# Chapter 2: Machine learning for snow cover mapping

# Kehan Yang1,2, Aji John1,3, Nicoleta Cristea1,2

1Department of civil and environmental engineering, University of Washington

2eScience Institute, University of Washington

3Department of Biology, University of Washington

# 1 High-resolution snow-covered area maps are needed for Earth science research

Snow is an important component in the Earth's hydrological and energy cycles. Snowpack functions as a natural reservoir that stores water in cold months and slowly releases water into streams, rivers, and lakes, substantially mitigating water stress during the summer. Snow is one of the most reflective land cover types on the Earth’s surface and has positive feedback on climate warming. Temperature increases accelerate snowmelt rate and reduce snow-covered area, SCA, leading to a lower land surface reflectance. More solar radiation will be absorbed by the land surface, which in turn accelerates climate warming. Therefore, to better understand water availability and the Earth’s energy balance, it is important to estimate the spatial and temporal distribution of the snowpack.

Snow-covered area (SCA) is a critical variable for studying land processes such as snow-albedo feedback at different spatial scales, impacts on plant phenology and other associated environmental processes, and is also used in land surface models. While mapping SCA using optical satellite remote sensing imagery is very well established, its implications have been restricted by the tradeoff between spatial and temporal resolution over the past decades. Because mountain snowpack has very high spatial heterogeneity and the SCA can change dramatically in the melting season, it is necessary to have SCA observations at both high spatial and temporal resolution. In this context, the Planet small satellites provide new opportunities for SCA mapping. The Planet is a commercial company that launched about 200 satellites, aiming to provide daily coverage of the entire Earth at about 3-5-meter spatial resolution. All these small satellites can provide observations of visible (i.e., red, green, and blue), and near-infrared bands, where snow surface usually has a very high reflectance compared with other land cover types.

The first generation of Planet satellites was launched in 2016, which is relatively new to users and its application in SCA mapping is rarely studied. Because Planet imagery does not have a shortwave infrared band, which is necessary to distinguish between snow and clouds, it is inapplicable to use the traditional approach (i.e., Normalized Difference Snow Index) to map SCA from Planet images. It becomes even more challenging that only four bands of information are provided which may not be enough to capture the main spectrum features of land cover types in remote sensing classification. In this context, we propose to use machine learning models to map SCA from Planet imagery. Specifically in this chapter, we will demonstrate how we prepare the model inputs, how we tune model parameters and select the best model, and how we evaluate model performance. The objectives of this chapter are to:

1) demonstrate how we can use machine learning models to map SCA from high-resolution satellite imagery.

2) generate a workflow that can be used repeatedly for future applications.

3) develop tutorial materials for educational purposes. All the scripts and datasets are released through the GitHub Repository.

# 2 Machine learning tools

## 2.1 What is ‘scikit-learn’

[Scikit-learn](https://scikit-learn.org/stable/) is one of the most powerful and popular python packages designed to facilitate the use of machine learning. It provides various algorithms used for classification, regression, and clustering. In this section, we will be using the random forest algorithm in the scikit-learn package to map SCA.

## 2.2 What is the random forest algorithm

Random forest is a widely used machine learning algorithm developed by Leo Breiman (Breiman, 2001) and Adele Cutler (Cutler et al., 2012). It is an ensemble of multiple decision trees that are eventually aggregated to get the most likely result. A decision tree is a type of supervised machine learning that bases on a series of questions to categorize or make predictions. Each question forms a tree node that splits the data into different branches. If the answer to the question is ‘yes’, the decision follows the ‘yes’ branch; otherwise, the decision follows the alternate path until it reaches a result. The quality of the results is evaluated by metrics, such as the mean squared error (MSE), Gini Impurity, and information gain.

While the decision trees algorithm is very easy to use, it can be prone to overfitting issues. Using an ensemble of decision trees can largely reduce the overfitting and prediction variance, providing more accurate results. Bagging, also known as bootstrap aggregation, is the most well-known ensemble learning technique, which trains multiple models independently with the training sample set randomly selected with replacement. The final prediction is determined by the average (for regression) or majority (for classification) of all the models. Random forest is an extension of the bagging approach, which generates a random subset of both samples and features for each model training. While a decision tree is based on all features to make decisions, the random forest algorithm only uses a subset of features, which can reduce the influence of highly correlated features in model prediction.

While many other more complex machine learning models exist for land cover classification, for example, mapping SCA using the convolutional neural network (CNN) (Cannistra et al., 2021), we chose to use the random forest model for snow classification as it is a very robust and versatile method to address both classification and regression problems. Additionally, random forest is much less computationally expansive and does not require a GPU to conduct model training, which means we can easily set up the model using our laptop.

# 3 Prepare Input data

The satellite image used in this chapter is provided by the Planet company. Planet company has an [Education and Research Program](https://www.planet.com/markets/education-and-research/) that provides limited, non-commercial access to PlanetScope imagery. The product used in this chapter is the Planet orthorectified product `PS2`, which includes four bands, blue (Band 1, 455-515 nm), green (Band 2, 500-590 nm), red (Band 3, 590-670 nm), and near-infrared (Band4, 780-860 nm). The spatial resolution of the image is 3.7 meters.

As a first step, to visualize the Planet data, we use the following codes to read and display the Planet image over a region in California.



The figure below shows the surface reflectance of the four bands of the PlanetScope image `20180528\_181110\_1025\_3B\_AnalyticMS\_SR\_clip.tif` within a region of the Tuolumne Basin, California. The red and orange colors represent high surface reflectance, while the cyan and dark blue colors represent low surface reflectance. Because snow has a very high reflectance in the visible bands, those red and orange regions are very likely to be covered by snow.

Graphical user interface

Description automatically generated with low confidenceIn the next step, we carefully drew a few ROIs (i.e., Region of Interest) on the image and labeled each ROI as ‘1’ or ‘0’. ‘1’ represents ‘snow’, while ‘0’ represents ‘no-snow’. We label the ROIs based on visual inspection. We only considered the binary classification because the mixing pixel issue is not significant for the Planet image at such a high spatial resolution (3.7 meters), even though it is also not negligible, especially at the edge of snowpack and no-snow land surface. For the demonstration purpose, we will only show the binary classification, “snow” and “no-snow” in this chapter.

