Coastal mapping in Greenland

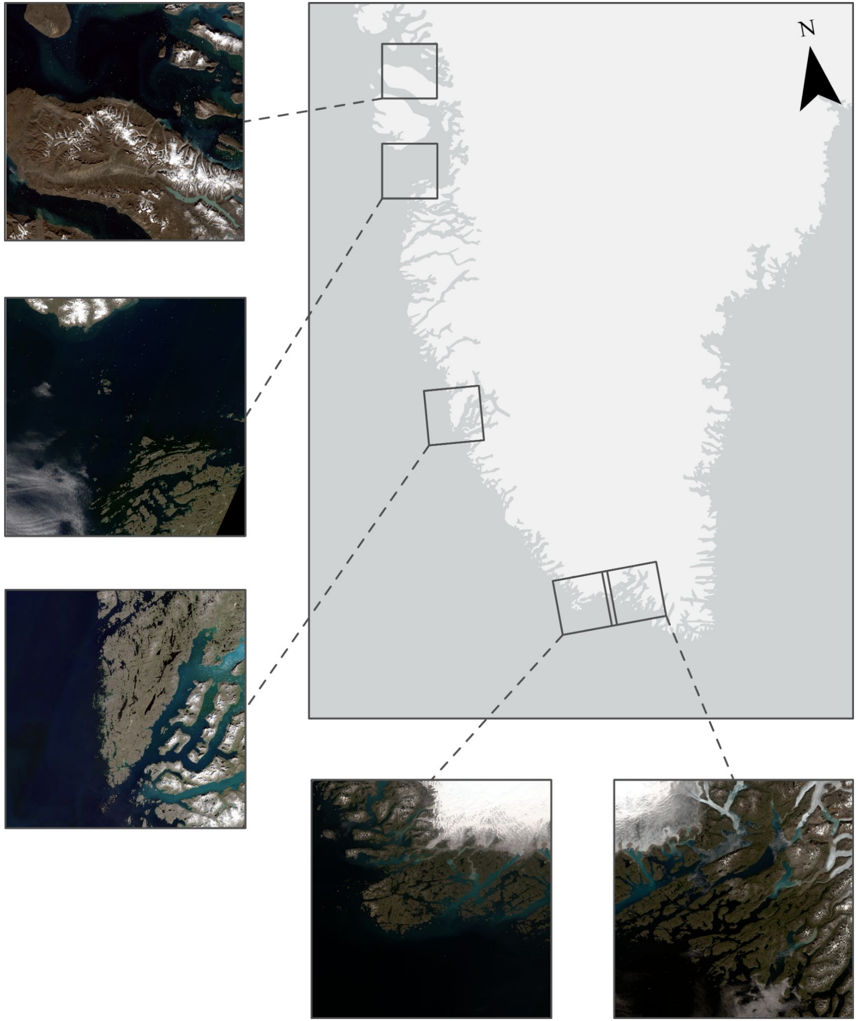
Final project

Project background

- Increase in maritime activity near Arctis
- Inherent navigation risks
- Lack of frequently updated and precise maps

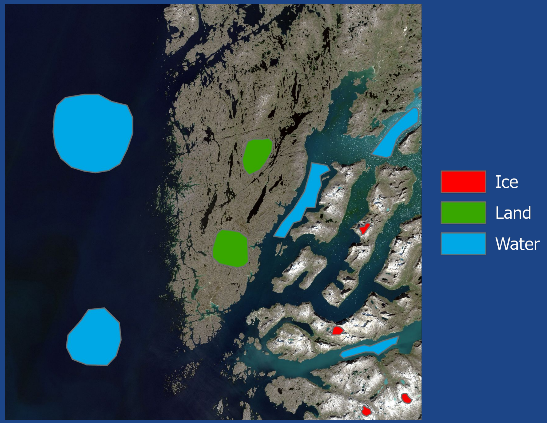
- Improving the map database of Greenland’s coastal areas
  - High-resolution satellite imagery
  - Finding the best model

