Lab 4

Fan Dong (3033122445), Mengling Liu, Nicholas Sim (26929549)

November 9, 2018

1 Introduction

Clouds play an essential role in modulating the sensitivity of the polar regions to increasing surface air temperatures. However, due to the presence of snow and ice, cloud detection remains a vital yet challenging part of Arctic/global climate modeling. With novel radiation recordings from NASA Terra satellite and state-of-the-art classification methods, this report seeks to provide a working solution to the cloud detection problem.

2 Data

The radiance measurements used in this project were collected by Multiangle Imaging Spectro Radiometer(MISR), which was placed on board the NASA Terra satellite in 1999. The MISR sensor comprises nine cameras, with each camera viewing Earth at a different angle. The nine view zenith angles are 70.5(Df), 60.0(Cf), 45.6(Bf) and 26.1(Af) in the forward direction; 0.0(An) in the nadir direction and 26.1(Aa), 45.6(Ba), 60.0(Ca) and 70.5(Da) in the aft direction. MISR cameras cover an approximate 360-km-wide swath on the Earth's surface that extends across the daylight side from the Arctic down to Antarctica in about 45 minutes. They collect data from all swaths ion a repeat cycle of 16 days with each pixel covers a 275m $\times$ 275m region on the ground, resulting in a huge amount of data over the whole globe.

The data also include three additional features, which are engineered by Shi & Tao et al.(2008) after substantial exploratory data analysis: the correlation of MISR images of the same scene from different MISR viewing directions (`CORR`) that characterizes the scattering properties of ice- and snow-covered surfaces, the standard deviation (`SD`) of MISR nadir camera pixel values across a scene and a normalized difference angular index (`NDAI`) that characterizes the changes in a scene with changes in the MISR view direction.

This project uses radiances and additional features of three MISR images to train and validate the cloud detection classification model.

3 Exploratory Data Analysis

3.1 Expert Labels for the Presence or Absence of Clouds

Figure 1 shows the expert classification of the three images. We will treat these labels as the true values to train and test the classification models.

3.2 Explore the Features

Next, we explore the relationships among the features. In Figure 2, we plot the correlation matrix of the true labels, radiances and additional features. As can be seen, the three engineered features are all negatively correlated with the true label while the radiances are all positively correlated. `NDAI`'s association is particularly strong, followed by `CORR`, `AF`, `AN` and `SD`. The correlation matrix gives us a good intuition for

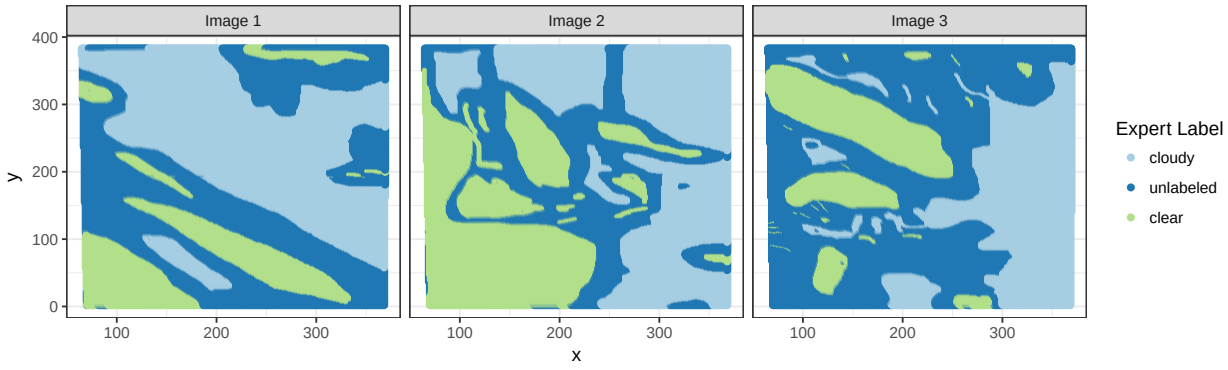

Figure 1: Images with Expert Classification

covariate selection, which will be detailed in the next section.

The radiances of different angles are highly correlated, with most of the correlation coefficients in the range of 0.8 0.9. This suggests that variable selection would be necessary before we train models, as linear and generalized linear models such as logistic regression model perform poorly with highly-correlated features.

