Naive Bayes Lab

`colour-data.csv` : A data set with almost 4000 data points that we can try to learn with.

Tasks

Using the Jupyter Workbook `bayeslabstart` that has the following: Normal importing for numpy/pandas/matplotlib. Also includes lab2rgb, rgb2lab, a listing of basic colours and a method for displaying your results.

Do the following in the workbook

1. Start by getting the data: read the CSV with Pandas. Extract the X values (the R, G, B columns) into a NumPy array and normalize them to the 0-1 range (by dividing by 255: the tools we use will be looking for RGB values 0-1). Also extract the colour words as y values.
2. Check the shape of these arrays, are they as expected?
3. Partition into training sets and test sets. You will need to import train_test_split
4. Create a GaussianNB model, call the model model_rgb You will need to import GaussianNB.
5. Print the score for the model and then plot_predictions for the model.

First model done.

The naïve Bayes approach implicitly assumes that distances in the input space make sense: distances between training X and new colour values are assumed to be comparable. That wasn't a great assumption: distances between colours in RGB colour space aren't especially useful.

Possibly our inputs are wrong: the LAB colour space `https://en.wikipedia.org/wiki/Lab_color_space` is much more perceptually uniform. Let's convert the RGB colours we have been working with to LAB colours, and train on that. The skimage.color module has a function for the conversion we need.

Do the following in the workbook

1. Create a pipeline model where the first step is a transformer that converts from RGB to LAB, and the second is a Gaussian classifier, exactly as before. Call this model model_lab
2. There is no built-in transformer that does the colour space conversion, but if you write a function that converts your X to LAB colours, you can create a FunctionTransformer to do the work. Create a function that does the following:
   - Some Numpy reshaping will have to be done as part of the function you create. skimage.color assumes a 2D image of pixel colours.
   - `.reshape(1,-1,3)` should work
   - then convert to lab (rgb2lab)
   - Reshape back to original shape `.reshape(-1,3)`
3. Make a pipeline that first uses FunctionTransformer(makelab) and then creates a GaussianNB model.
4. Same as above print score and the pictures

Maybe Naïve Bayes isn’t the best approach afterall. Repeat the above (both RGB and LAB) with a `LogisticRegression` model. By default, LogisticRegression model is a binary classifier, but in Sklearn you can make it a Multi-classifier by adding the option `multi_class='auto'` when declaring it.

What model performed best?

Examining the colour-data.csv the confidence column was not used. What happens to the models if we eliminate the ones with ”poor” confidence? (this will affect the training and test data sets!). This would require creating a new dataframe.