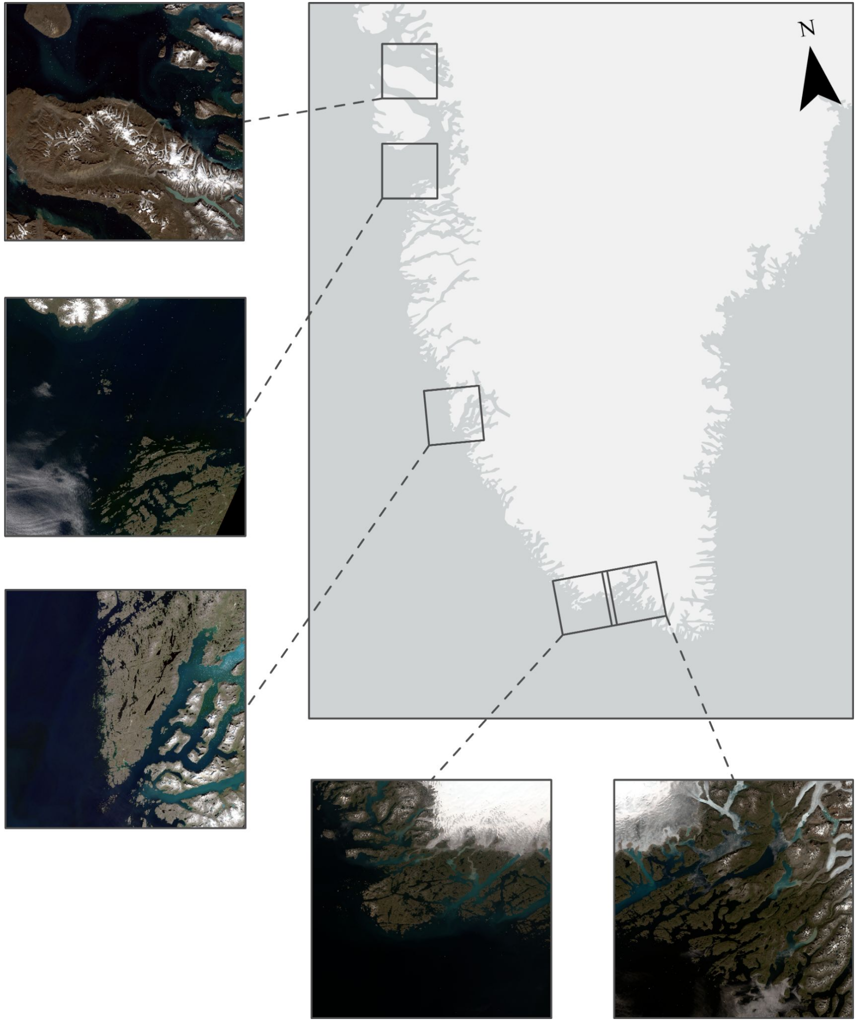
- Trying to beat base model accuracy of 99,03% (LightGBM)



Data description

- Sentinel-2 imagery (10x10m)
- 16 features
  - Pre-calculated convolutional layers (30x30) & (50x50)
- 78 mill observations
- Data collected by drawing polygons





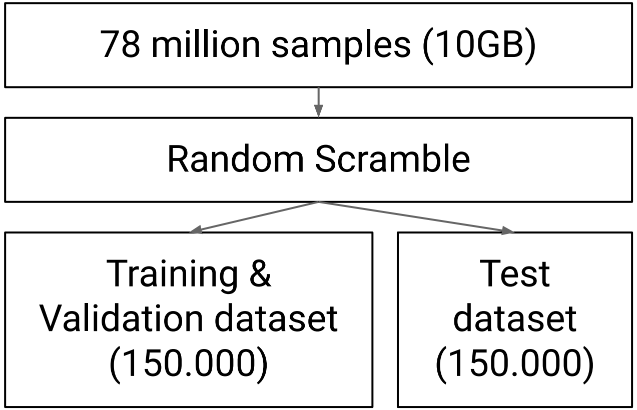
Data preprocessing

Initially we had 10 GB of data. We wish to train and test our models on a smaller subset.

Data preprocessing was done in 2 steps:

1) Random scramble full dataset of 78 million samples.