Lastly, we create the density plots of all covariates by the expert label in Figure 3, in order to check how the radiances and engineered features behave in the presence or absence of cloud. In line with the correlation plot, `NDAI` does the best job in separating cloudy region from the clear part, as the great majority of `NDAI` > 1 pixels indicate absence of cloud. The distributions of CORR and SD are not as indicative. While the larger-value pixels are more prone to be clear, in the lower ranges, the two labels mostly overlap, making it hard to separate one from another. The radiances of different angles share similar pattern. While the cloudy regions clearly have higher radiances, due to snow and ice-coverage, the radiance of clear regions spans the entire range. The large overlapping area makes it challenging to detect cloud coverage.

4 Modeling

4.1 Feature Selection

From the density plots of the covariates in Figure 3, `NDAI` appears to be the best able to separate points where there are clouds from those that are clear. This distinguishing feature can also be seen in `CORR`. The radiance values and `SD` mostly have overlapping densities between cloud and non-cloud labels, and it was thus difficult to visually select another covariate. Therefore, in preparation for the ensuing modeling, we utilized a more quantitative approach to select the most predictive features.

Lasso with cross-validation was employed here for variable selection and we set the loss function to be the Area Under Curve (AUC). As Figure 4 shows, AUC increases as λ approaches 0 and more features are included in the model (the sequence of numbers on top of the chart indicates the number of nonzero coefficients of the corresponding point). The AUC gradually stabilizes as number of selected features moves beyond 4. Thus, four features would be ideal if we want to get relatively accurate prediction results with minimum number of features.

Taking a closer look at the `glmnet` results, we see that the four best predictors selected by Lasso is `NDAI`, `CORR`, `DF` and `SD`. Comparing with their density plots in Figure 4, the order of the features' predictive power agrees with how well they are able to separate cloud regions from non-cloud.

