TU Delft

Codelab - 4

EE4C12

29.09.2022

Feature Engineering in Weather Forecasts

Intro

Extreme precipitation usually leads to substantial impacts. Floods in the Netherlands, Belgium and Germany in the summer of 2021 has caused loss of lives, destruction of infrastructures, and long-term effect on economics. To avoid such disasters, it is essential to develop a reliable and accurate method to predict heavy rain. In this CodeLab, we will explore and analyze rainy events in weather forecasts using features universal approximators in classification and regression techniques. Notably, we will study a dataset containing real precipitation maps of the Netherlands (Fig.1). We will process such datasets to create different versions that present different balances between rainy and not rainy events. Moreover, we will downsample the given dataset to apply regression techniques.

Provide your answers in the Jupyter Notebook either as a comment or insert a markdown cell. Please print out your Jupyter Notebook in PDF format (or html) and upload it to Brightspace. You will get points for each part separately and based on your overall performance you will get the grades. Table 1 illustrates the points of each part and Table 2 shows the grading scheme of CodeLab 4.

Table 1: Points of CodeLab4

Part	Points
Task 1	3
Task 2	3
Task 3	1
Task 4	1
Task 5	3
Total	11

Table 2: Grading scheme of CodeLab4

Grade	Points
0	≤5
6	6-7
8	8-9
10	≥10

Task 1 - Data preparation (3 points)

The dataset contains precipitation images of 2017 in 'png' format. Exploring the dataset and seeing what samples look like is a good idea. First, upload the zip file "dataset_codelab_4" to Jupyter environment and unzip the file with the given code. Note: Unzipping operation might take up to 10 minutes, do not interrupt the process; you have to do unzipping only once.

Use the given function to retrieve the given sample. Print out the shape of the image and plot using $plt.imshow()$ function.

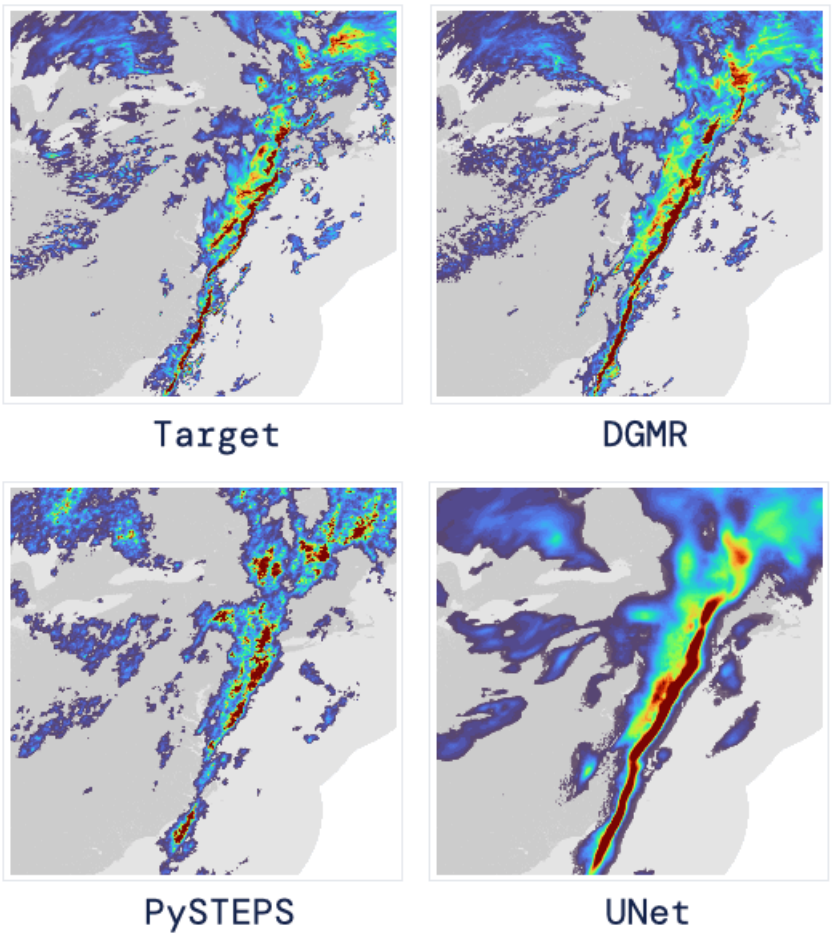

Figure 1: Example of precipitation image and predictions from different models (e.g., PySTEPS, UNET, etc)

Every sample is defined as png file, as the format of the defined name is "RAD_NL25_RAP_-10min_" + year + month + day + hour + minute. Let's define the years, months, days, hours, and minutes to extract all images in a structured way. In order to reduce the size of the dataset, we have set the minutes portion as 0. Months and hours variables are divided by 100 to obtain strings from the last 2 decimal digits. Use nested for loop structure to read images sequentially and insert them into a NumPy array called "images". Use cv2.imread() function to read images that require string concatenation to obtain the file path (Check the first example of how it is called, Python strings: https://docs.python.org/3/library/string.html).