2) Subset 150.000 samples each for Train and Test dataset. 50.000 samples were selected from each of the 3 classes to reduce bias.



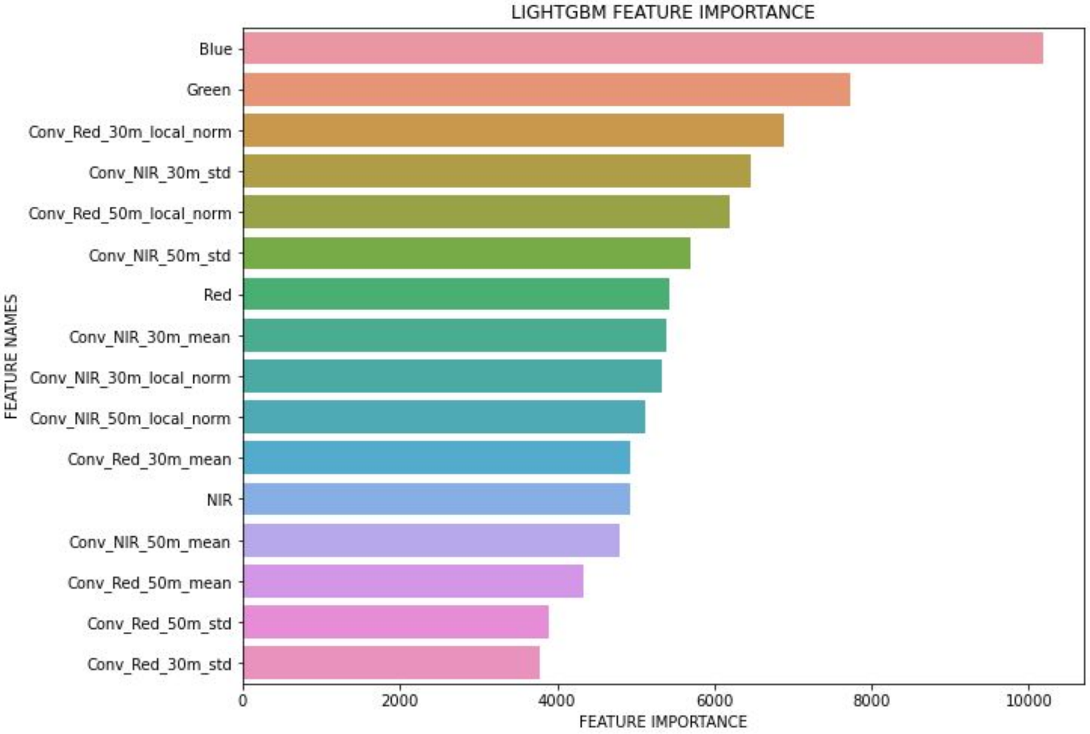
78 million samples (10GB)

Random Scramble

Training & Validation dataset (150.000)

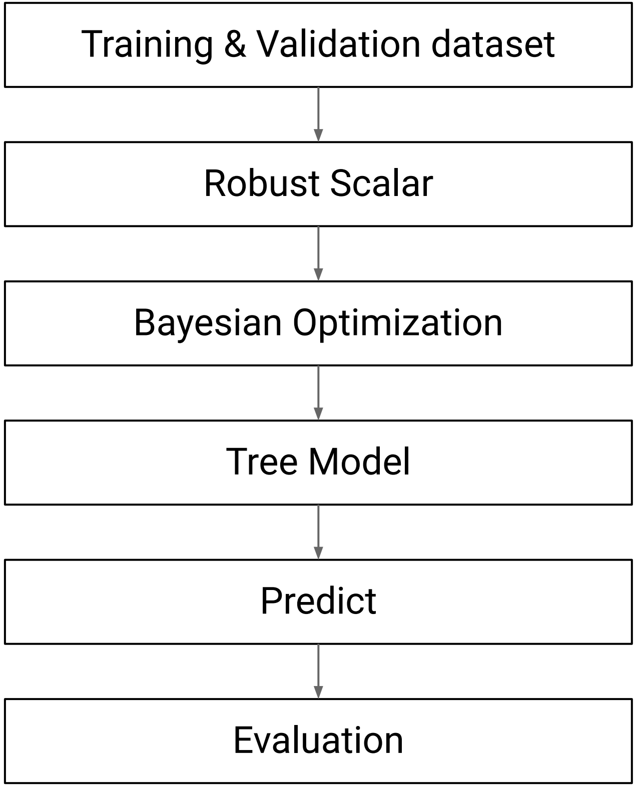
Test dataset (150.000)

