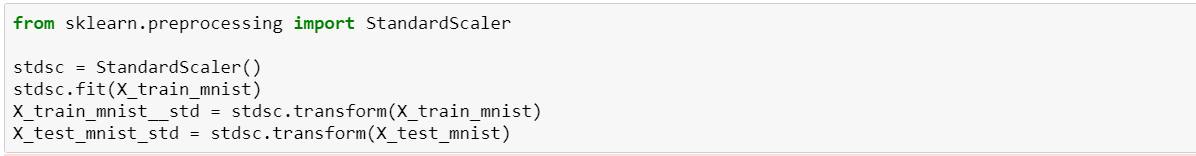
1. **Proposed Prediction Method**

My proposed prediction method was the Random Forest Classifier. Initial analysis of the datasets showed me that there was a lot of data and I struggled even loading the entire set into my notebook with the computing resource available to me. Immediately I knew that I would not be able to train my model with all this data and therefore I would need to select a model that does not absolutely require a large training set. At the start, I thought a decision tree would perform the best – which it did but then I realized that ensemble learning could be used to enhance this model to a Random Forest. Further the Random Forest is a predictive tool rather than a descriptive tool – this works in my favor as we are only interested in predicting the classes.

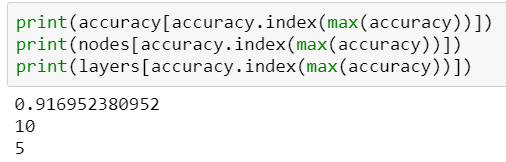
Just to confirm this hunch, I built several models using the training set (of only 101 examples) and tested the performance on the validation set (only 31 examples) – Random Forest did indeed performed the best. This makes sense to me as the random forest can be flexible and able to incorporate different kinds of features.

Design decisions were based on the complexity of the underlying relationship. Building a multi-modal that uses computer vision and natural language processing to concatenate input features and then predict a class is a complex problem. On top of that, the task of predicting if a visual question is answerable is clearly highly non-linear and therefore I decided to go the decision tree/ random forest route – non-linear relationship tend not to affect its performance as much.

It would be interesting for me to run a similar analysis but using the complete dataset (given enough time and resources). The Random Forest takes a lot of time to train and therefore I would need a longer time/faster computer. I suspect that my classification system is quite rudimentary as it is trained with such a small amount of data and I would be curious to see how the other models perform when built with a larger training set.

1. **Classification Using Neural Networks**
   1. **Load MNIST and create 70/30 train/test split**
   2. **Optimize Hyperparameters – 10 hidden layers and 10 number of neurons**





* 1. **Optimal Hyperparameters and number of weights found**

As seen from the results above, the optimal hyperparameters were 10 neurons per layer and 5 hidden layers. This neural network gave me an accuracy of 91.7%. Below is my calculations for the number of weights and parameters:

# of inputs = 784, # of outputs = 1, 10 neurons per layer, 5 hidden layers

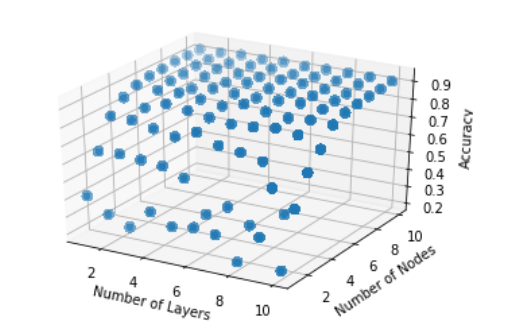
# of weights = (784\*10)+(10x10)+(10x10)+(10x10) )+(10x10)+(10\*1) = 8250

# of parameters = 8250 weights + 784 inputs = 9034

* 1. **Discussion regarding performance of neural network with different hyperparameters**

The performance of the neural network differs depending the hyperparameters used. When considering the number of nodes independent of the numbers of layers used, it is clear from the scatter plot below that the highest accuracy occurs when the number of neurons is at 10 per layer. There is a definite positive correlation between the number of nodes in a layer and the performance of the network. Although, the incremental gain after around 6 nodes per layer is not as much as going from 1 neuron to 5 neurons.

On the other hand, the number of hidden layers did not seem to contribute much to the performance of the network. Increasing or decreasing the number of hidden layers had no affect on the accuracy as the performance was poor for a low number of nodes at both 2 layers and 10 layers. Similarly, the performance was good at 10 nodes at both 2 layers and 10 layers.

These results make sense to me as increasing the number of neurons per layers should allow for a more intricate fit of the data – although I suppose, increasing the number of neurons to thousands can lead to overfitting – and that is why the accuracy is increasing when number of nodes is largest in this case. In contrast, it also makes sense to me that increasing the number of hidden layers does not help the accuracy. Adding more number of hidden layers enables the modelling of highly non-linear relationships and the fact that adding layers does not affect performance tells me that the underlying function in the data – it the relationship was more non-linear, I would expect the number of hidden layers to have a greater affect.