We extracted the surface reflectance of all bands of each pixel inside the ROIs and generated an input feature table with 100,000 samples (i.e., ‘sample\_100K.csv’). Here, one pixel is equivalent to one sample. Each sample has four feature columns (blue, green, red, and nir) and one label column (label). In Section 4, we will discuss the influence of sample size on model performance.

Graphical user interface, text, application

Description automatically generated

* Blue: The surface reflectance of the blue band (455-515 nm). Surface reflectance is the fraction of incoming solar radiation that is reflected further from the Earth's surface. It typically ranges from 0 to 1. The original surface reflectance value extracted from the Planet ‘PS2’ product is scaled by 10,000. Here, we have transferred the original values to real surface reflectance as shown in the table.
* Green: The surface reflectance of the green band (500-590 nm).
* Red: The surface reflectance of the red band (590-670 nm).
* Nir: The surface reflectance of the near-infrared band (780-860 nm).
* Label: '0' is no-snow land surface, '1' is snow surface.

# 4 Model parameter tuning

Parameter selection is a very critical step in machine learning model training process. In our experiment, we apply the random forest model to map Planet SCA, and there are several important parameters used in the model training process. To get an optimal set of these parameters as well as the optimal sample size, we conducted a series of sensitivity test of the following four parameters on the overall model accuracy, including the number of samples, the number of features, the number of trees, and tree depth.

Specifically, we will use the ‘RandomForestClassifier’ function from [‘sklearn.ensemble’](https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html?highlight=randomforestclassifier#sklearn.ensemble.RandomForestClassifier) to define the random forest model. The main parameters to customize the model include ‘n\_estimators’, ‘max\_depth’, and ‘max\_features’.

* ‘n\_estimators’ is the number of trees in the forest. Normally, a larger tree number would result in better model performance, but it also means a longer model training time. Additionally, the model performance will stop getting significantly better beyond a critical number of trees.
* ‘max\_features’ is the size of feature subsets used in splitting a node. A good empirical value of max\_features is ‘None’ for regression problems, which considers all features instead of a random subset, and ‘sqrt’ for classification tasks, which uses a random subset of size sqrt(n\_features).
* ‘max\_depth’ is the maximum depth of the tree. A deeper tree has more splits and captures more information about the data. Similar to the number of trees, the model performance will stop getting significantly better once tree depth is deep enough.

First, we read the sample data using the code below. To reduce the calculation time, we select 10,000 samples instead of the entire 100,000 samples as the model performance does not show significant improvement when the sample size reaches around 4000 (section 4.1). We use k-fold cross-validation (k = 10) to evaluate model performance with 100 repeat times.

Graphical user interface, text, application

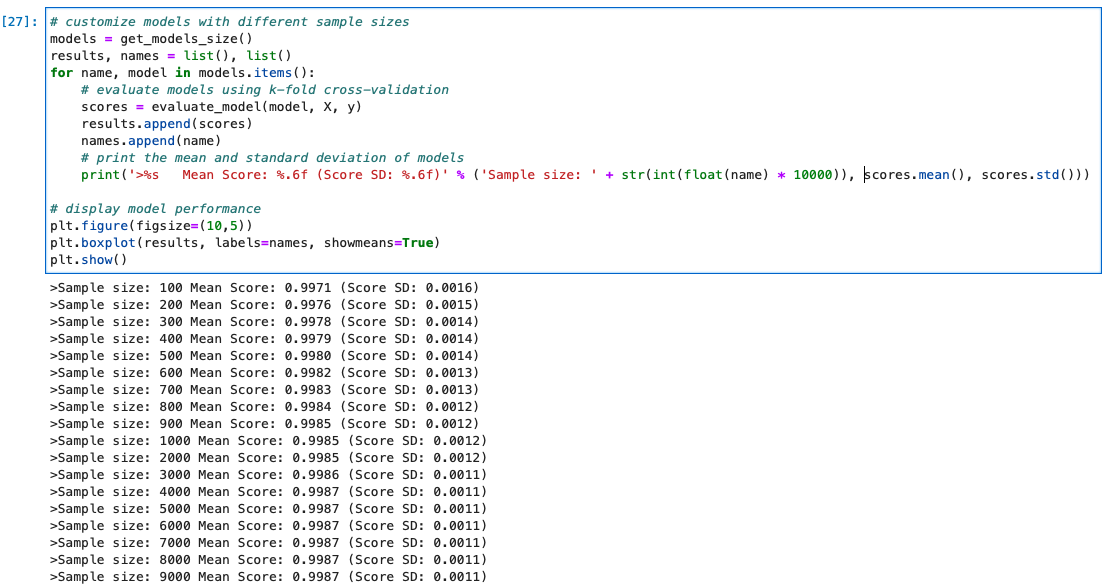
Description automatically generated

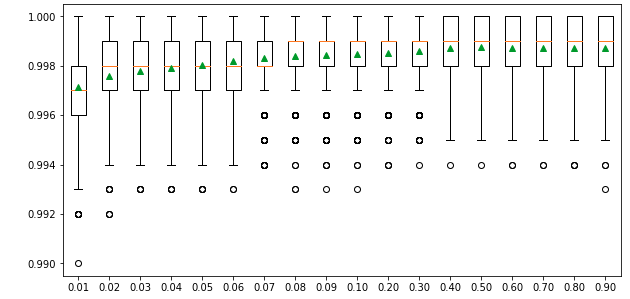
## 4.1. Number of samples

Theoretically, the performance of random forest models can continually improve as the sample size increases. However, training a model with a large sample size would result in a high computational expense, and oftentimes, the model accuracy does not change significantly beyond a critical number of samples.

We generated a custom function ‘get\_model\_size()’ to train random forest models with different sample sizes. We proportionately selected samples by changing the ‘max\_samples’ parameter. The “max\_samples” parameter can be set to a float between 0 and 1 to control the percentage of the training dataset to make the bootstrap sample used in each decision tree training. All custom functions are organized in the ‘functions\_SCA\_mapping.py’ and available to download in the GitHub repository for your interests.

To reduce the calculation time, we only show the percentages ranging from 0.01 to 0.1 with a 0.01 interval, and the percentages ranging from 0.1 to 1.0 with a 0.1 interval in the experiment below. The result shows that the overall model accuracy improves with the increasing sample size when the percentage increases from 0.01 to 0.08 (i.e., 100 to 800 samples), followed by a very slight improvement with the percentage increases from 0.08 to 0.4 (i.e., 800 to 4 k samples). The changes in model performance is negligible after the sample size reaches 4000 (i.e., 0.4 percentage), meaning that the whole dataset can be very well represented by a subset with 4000 observations, and it is not necessary to increase the sample size further. Therefore, we will use 4000 as the optimal sample size to train the SCA model.