LightGBM feature importance



Choice of tree models & methods

- XGBoost - Popular gradient boosting tree model
- LightGBM - Faster training speed and higher efficiency than XGBoost
- RandomForest - Very popular in Earth Observation due to its unbiased bagging approach.



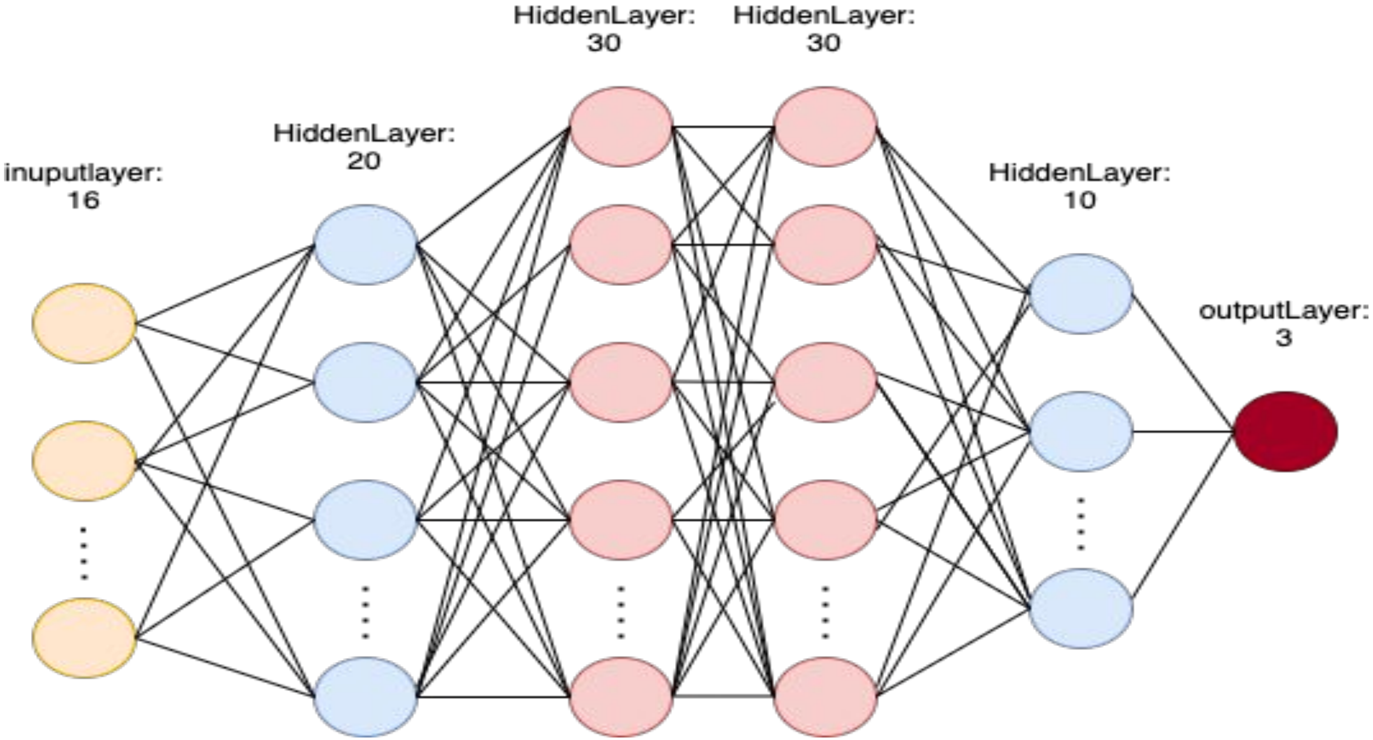
Biased model

- Handling navigational risks
- One would rather classify water as land, than land as water
- Minimizing land or water classified as water