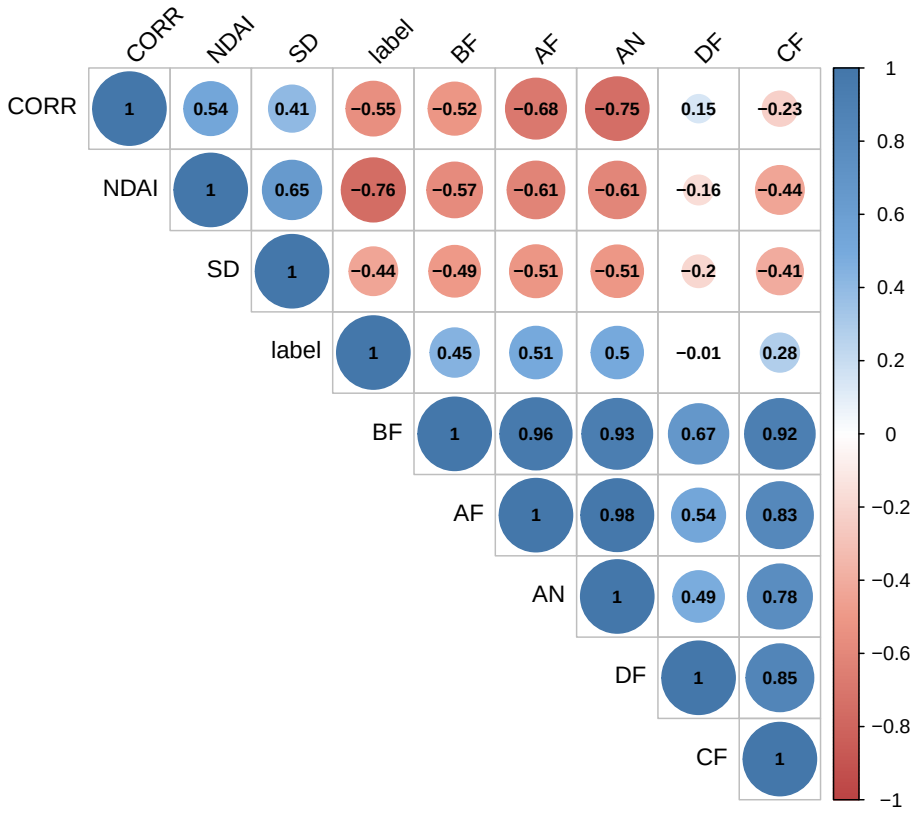

Figure 2: Correlation between expert labels and covariates

4.2 Training the classifier

As seen in Figure 1, Image 1 and 2 contains a large amount of either cloudy (green) or clear (light blue) data points which we can use to train our classifier. In contrast, Image 3 has a substantial proportion of unlabeled points (dark blue) which would not useful in the learning process. As such, we kept Image 3 as a holdout set to test potential candidate classifiers after they have been tuned. While we could have used a portion of Image 3 to train our classifier, we decided it would be more kosher if we kept the entire image out since there could be hidden correlations among the different variables within the same image.

In order to train our classifiers, we needed to first tune their hyper-parameters. We selected 3 base classifiers known to be computationally efficient while not sacrificing accuracy, viz. LASSO (from `glmnet`), XGBoost (from `xgboost`), and Random Forest (from `ranger` - which is supposed to be a faster implementation of `randomForest`). For both LASSO and XGboost, we needed to tune the penalization parameter, whereas for Random Forest, we tuned the number of variables to possibly split at each node. We tuned these parameters via 5-fold cross-validation where the learning data comprising of Images 1 and 2 were split into 5 folds. At each iteration, all our candidate classifiers were trained on 4 of the folds, and the prediction error on the remaining fold was evaluated and the averaged across all 5 iterations. The optimum value of the tuning parameter for each family of classifiers was then chosen based on the lowest average risk.

