Exercise 9

Evaluation

Evaluating explanations is hard, because typically there is no groundtruth data available. Additionally, the evaluation heavily depends on the stakeholder and their objective. In this exercise, the task is to implement some proxy metrics based on feature removal.

- You can get a bonus point for this exercise if you pass at least 85% of the tests. Code is automatically tested after pushing. If your tests fail, either your code is wrong, or you solved the task differently. In the second case, we will manually check your solution.
- Three collected bonus points result in a 0.33 increase of the final grade.
- You are allowed to work in groups with up to three people. You do not need to specify your ErgebnisPIN as it is already collected from the entrance test.
- Follow the 'README.md' for further instructions.
- Finish this exercise by 17th December, 2021 at 11:59 pm.

Comprehensiveness and Sufficiency

The idea behind both comprehensiveness and suffiency stems from the following hypothesis:

> *If a feature is important, then removing it should result in a significant change of the prediction.*

The comprehensiveness measures the difference between the original prediction and the prediction after removing important features:

$$\text{comprehensiveness} = m(x_i)_j - m(x_i \setminus r_i)_j \tag{1}$$

where m is the model, x_i are the features of instance i and $x_i \setminus r_i$ is the same instance i with some of its most important features r_i removed or replaced by a baseline value. In our case, we want to measure the difference in output probability of the predicted class between the original prediction and the prediction with some features removed. Before complete the *compute_comprehensiveness* function, you have to complete *generate_explanations*, to generate an explanation for each instance in the dataset based on the input gradient saliency method. Additionally, fill in the helper functions *get_most_important_features* and *get_predictions*.

The sufficiency metric assumes that the most important features alone should be sufficient to make a good prediction:

$$\text{sufficiency} = m(x_i)_j - m(r_i)_j \tag{2}$$

It measures the difference between the predictions for an original instance and the prediction for the most important features of that instance. Complete *compute_sufficiency* that computes this metric for a given model and a dataset.

Associated file: *evaluation.py*.

Remove and Retrain

The above metrics are computed without retraining the model. The problem is that the partial instances created by removing some features can be considered out-of-distribution examples; one could argue that we cannot expect the model to perform any good on these examples, as it was not trained on similar examples. To overcome this issue one can retrain the model after removing the most important features from a dataset and measure the resulting performance. If a feature is important, then removing it from the dataset should result in reduced performance of the retrained model. The hypothesis is that

- If the test accuracy drops significantly, the removed input features were informative, and thus important.
- If the test accuracy does not drop, the removed input features were uninformative or redundant.

Complete the *remove_and_retrain* function that trains a new model after removing the k most important features of each instance in the train and test data.

Associated file: *evaluation.py*.

Visualization

Finally, complete the *plot_scores* function, that can be used to plot a series of comprehensiveness, sufficiency or accuracy scores in a line chart. Think about what insights can you takeaway from each plot.