- Implemented as a Sklearn-DecisionTreeClassifier.
  - AdaBoost can take a DTC as input. The DTC setup was easy to reuse
- Pointing the model away from one type of error achieved through:
  - Custom scorer of hyperparameters using Gridsearch
  - Weighting of classes
  - Land was most frequently classified as water. Weighted accordingly
- Best score: 99.71% certainty that classification of water is actually water
- Overall precision is 97.33%

NN classification model

Feed forward neural network



NN classification model

3 classes

- Optimization - Adam
- Loss - Cross Entropy Loss
- Activation functions for each layer - ReLU
- Learning rate: 0.001

Tuning hyperparameters

- Trial and error
- Adding small dropout

Lowest loss model vs. fully trained model

Unsupervised methods

Normalization of data:

- No normalization
- Standard normalization
- Robust normalization

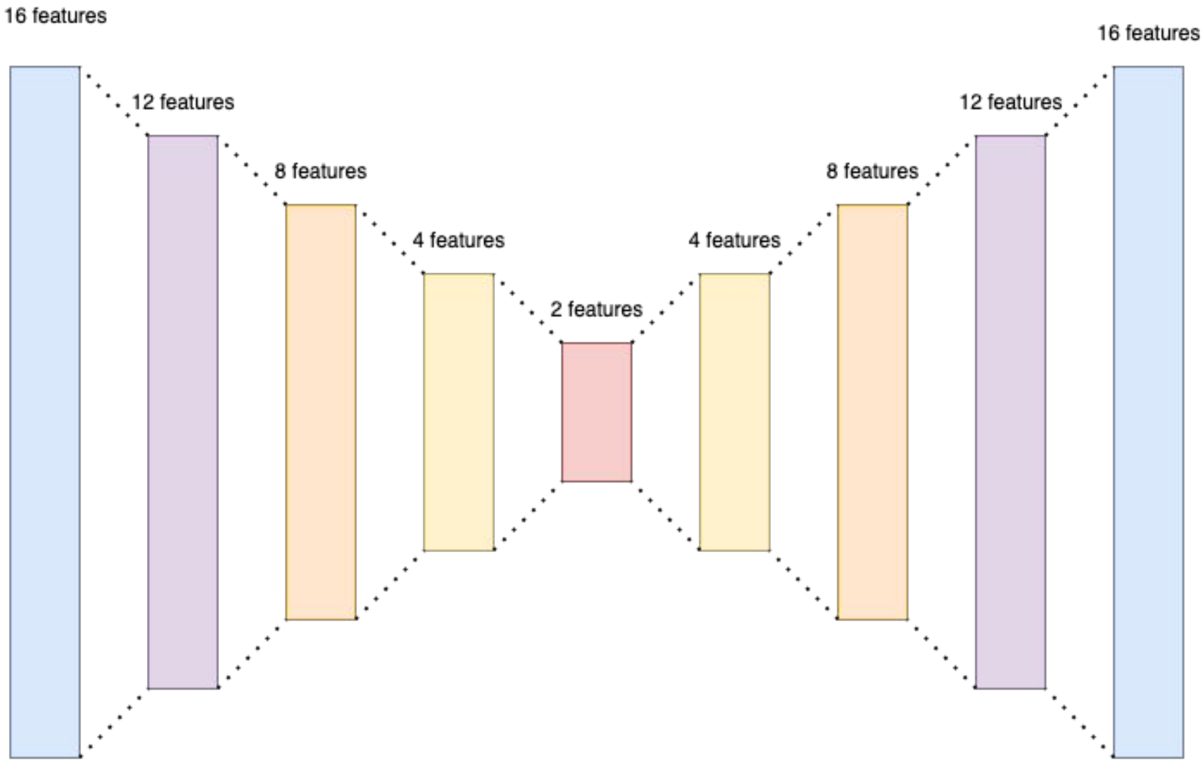
Dimensionality reduction

- Principle component analysis (PCA)
- Auto Encoder (AE)

