Exercise 8

Due Friday July 09 2021.

Some files are provided that you need below: E8.zip. As usual, **you may not write any loops** in your code.

Project

As we come to the end of the Machine Learning topic, this is probably a good time to look at the Project. There's no reason you shouldn't be able to start working on it any time.

Colour Words, Again

Last week, we used a Naïve Bayesian classifier to do the RGB values to colour words task. Since then, we have explored some more techniques: *k*-nearest neighbours and decision tree classifiers. Let's compare techniques…

Create a program `colour_predict.py` that takes the input CSV file on the command line, as last week. A **new** hint has been included this week to help get your input/output into the right shape.

Include your `GaussianNB`-based classifiers from last week, so we can compare: You should have a model that gives the original RGB colours to the classifier, and one that converts to LAB colours and then trains the classifier.

New this week: you can convert to either LAB *(https://en.wikipedia.org/wiki/CIELAB_color_space)* colour (as last week) or to HSV colour *(https://en.wikipedia.org/wiki/HSL_and_HSV)* . In LAB colour, distances make sense; in HSV colour, there is an axis "hue" that seems quite related to the predictions we're making and might be more useful for some models to work with.

In addition to `GaussianNB`, we will try two additional models: a *k*-nearest neighbours classifier (`KNeighborsClassifier`) and a random forest classifier (`RandomForestClassifier`).

For each of the three models, create a version that takes RGB colour directly, and a version that converts to LAB or HSV colour (your choice of which for each model). For each, adjust parameters for best results.

When finished, your `colour_predict.py` should **print the scores on your validation data in the format included in the hint**. Please do **not** have a `plt.show()` in your code when you submit: it makes marking a pain.

Case of the Unlabelled Weather

We have met the GHCN *(https://www.ncdc.noaa.gov/data-access/land-based-station-data/land-based-datasets/global-historical-climatology-network-ghcn)* data before… but oh no! When recording the 2016 data, somebody forgot to record *which city* the observations came from. Whatever will we do? *

For this question, I took data from 26 North American airports and extracted several features: daily minimum and maximum temperature (in 0.1°C), amount of precipitation (in 0.1mm), snowfall (in mm), and snow depth (in mm). For each feature, I calculated the monthly average for each month of the year, for a total of 60 features for each city and year.

In the provided `monthly-data-labelled.csv`, you will find all of these features, as well as the name of the city and year of the observations. The file `monthly-data-unlabelled.csv` is the same, but with the city name redacted: that's what we're hoping to reconstruct.

Create a program `weather_city.py` that reads the labelled data and trains and validates a machine learning model for the best possible results. It should then predict the cities where the unlabelled 2016 weather came from.

The command line should take filenames for the labelled, unlabelled, and output files:

```
python3 weather_city.py monthly-data-labelled.csv monthly-data-unlabelled.csv labels.csv
```

The output format (into the file given as the third command line argument) should be one city name per line, in the format this line produces:

```
pd.Series(predictions).to_csv(sys.argv[3], index=False, header=False)
```

Your program should **`print` one line**: the "score" of the model you're using on a validation subset of the labelled data.

The features have very different magnitudes: the features with units 0.1°C maximum temperature and millimetres of snow are nowhere close to the same scale. You'll probably want to normalize into a predictable range. Hint *(http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html)* .

* Nobody from the GHCN forgot to record what city data originated. They are all fine, upstanding data collectors.

Exploring the Weather

Why did that work? How was a machine learning model able to take the weather observations and (usually) come up with the correct city? We can explore the data a little more to get a sense of its structure.

See the attached `weather_clusters_hint.py` which (when completed) can be run like:

```
python3 weather_clusters.py monthly-data-labelled.csv
```

Start with the same `X` and `y` values you used in the previous part: all of the observations, and the correct cities.

We will first use principal component analysis to get two-dimensional data that we can reasonably plot. Fill in the provided `get_pca` function so it returns the `X` data transformed to its two most "important" features. Hint: MinMaxScaler *(http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html)* seems to work better here, and of course you'll need the `PCA` model.

We can also use a clustering technique to find observations with similar weather. Fill in the provided `get_clusters` to find 10 clusters of similar weather observations using KMeans *(http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html)* clustering.

With that, you should get a scatter plot of the clusters: note that none of the input to the plot used the `y` values. It was created just by examining the observed `X` values.

The provided code also creates and prints a table of how many observations from each city were put into each category (using the `y` values now). You should be able to see here which cities have similar weather patterns.

Questions

Answer these questions in a file `answers.txt`.

1. Which model did the best for the colour-prediction task? Can you give a theory about why? Do you have any theory about why RGB/LAB/HSV worked better for different models?
2. Have a look at the cities in your validation data where the weather model makes the wrong prediction. Do you feel like the model is making reasonable mistakes? Can you think of any weather features that we could potentially add to make better predictions?

Here's a hint for that, but please **comment-out** the `print` before submitting, so we don't have to wade through the output. I'm not as concerned about your answer here as you looking at the predictions your model makes and evaluating it with a human-brain-based critique, not just an accuracy score.

```
df = pd.DataFrame({'truth': y_valid, 'prediction': model.predict(X_valid)})
print(df[df['truth'] != df['prediction']])
```

Submitting

Submit your files through CourSys for Exercise 8.

Updated Wed April 07 2021, 09:16 by ggbaker.