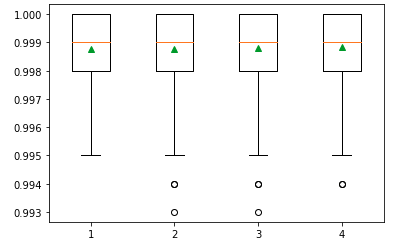
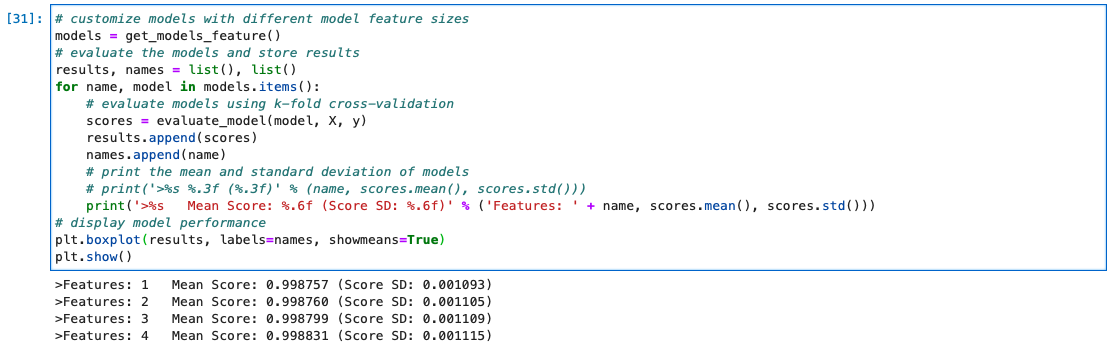




## 4.2 Number of features

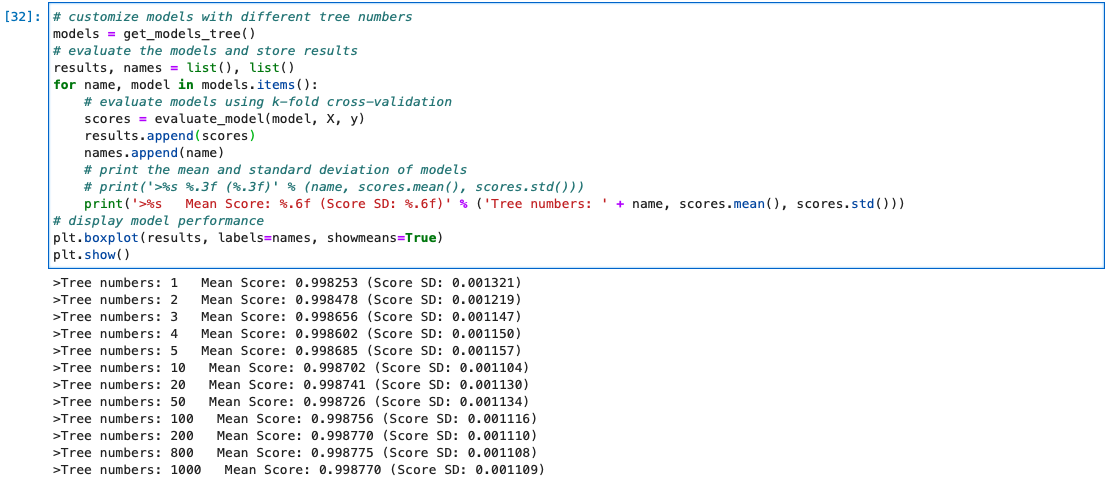
The number of features for each split node is perhaps the most important feature to configure in our random forest model, and it is set via the ‘max\_features’ parameter. To explore the influence of feature numbers on model accuracy, we tested model accuracy with the ‘max\_features’ ranging from 1 to 4. The result shows that the median accuracy (green triangle) of the experiments with max\_features = 4 is slightly higher than the other three max\_features values, though no significant difference is observed among all four test sets.

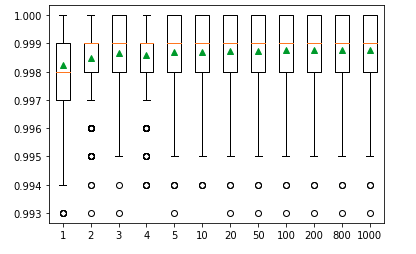
The default of max\_features is the square root of the number of input features. This would be sqrt(4) or two features for our test dataset. However, the total feature size we have is already very small and the model shows slightly better performance when max\_features = 4. Also, we do not want to lose any information from these four bands. Therefore, we decided to use the max\_features = 4 in the SCA model training.



## 4.3 Number of trees

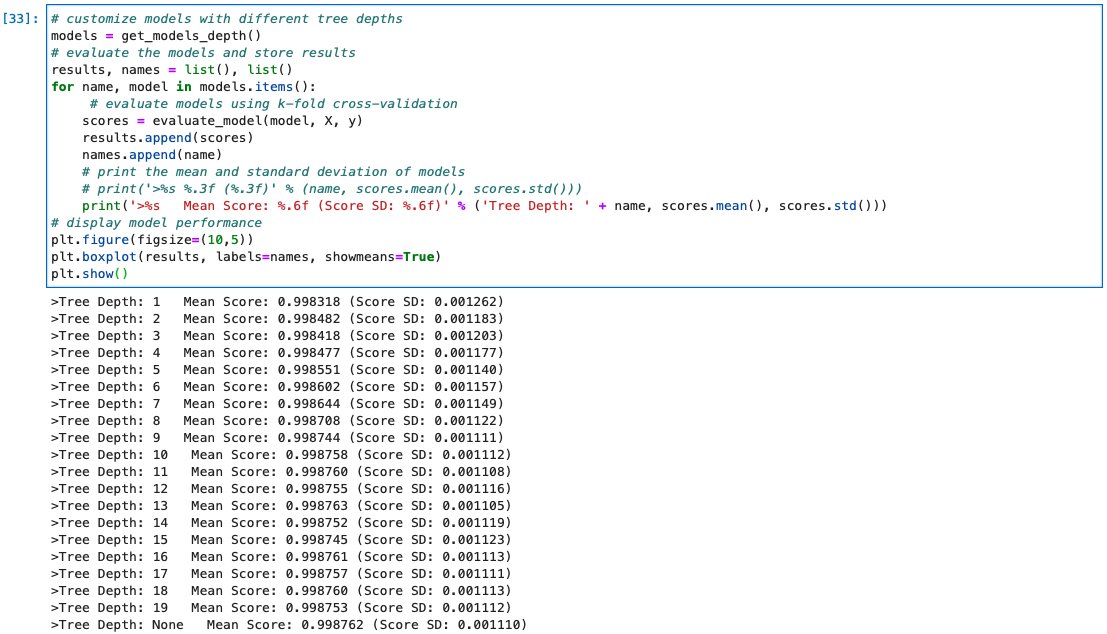
The number of trees is another key parameter to configure in a random forest model. The number of trees can be set via the “n\_estimators'' and the default value is 100. The example below explores the effect of the number of trees on model performance. We set the ‘n\_estimators’ to the values between 1 to 1,000, with only a few selected tree numbers displayed in the boxplot. Typically, when we increase the number of trees, the model performance increases but it will stabilize when we have enough trees. In this case, the change in model performance is negligible when the number of trees reaches 10, and thus we will use n\_estimators = 10 in our SCA model.