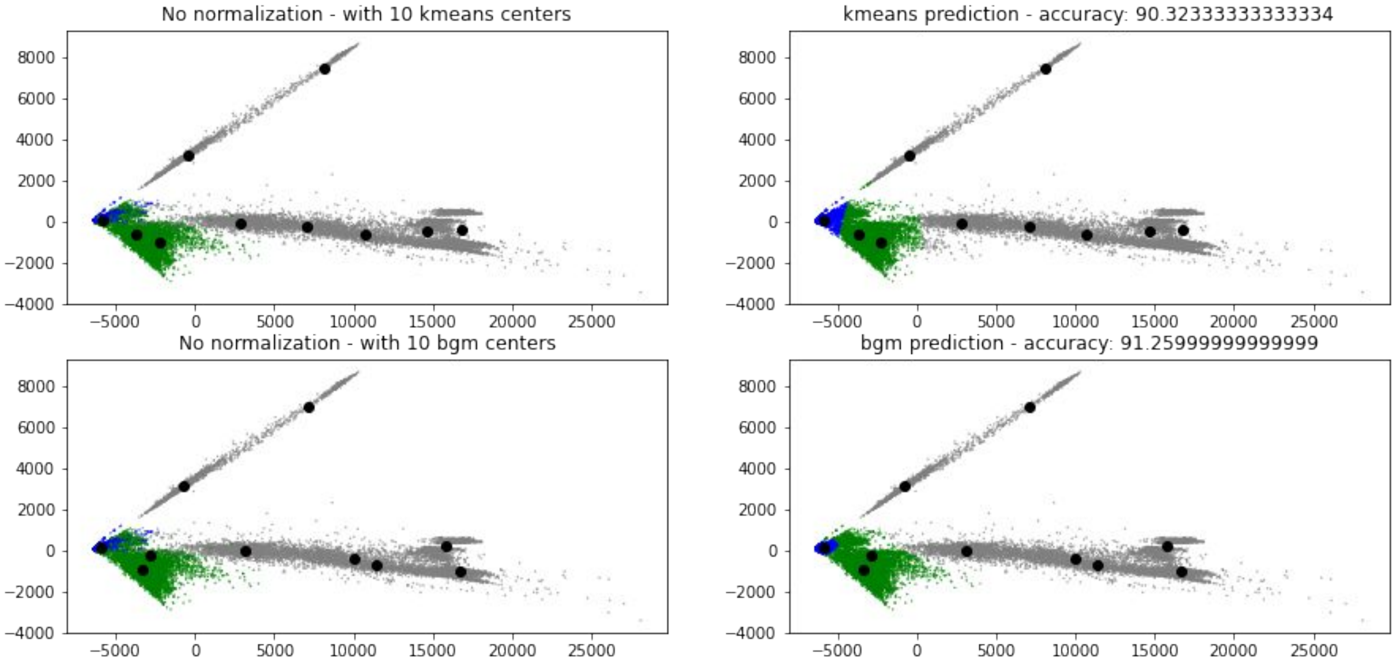
Clustering

- K means
- Bayesian Gaussian Mixture

Auto Encoder neural network



Unsupervised learning results (PCA)