Once you collected all images into a unique tensor (3d array), we can preprocess the data to use them as we want. We are going to create six different versions of the dataset. We'll first change the balance between rainy images and not rainy ones. Then we will downsample the images from 64x64 to 8x8 in order to use them for regression tasks.

Dataset balancing and downsampling

First, we should preprocess and investigate the data by taking the average and sorting it.

- Subtract 33 from all images in the dataset to obtain minimum (dry conditions) as zero.
- Compute the average precipitation of each image and store it in the array "avg_data".
- Sort the average precipitation in ascending order.
- Compute the mean of the entire dataset.
- Calculate and print out the sample portion that is higher than the threshold.

Heavy rain condition is selected as any processed image that contains an average greyscale pixel value of more than 3. Plot the following variables in the same figure: Sorted average precipitation, dataset mean, and the threshold value (3).

Dataset distribution shows us a significant imbalance between heavy rain and regular conditions which is an expected outcome. However, this situation could cause a certain bias in our machine learning models, reducing the performance. Although there are many methods to solve the imbalance, we will apply downsampling to the majority class.

The balanced dataset contains 40% of the original dataset. Not to lose any information from the heavy rain conditions, we should retrieve the top 20% of samples from the sorted average vector. From the rest you can use the $resample(X, replace = False, n\_samples = N, random\_state = 4720)$ function from sklearn to sample. (Hint: you can use indices of average vector to obtain both sample cases.)

Apply the same procedure to obtain 60% of the original dataset where the top 30% is directly included and another 30% is sampled from the rest (70%), called lightly balanced.

After obtaining 3 datasets, let's create the corresponding label vectors. Images with mean lower than the heavy rain threshold (3.0) are labelled as 0 and higher cases as 1. Create 3 different label vectors correspond to each dataset.

Still our datasets are large for processing so we will reduce image size from 64x64 to 8x8. Use cv2.resize function to obtain 3 new datasets for the original, balanced and lightly balanced cases and their labels.
Plot the first images of each dataset from both large images (64x64) and small images (8x8).
Questions:

- Why do we need to balance the dataset between rainy and not rainy eventes?
- What other methods could be used to balance this dataset?

Task 2 - Spatial Regression (3 points)

In this exercise, we are going to perform regression over images' spatial coordinates. In other words, given a specific position in the image, we want to infer the precipitation value of the center pixel (target) of small balanced dataset (8x8) which corresponds to $X[4, 4]$. Features will be the neighbouring pixels of the center, you can see an example in Fig. 2. The aim of this exercise is to analyze spatial regression performances using the information of neighbours (e.g., adjacent pixels).

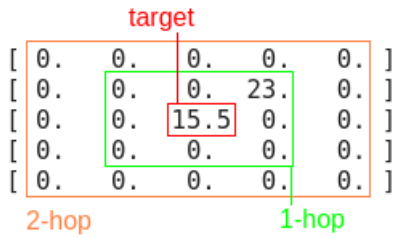

Figure 2: Example X_reg sample. Target, 1-hop and 2-hop of neighbors are explicitly shown.

First, start with obtaining 3x3 matrices from each image and remove the center value as the target variable y_{reg} using the small balanced dataset (!Warning: If you use np.delete(), make sure to first copy the data with np.copy(), otherwise you can delete the variable from the original dataset permanently). Then flatten the 3x3 matrices to obtain 8 features, since the center is removed, per sample in the feature matrix X_{reg}. After obtaining both X_{reg} and y_{reg}, split into training and validation set by using $train\_test\_split$ function from sklearn where the random state is 4720. Finally build K-Nearest Neighbour (KNN) regressor with 20 neighbours ($n\_neighbours = 20$) and train the model with the training data. Test the model performance with the validation data using RMSE as the performance metric. Expand the same methodology by creating 5x5 and 7x7 matrices which is equivalent to taking second and the third adjacent pixels.