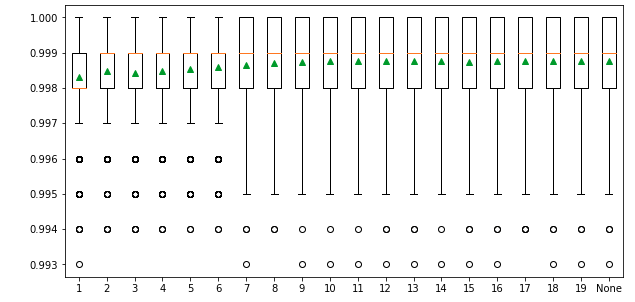




## 4.4 Tree depth

The last parameter is the maximum depth of decision trees and can be set via ‘max\_depth’. Ideally, we would like as many trees as possible to improve model performance, so the ‘max\_depth’ is set to ‘None’ by default, meaning no maximum depth. Reducing tree depth would make the ensemble converge a little earlier, but a large tree depth will lead to a longer computing time. We need a tree depth that is enough to split each node for our samples within an acceptable time. The example below explores the effect of maximum tree depth on model performance. The result shows that the model performance does not have a significant difference when the ‘max\_depth’ is greater than 8 and stabilized when the tree depth is 10. So, we decided to set ‘max\_depth’ to 10 in our SCA model.





# 5 How to train random forest models

With the tests conducted in the last section, we have the optimal values of the four main parameters that we will use in our SCA model. Our next step is to train the random forest model for SCA mapping.

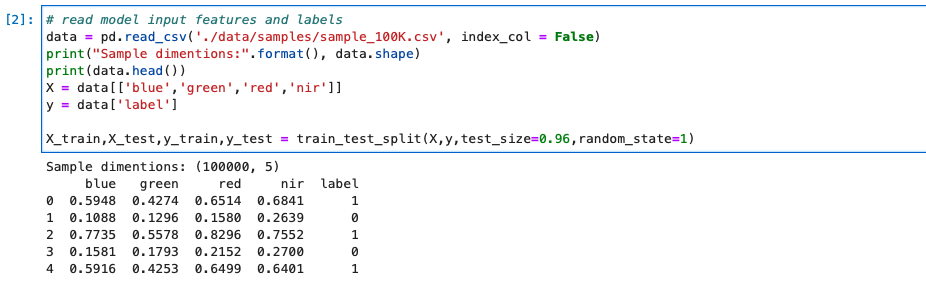
## 5.1 Split data into training and testing subsets

In this step, we split observations into a training subset and a testing subset. Usually, we want to use 70-80% of the data for training, and the remaining 70-80% of the data for testing. However, we find that the model accuracy is reaching a stable stage when the sample size reaches 4000 in the model parameter sensitivity analysis (Section 4). Therefore, we only use 4000 samples, with the remaining 96,000 samples used as an evaluation testing subset to improve model training efficiency.

We use the ‘train\_test\_split’ function from the ‘sklearn.model\_selection’ module to randomly split the sample dataset into training and test subsets.

The parameter ‘test\_size’ can be float or integer. If it is a float, the value should be between 0.0 and 1.0, representing the proportion of the dataset to include in the test split. If it is an integer, it represents the absolute number of the train samples.

The ‘random\_state’ parameter is very useful for making the model sampling reproducible. Here, we assign ‘1’ to ‘random\_state’. It does not matter what the actual ‘random\_state’ number is. It could be any number, but the important thing is that every time we use ‘1’, just the same as the first time we make the split, we will get the same splits, which is very useful for the demonstration.

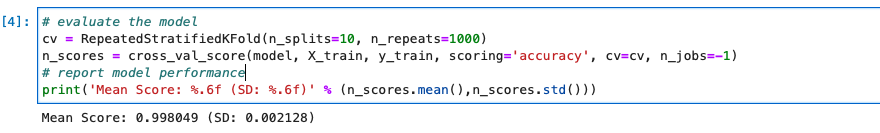


## 5.2 Define the random forest model

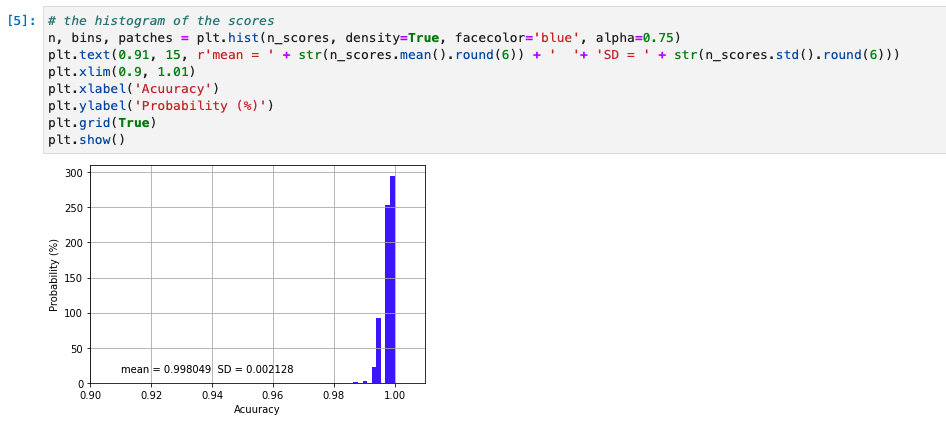
Now, as we have the training subset and the optimal parameters, we can run the ‘RandomForestClassifier()’ to train our model using the code below:



To evaluate the model performance, we conduct K-fold cross-validation using ‘RepeatedStratifiedKFold’ and ‘cross\_val\_score’ from ‘sklearn.model\_selection’. Here, the training subset is randomly split into 10 folds evenly, and each fold is literally used to test the model which is trained by the remaining 9 folds of data. This process is repeated until each fold of the 10 folds has been used as the testing set. The average evaluation metric, here the ‘accuracy’, is used to represent the model performance. This whole process is repeated 1000 times to get the final model performance reported as below:



The overall model training accuracy is 0.998 with 0.002 standard deviation over the 1000 repeated cross-validations, indicating that only 0.2% of samples or pixels on average are incorrectly classified. If we look at the distribution of the accuracy values as shown below, most accuracy values are clustered near 1.00 and all values are higher than 0.98, indicating the model training is very precise and robust.



## 5.3 Feature importance

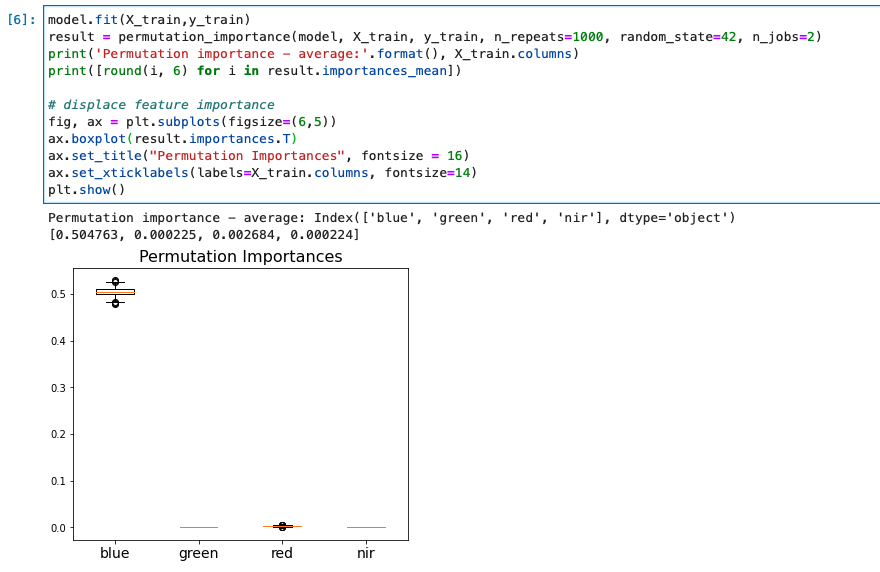
It is important to explore each feature's importance, especially when the feature size is very large, and many features are redundant. In our case, it is not necessary to reduce feature size as we only used four features, but we want to know which band provides the most significant information for the SCA mapping.

We use [‘permutation\_importance’](https://scikit-learn.org/stable/modules/generated/sklearn.inspection.permutation_importance.html#sklearn.inspection.permutation_importance) the form ‘sklearn.inspection’ module to estimate feature importance. The function of the permutation feature importance is described as [below](https://scikit-learn.org/stable/modules/permutation_importance.html#permutation-importance):

‘The permutation feature importance is defined to be the decrease in a model score when a single feature value is randomly shuffled. This procedure breaks the relationship between the feature and the target, thus the drop in the model score is indicative of how much the model depends on the feature. This technique benefits from being model agnostic and can be calculated many times with different permutations of the feature.’

As the random forest algorithm trains the model with randomly selected sample subsets and feature subsets, each model run would have a different estimate of feature importance. Thus, to get a robust estimate of feature importance, we repeated the process 1000 times as shown below.

The result shows that the blue band provides the most important information for SCA mapping, while all the other three bands show much less importance.



## 5.4 Save the model

We now have our model trained and evaluated. We can save the model using the ‘dump()’ function from the ‘joblib’ package as shown below, so that next time when we want to apply this model, we do not have to run through the process mentioned ahead again. In the next section, we will discuss how we load this model and apply it to a satellite image.



# 6 Evaluation of model performance

In the previous sections, we trained a model that can accurately predict SCA within the training samples and the overall model accuracy is 0.998 estimated through the k-fold cross-validation approach. However, we do not know how the model performs outside the training subset. To provide a more comprehensive model evaluation, and especially to test the model transferability, we provide the following two levels of assessment:

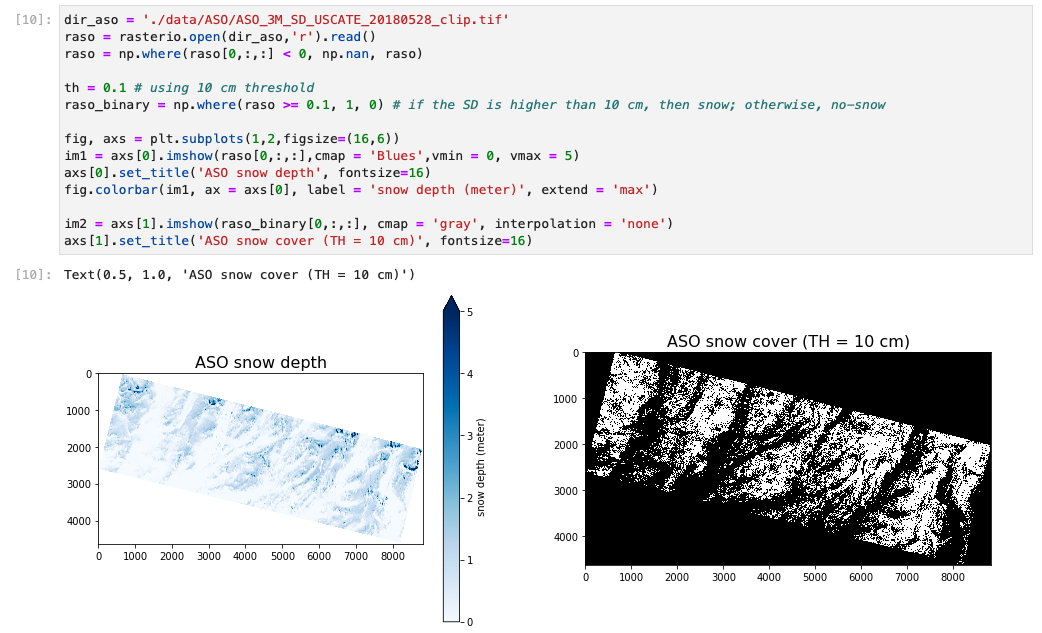
1) As the model training only needs 4000 samples, we have 96,000 samples remaining for evaluation which are geographically located in the same region as the training samples. We can therefore assume that this testing subset has similar spectrum and physiographic features compared with the training subset, and thus the prediction is expected to be accurate.