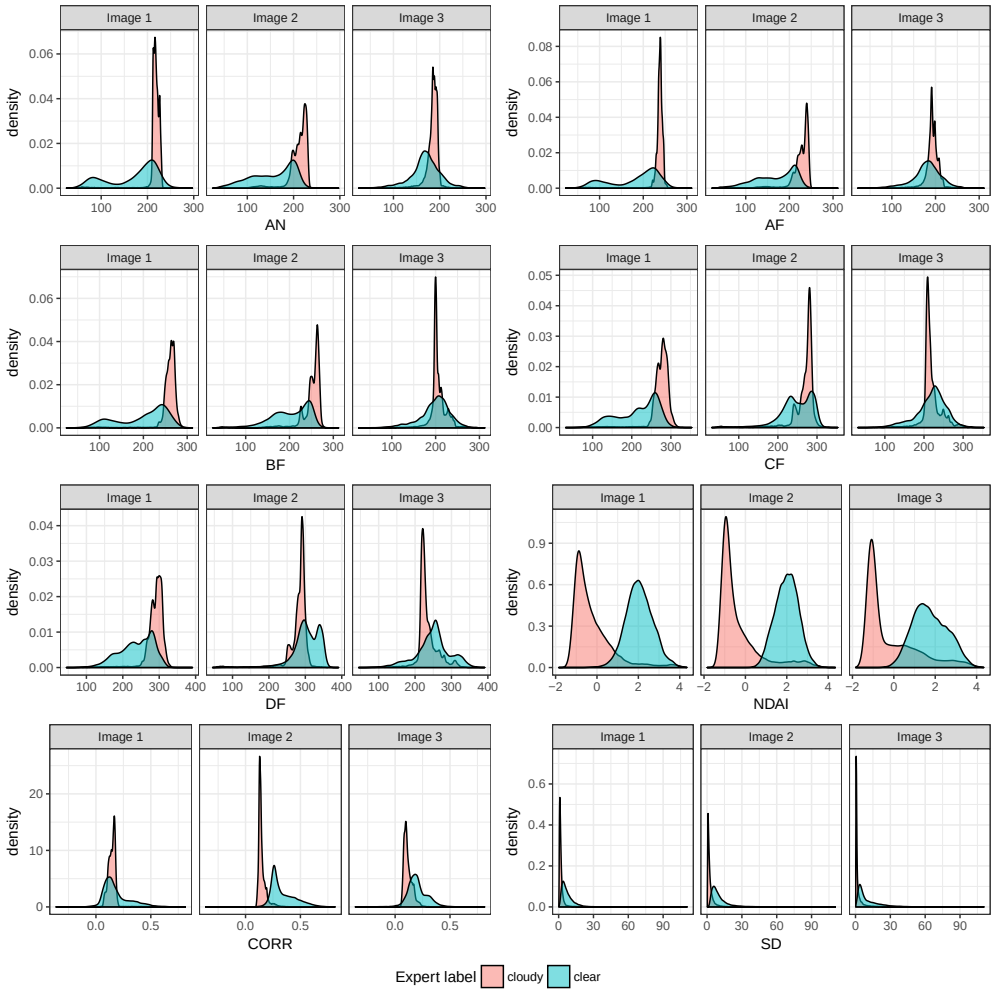

Figure 3: Density plots of all covariates

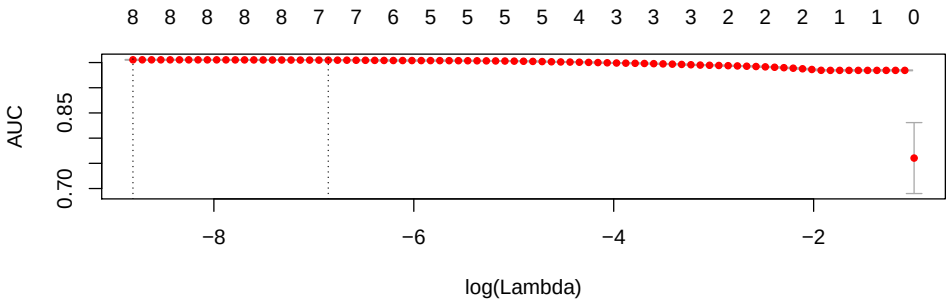

Figure 4: LASSO: Number of non-zero coefficients, lambda and corresponding AUC

There were some specifics with regards to the cross-validation process which we had to contend with. We implemented the procedure by making use of the `SuperLearner` package usually used for ensemble learning.

- (Loss function) The optimal tuning parameters for each family of classifiers were chosen to be the one with minimal risk among those we considered. We chose to minimize the rank loss function using

the Nelder-Mead method which has been shown to be equivalent to maximizing the AUC. This was specified using the `method` option in `SuperLearner`.

- (Spatial data) As the data we had was spatial in nature, points in some neighborhood of each other can be thought to be more related to each other. This violates the iid assumption if we formed the folds not acknowledging this relationship. As such, we split both Images 1 and 2 into 10 equally spaced columns and rows for a total of 100 cells. Then, we formed folds of for our cross-validation by ensuring that all observations in the same cell were in the same fold. This was specified using the `id` option in `SuperLearner`.
- (Tuning parameters) For the shrinkage parameter for `xgboost`, we chose to optimize over the values 0.1, 0.2 and 0.3. For the number potential variables to split at each node, we chose to optimize over the values 1, 2 and 3. Other than `cv.glmnet` which has its own cross-validation procedure built in to tune the penalty term, we had to specify variants of each of the other two classifiers based on the list of tuning parameters we specified above.
- (Parallel computing) As the dataset is large and the classifiers computationally expensive, we chose to parallelize our model using `snow`.
- (Superlearning) Although `SuperLearner` was built for ensemble learning, the eventual ensemble classifier performed just as well as our final classifier and hence, we chose not to present findings related to that.

We present the the average risk for the classfiers we considered in Table 1 below. Note that we only present the optimal `glmnet` classfier as `cv.glmnet` by default searches over a grid of 100 values. It is clear from the cross-validated risk that with each family, we should be choosing `glmnet` with `lambda` = 0.0007, `xgboost` with `shrinkage` = 0.1, and `ranger` with `mtry` = 1.