Apply the same procedure for the balanced dataset (64x64) but we have to consider more adjacent pixels. 5th, 7th, and 9th hops (adjacent pixels) will correspond to 7x7, 11x11 and 15x15 matrices before flatting to X_{reg} and the center pixel should correspond to y_{reg}. Use the same number of neighbours in KNN regressor.

Questions:

- Do we get better results using more pixels in the small balanced dataset and the balanced dataset? If yes/no, why?
- Is the model underfitting or overfitting? Motivate the answer.

Task 3 - Precipitation Forecasting (1 point)

In this task, we are going to perform precipitation forecasting of the image averages. In order to do that, we are going to use an autoregressive integrated moving average (ARIMA) model. ARIMA model is a known model in time series analysis, It is used either to better understand the data or to predict future points in the series (forecasting). In such analyses, the amount of data used to fit the model also represents how far we can go back in time in our time series analysis and affect the quality of the predicted points.

Using the original average precipitation dataset's (8x8) first 2000 samples to train and obtain model predictions with $model.fit().fittedvalues$. Plot true data and fitted values in the same figure. Calculate and print the MSE.

Since the model is ready we can start predicting the future average precipitation values. Predict and the next (3, 5, 7, 12, 15) hours with the function $model.fit().predict(1, i)$ where i is the future prediction window, compare the accuracy of such predictions using MSE. Plot both true and predicted future average precipitation.

Questions:

- What happens when we increase the future time-window in terms of predictive performance?
- Is the model overfitting or underfetting? Motivate the answer

Task 4 - Extracting features using Principal Component Analysis (1 point)

Principal component analysis (PCA) is a statistical procedure that allows you to summarize the information content in large data tables by means of a smaller set of "summary indices" that can be more easily visualized and analyzed. In this case, we are going to extract the most relevant features from our data. We will do that using the sklearn package https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html. It is important to note in order to apply PCA we need to first flatten our data. Moreover, we can choose the dimensionality of the extracted features.
In this task we are going to:

- Implement a PCA feature extractor with 30 principal components for each 6 datasets (original, balanced, lightly balanced, small original, small balanced, and small lightly balanced).
- Extract the features from each 6 datasets.

Questions:

- What are the limitations of the adopted approach?
- How the dimensionality N affect the extracted features?

Task 5 - Classification of Rainy Events (3 points)

In this task we are going to classify images and the related extracted features as rainy or not rainy events of the defined dataset. We will define a function $Classifier$ that splits the given dataset intro training and validation sets, fit the classification model, compute Accuracy, Recall, Precision, F1, and AUC and plot both ROC curve and the confusion matrix.

$Classifier$ function takes the following inputs: sklearn classification model, feature matrix X_in and classification labels Y_in. Train-test split uses 25% of the data as test data with the random state as 4720. The function prints out Accuracy, Recall, Precision, F1 and AUC scores and plots confusion matrix, ROC curve of training and ROC curve of test data (either in same figure or different).

Two classic classifiers Logistic Regression and Decision Tree Classifier will be used for the binary classification task at hand. Apply the following steps.

- Develop the function $Classifier$

- Call the function with logistic regression and the original (flattened) dataset.
- Call the function with decision tree classifier and the original (flattened) dataset.
- Call the function with logistic regression and the small original (flattened) dataset.
- Call the function with decision tree classifier and the small original (flattened) dataset.
- Call the function with logistic regression and the balanced (flattened) dataset.
- Call the function with decision tree classifier and the balanced (flattened) dataset.
- Call the function with logistic regression and the small balanced (flattened) dataset.
- Call the function with decision tree classifier and the small balanced (flattened) dataset.
- Call the function with logistic regression and the lightly balanced (flattened) dataset.
- Call the function with decision tree classifier and the lightly balanced (flattened) dataset.
- Call the function with logistic regression and the small lightly balanced (flattened) dataset.
- Call the function with decision tree classifier and the small lightly balanced (flattened) dataset.

Repeat the previous steps with the extracted features as well. Write down your observations about different models, datasets and feature extraction. Answer the following questions.

Questions:

- What differences do you find between train and test ROC curves in logistic regression and the Decision Tree model? Elaborate on those differences.
- What does the confusion matrix of the decision tree classifier applied to the balanced dataset show? Explain your observations.
- Plot ROC curve and confusion matrix of logistic regression classifier and Decision Tree Classifier.
- Compare the results you got by classifying directly images and classifying the related features. To what extent the results we get are correct? What phenomenon are we observing when we use unbalanced datasets?
- What methods could be used to fix such behaviours?