Report

This project aims to explore how weather of some place at a given time can be predicted by values of a few weather indicators or by simply an image at a particular time, and also possible similarities or differences found during implementation or results, between the two approaches. In specific, data about a same period of time (09/01/2017 – 10/31/2017) is taken as the experimental input as we want to compare the results of weather prediction in this period, while data prior to 09/01/2017 is used for training the model so that we could know which approach would perform better under the same initial circumstances.

All original data in need are downloadable under the project's description. There are two sets of data: one set came from Vancouver Airport weather station that contains the values of weather indicators, and the other is a bunch of images taken by Kat Kam somewhere in downtown. Before utilizing the data immediately, data cleaning and refinement is necessary since 1. Data is read in format of Spark dataframe as data are partitioned in files, I'd rather convert it into Pandas to make it easier for operations; 2. There are rows in the original .csv input file that are useless or meaningless; 3. Columns may be needed to be split or redefined; 4. Some weather description should be excluded for classification with convenience. etc. At the very beginning I ever tried to deal with those data on Spark dataframe because there is more than one .csv input file. But afterward I gave up since I found it not that convenient and efficient when doing operations on data. Also, another advantage of converting the data into Pandas dataframe right after reading in the original data would be that when I end up clearing up the data I can easily export it into a single file without coalescing it with Spark, especially when the whole size of data is small.

The two methods of weather prediction are implemented in separate scripts and they both read the cleaned input data as described above. So they are processing independently to each other. As known, Machine learning (with scikit-learn) enables us to predict something by classification using a model after trained on data with known results, which means the data is split into two part, one, with both x and y values, used for training, and the other, with only unseen x values, for prediction. I found this one of the difficult part in the project because this is actually a multi-label classification problem since there is probably more than one sub-class for each category, like 'heavy rain', where the former of the phrase describes the degree of the later. This does make significant difference because each label appearing in y, the target data, should be classified and treated independently. In scikit-multilearn package, this can be achieved by transforming y (originally a series in Pandas dataframe) into a sparse matrix, with every label representing a column, and for every observation of weather data, an entry of '1' implying the presence of corresponding label, otherwise '0'.

As for the Predict-by-image approach, another difficulty that I have encountered the first time is that how to read all the images in one folder into my program and represent them as a 2-d array containing features as inputs. There are some nice ideas on the internet showing that some package like 'cv2' is able to read image files very efficiently, into a format of a n-d array (RGB values). Therefore, all images can be read one by one in a for loop. Moreover, for each image loaded, the time the image was taken can be determined and the image can be easily reshaped into a 1-d array as wanted. One thing to notice is that the number of images is less than the number of weather observations, this is why we need to know the time of the image so that those weather data matched by the images are considered only.

The training and testing processes are similar between two approaches: training the model using different classifiers and pick the one which produces the highest testing score. Here we need either a combo: a basic classifier we have seen in lectures such as GaussianNB, SVM, plus a multi-label transforming method such as BinaryRelevance and LabelPowerset, or simply an adapted method, MLkNN, for example. Each of them performs slightly differently between two prediction approaches. Interestingly, the combination of KNeighbors and LabelPowerset wins in both approaches, as though none of their accuracy scores is over 5.0. As expected, I ran into an issue of efficiency when training with the image data since there are still nearly 10 thousand features of each image, after resizing them from 256*192 to 64*48. So, as seen in class, I applied the PCA method to reduce the number of features of each observation to 300. I did not count how much exactly it improves the speed of processing but there must be some.

At last, as the output prediction data is as well in a format of sparse matrix, I need to do some transformation to turn it back to something verbal. This is a little tricky I think but actually it can be done by applying an ufunc to the matrix. As we have discussed above, the results of predictions are both not that satisfying. I found that other than 'Clear', 'Cloudy' these kinds of weather that present most frequently, those rarely appearing weather such as 'Snow' and 'Fog' etc. are nearly never predicted correctly. Checking through the original weather data, I suppose this might be due to that the vast majority of the days are indeed not snowy or foggy, so these kinds of predictions are hardly generated even though the weather data seems to indicate that there is a great probability of a snowy or foggy day. Another guess is that the process of either training or predicting does not take into account the date, especially the month, as one of the important features, since which seasons it is does have huge effect on the tendency of which kind of weather will be present.

So far in my opinion predicting the weather is not an easy job, at least it is not enough to make it by simply putting some data in a simplified ML model. I did not combine the weather data features with the images features together to make predictions because I think it would be a bit confusing. But if I had more time I will certainly explore this part.

Project Experience Summary

For the sake of fast debugging and testing, all scripts are written and tested on Jupyter notebook as the whole project is only about using Python. All calculations and input/output are mainly done with Pandas but also Spark since this is the one of most common ways to manipulate small-size data. For I/O, some techniques are used to read and write the data in format of table or image, and for calculation, different kinds of dataframe operations are carried out in order to do aggregation, iteration and data cleaning etc., which can be efficiently accomplished on Pandas.

To make predictions based on provided data, idea of Machine Learning is applied to establish models for classification use. Data is split into training data and testing data such that a ML model is able to set parameters to learn how to predict y from x, and also, users are able to see how the model performs using testing values. When necessarily, input data sometimes needs to be transformed: to improve processing efficiency while not losing too much reality and accuracy. The best predicted results are taken as final predictions, whereas clearly the results it produces is not that 'good'. I personally think there are two reasons: 1. Predicting weather itself is actually not an easy task. 2. The data that is used might be somewhat superficial and the model that is built on the basis of it is too simplified to have great accuracy.