Table 1: Average rank-loss risk for classfiers

classifier	**tuning parameter**	**rank-loss risk**
`glmnet`	0.0007	0.037239
`xgboost`	0.1	0.011678
`xgboost`	0.2	0.011965
`xgboost`	0.3	0.012211
`ranger`	1	0.011725
`ranger`	2	0.012099
`ranger`	3	0.012502

4.3 Choosing the threshold

When each classifier is applied to a set of data, it returns as output a set of predicted probabilities that the observation should be classified as cloudy. In order to determine the actual class, we need a threshold for the predicted probabilities above which we classify the observation as cloudy and below which we classify the observation as clear. To determine these threshold, we fit each of the tuned classifiers to the entire learning set (i.e., both Images 1 and 2 in their entirety) and computed the threshold that would maximize the True Positive Rate of classification whilst keeping the False Positive Rate below 0.05. As seen from Figure 5, at a maximum False Positive Rate of 0.05, both `glmnet` and `xgboost` have a True Positive Rate of 0.964 and 0.959 with thresholds of 0.446 and 0.466 respectively. `glmnet` achieves only a False Positive Rate of 0.796 at a threshold of 0.733.

4.4 Testing classifiers on holdout set

We then tested our three classifiers on the holdout set as a fair assessment of its performance. Visually, from Figure 6, as we compare the expertly labeled points on Image 3 to that produced by our three tuned

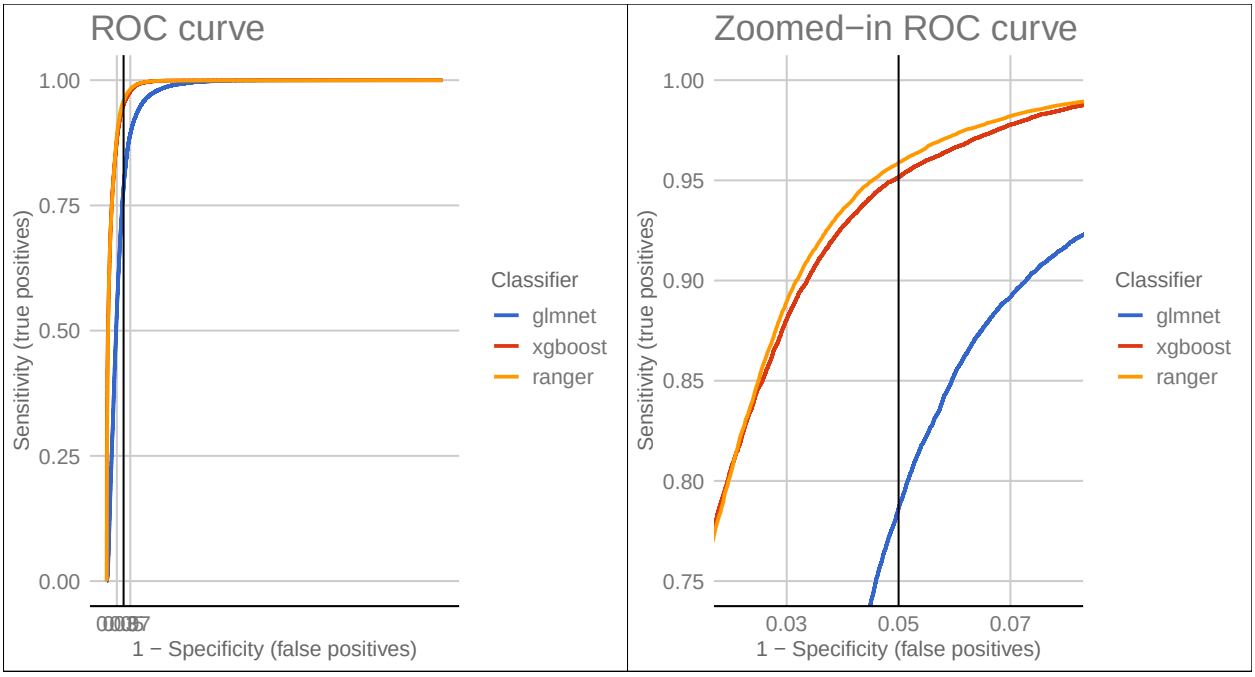

Figure 5: Receiver operating characteristic curves for each tuned classifier.

classifiers, we see that `xgboost` performed the best with an accuracy of 70.5% (ignoring the regions that were unclassified by the experts). This is followed surprisingly by `glmnet` with an accuracy of 69.5%. Lastly, `ranger` achieved only an accuracy of 46.53% (with 95% confidence interval from 46.12% to 46.95%) due to mis-classifying a huge number of points for $x > 300$ and $y \in (75, 275)$. All three classifiers also failed to accurately classify the region in the bottom right corner ($x > 300$ and $y < 75$).