2) To ensure that we have a robust model, we would also want to know how the model performs over the entire satellite image, and across various land cover types. One challenge we usually face when working with remotely-sensed data is that there is limited high-resolution ground-truth datasets we can use as validation datasets. We have the same issue. Previous study has applied high-resolution lidar-derived SCA data as ‘ground truth’ to evaluate snow cover (Cannistra et al., 2021) given the relatively high accuracy of lidar snow depth measurement. Here, we chose to use the same dataset for SCA evaluation. Specifically, we will use the lidar-derived high-resolution snow depth data at 3-m spatial resolution provided by the Airborne Snow Observatory (ASO, Painter et al., 2016) to map SCA. The lidar snow depth dataset does not provide direct SCA information, so we will use a threshold on the snow depth data to calculate SCA.

The ASO conducts airborne surveys for some selected watersheds in California and Colorado. The aircraft carries a lidar sensor to map snow depth based on the elevation difference between snow-on and snow-off surfaces. The uncertainty of the final snow depth product at the 3-meter resolution is unbiased with a root mean squared error (RMSE) of 8 cm (Painter et al., 2016).

A previous study by Cannistra et al., 2021 used 10 cm as a threshold to convert the ASO 3-meter snow depth data into a binary SCA map. If the snow depth is deeper than 10 cm, the pixel is classified as a snow pixel; otherwise, the pixel is a no-snow pixel. While using different thresholds will result in different SCA maps, we find that using a threshold between 8-10 cm gives the best agreement between planet SCA and ASO snow depth derived SCA (John et al., 2022, in review). So, we follow Cannistra et al., 2021 and apply 10 cm to derive SCA from ASO snow depth.

The left figure below shows the spatial distribution of snow depth for the study domain, and the right figure shows the distribution of binary snow-covered areas.



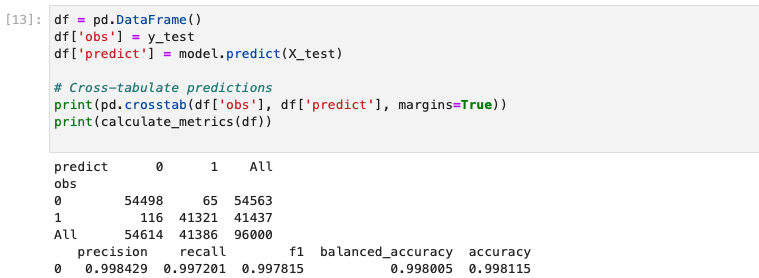
The following five evaluation metrics are used to discribe model prediction accuracy:

Where TP, TN, FP, FN are true positive, true negative, false positive, and false negative, respectively.

## 6.1 Testing subset model performance

We run the ‘model.predict()’ to get the snow cover prediction, where the ‘model’ is the SCA mapping model we trained in previous sections. We can use the code below to load the saved model from the directory:

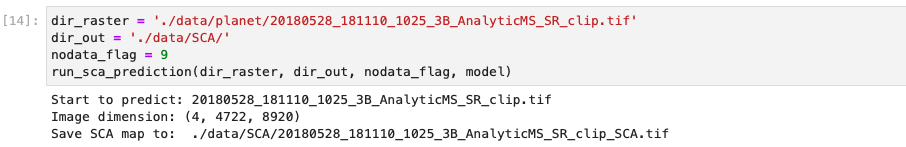


Then, we calculate the five evaluation metrics using ‘calculate\_metrics()’ which is a custom function we generated to evaluate model performance. 

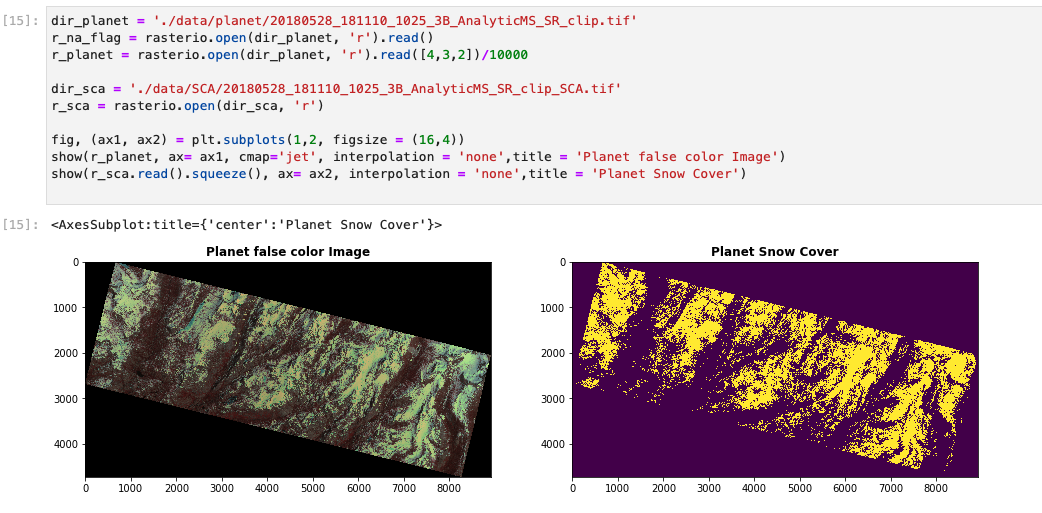
From the results printed out above, the false negative is 116 within 54614 no-snow pixels, and the false positive is 65 within 41386 snow pixels. All five evaluation metrics are very close to 1, indicating the model performs very well in the remaining 96,000 testing samples, which is not very surprising as the pixels within the testing subset are located close to the training samples and also have similar spectrum features.