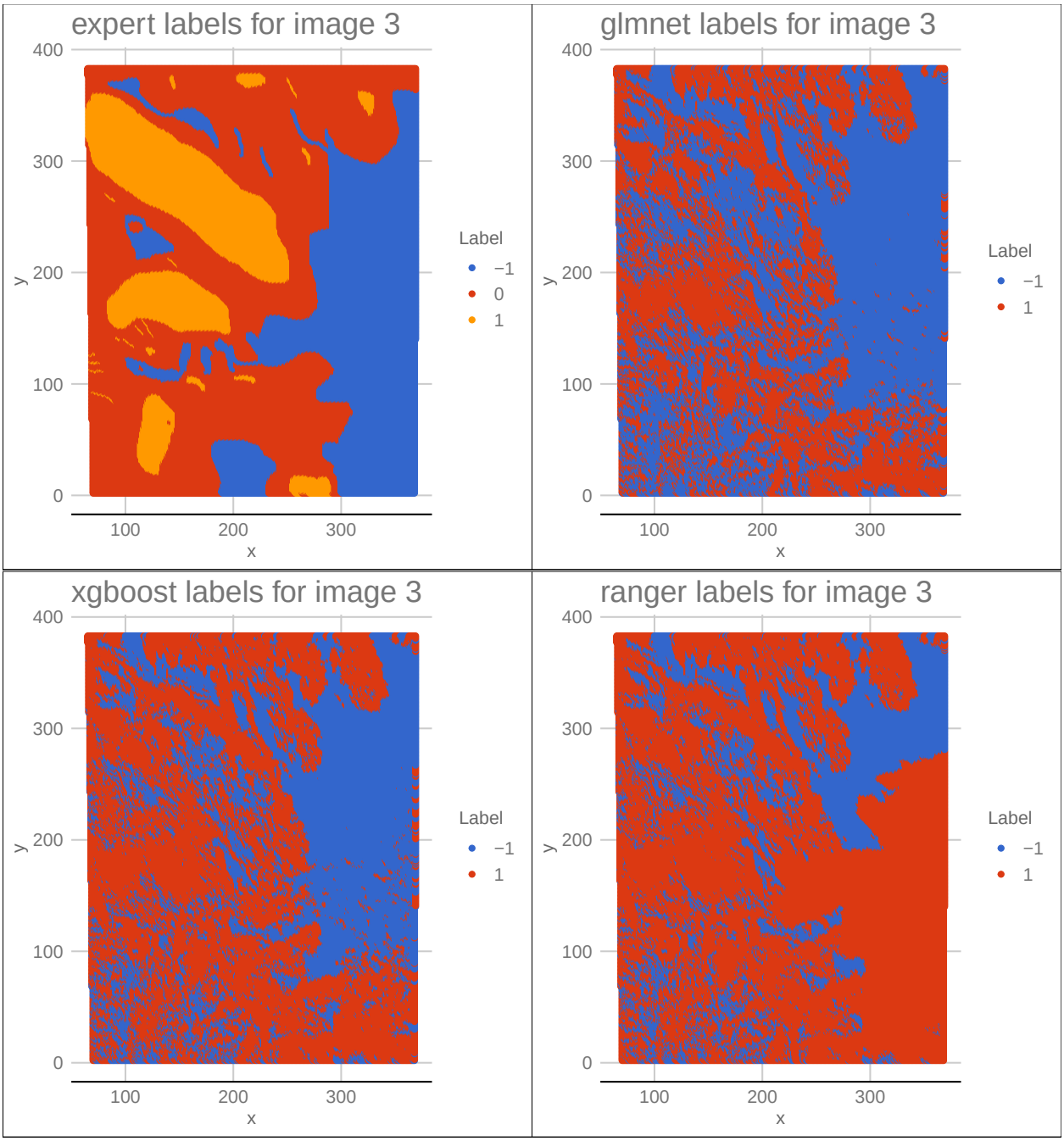

Figure 6: Comparison of expert labels versus classfier predicted labels.