## 6.2 Image wide model performance

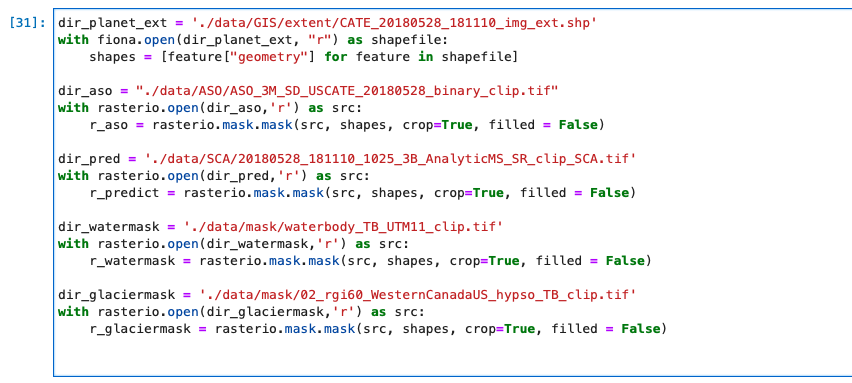
Now, let’s look at how the model performs across the entire image using the following steps. Firstly, we collect the four bands’ surface reflectance over the entire image, and then we apply a custom python function ‘run\_sca\_prediction()’ to predict SCA. Finally, we save the SCA image to ‘dir\_our’ directory.

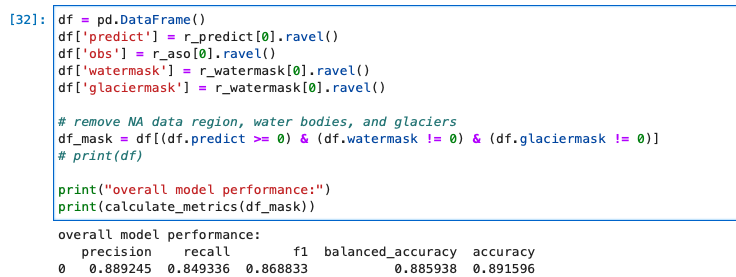


We display the original Planet false-color image (left) and the predicted SCA map (right) in the figure below. Based on our visual examination, the model predicts SCA precisely and captures the spatial distribution of snow very well. Next, we will compare this Planet SCA with the validation data set – ASO SCA.



To compare the ASO SCA and Planet SCA to the same extent, we firstly read data within the same extent as shown by the yellow outline in the figure below. Because the ASO snow depth product should exclude water bodies and glaciers, we apply the waterbody dataset (data downloaded from [NHDPlus](https://www.epa.gov/waterdata/nhdplus-national-hydrography-dataset-plus#:~:text=National%20Hydrography%20Dataset%20Plus%20(NHDPlus,with%20the%20U.S.%20%20Geological%20Survey)) and glacier dataset (data downloaded from [Randolph Glacier Inventory 6.0](https://www.glims.org/RGI/rgi60_dl.html)) to mask out those areas.

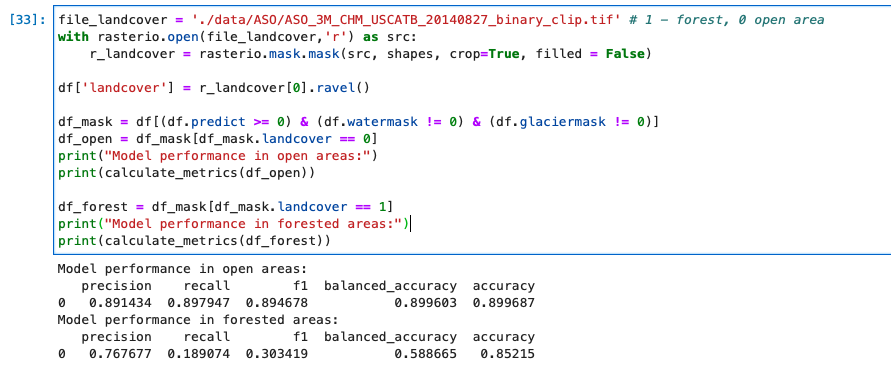




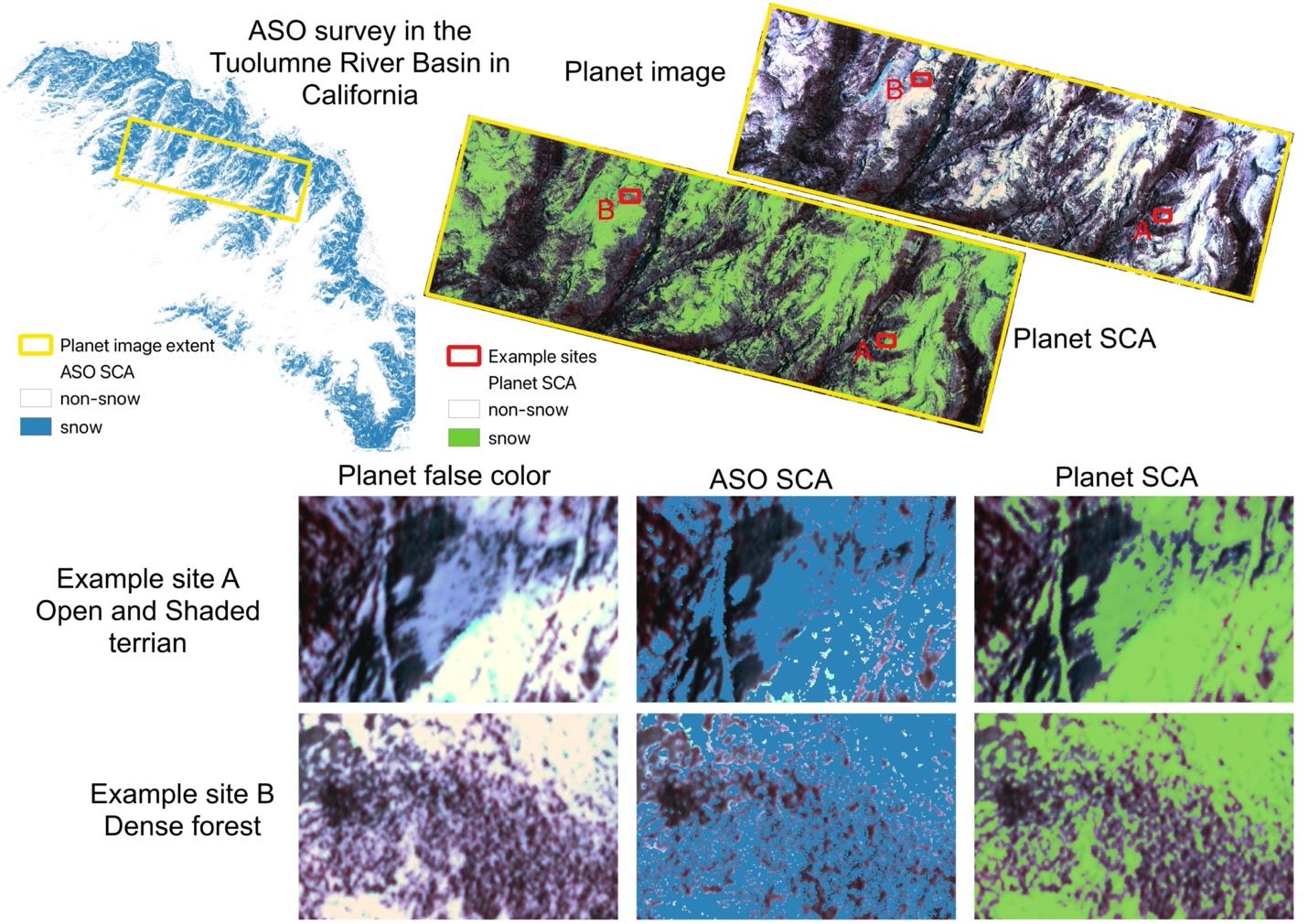
The result shows that, overall, 90% of the pixels are classified accurately with the f1 value of 0.87. The model has fewer false-positive predictions than false-negative predictions as the precision value (0.89) is slightly higher than the recall value (0.85), indicating a small overall underestimation in SCA modeling.

## 6.3 Model performance in open versus forested areas

To explore the potential reasons that explain the mismatch between ASO SCA and Planet SCA, we divide the entire domain into two categories: open areas and forested areas. We use a 3-meter canopy height model dataset provided by ASO. Inc to classify the domain into open and forested areas. If the pixel has a tree height value higher than 1 meter, then this pixel is classified as forest; otherwise, the pixel is classified as open.



The results show a difference in model accuracy between open and forested areas. For the open areas, the overall model accuracy is 90%, with very similar precision (0.89) and recall (0.90) values, indicating similar false positive and false negative predictions. However, the overall model accuracy for the forested area is only 85%, with relatively high precision (0.77) and an extremely low recall (0.19), indicating much higher false-negative predictions than false positive. The main reason for the high false-negative in forest areas is that Planet uses optical sensors which cannot penetrate canopy cover to get underneath snow cover information while ASO uses a lidar sensor that can penetrate the canopy.

To get a close look at the difference between ASO SCA and Planet SCA, we select two example sites: A and B as shown in the figure below. Site A is in open terrain with half of the area in a shaded valley and Site B is in a dense forest. 

The model accurately predicts Planet SCA for the open areas in Site A. Even for the shaded terrain where snow and all other land surfaces have low reflectance, the model precisely captured the snow cover with only a small underestimation over the northwest corner. ASO estimates larger SCA in the shaded areas than Planet, but it slightly underestimates SCA in open areas over the southeast corner.

For Site B, ASO and Planet show significant differences over dense forests. Planet well captures SCA over forest gaps although it shows slight underestimation along forest edges where the mixed pixel issue is not negligible, which is in line with our findings discussed in previous subsections.

# 7. Conclusion

In this chapter, we develop a random forest model to map snow-covered areas (SCA) from the high-resolution Planet imagery. We demonstrate the influence of four main parameters on model performance, including the number of samples, the number of features, the number of trees, and the tree depth. The optimal parameters determined by these tests are used in the SCA model generation.

The final model shows very precise performance in predicting binary SCA with overall model accuracy of 1.00 and 0.94 at 3-meter spatial resolution when evaluated by the testing subset and the SCA derived from Airborne Snow Observatory (ASO) snow depth dataset, respectively. The model predicted Planet SCA exhibits much higher accuracy for open areas than forested areas when evaluated by the ASO derived SCA, with an overall accuracy of 0.95 and 0.85, respectively. While the uncertainties in both SCA datasets are not negligible, the main mismatch between Planet SCA and the ASO SCA is caused by the difference in observation sensors. ASO SCA relies on a lidar sensor that can penetrate the canopy and ‘see’ snow under canopy, while Planet uses optical sensors that only receive surface reflectance above the canopy.

In summary, the SCA mapping model we demonstrated in this chapter is very feasible to set up and use. As we only used four bands of information in the model training, this method can be applied to most existing satellite observations like Landsat series imagery, Sentinel-2, and MODIS. The workflow generated in this chapter has critical implications for hydrological process monitoring and prediction, particularly in a world with more dramatic changes in snow cover.

# Reference

Breiman, L., 2001. Random Forests. Machine Learning 45, 5–32. <https://doi.org/10.1023/A:1010933404324>

Cannistra, A.F., Shean, D.E., Cristea, N.C., 2021. High-resolution CubeSat imagery and machine learning for detailed snow-covered area. Remote Sensing of Environment 258, 112399. <https://doi.org/10.1016/j.rse.2021.112399>

Cutler, A., Cutler, D.R., Stevens, J.R., 2012. Random forests, in: Ensemble Machine Learning. Springer, pp. 157–175.

John, A., Cannistra, A., Yang, K., Tan, A., Shean, D., HilleRisLambers, J., Cristea, N., 2022, High-resolution snow-covered area mapping in forested mountain ecosystems using PlanetScope imagery. Remote Sensing (In review)

Painter, T.H., Berisford, D.F., Boardman, J.W., Bormann, K.J., Deems, J.S., Gehrke, F., Hedrick, A., Joyce, M., Laidlaw, R., Marks, D., Mattmann, C., McGurk, B., Ramirez, P., Richardson, M., Skiles, S.M., Seidel, F.C., Winstral, A., 2016. The Airborne Snow Observatory: Fusion of scanning lidar, imaging spectrometer, and physically-based modeling for mapping snow water equivalent and snow albedo. Remote Sensing of Environment 184, 139–152. <https://doi.org/10.1016/j.rse.2016.06.018>