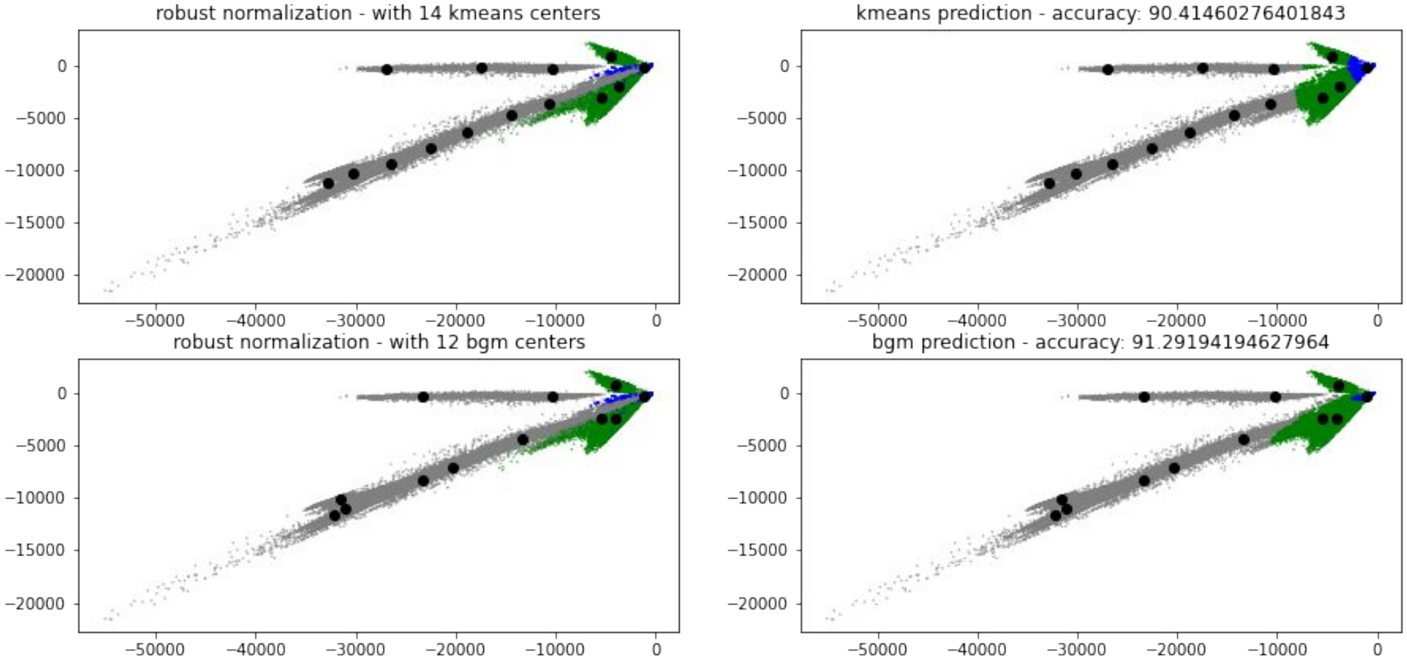
No normalization - with 10 kmeans centers

kmeans prediction - accuracy: 90.32333333333334

No normalization - with 10 bgm centers

bgm prediction - accuracy: 91.25999999999999

Unsupervised learning results (AE)



robust normalization - with 14 kmeans centers

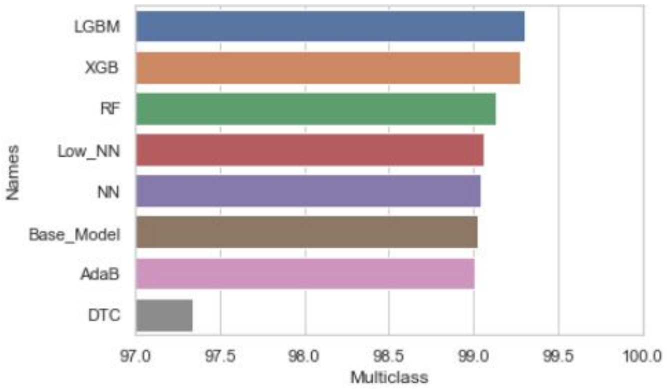
kmeans prediction - accuracy: 90.41460276401843

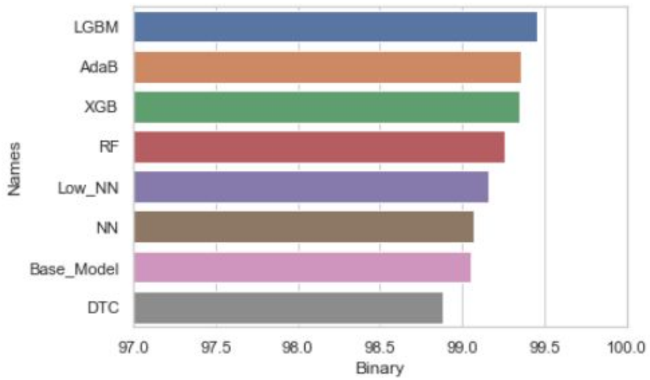
robust normalization - with 12 bgm centers

bgm prediction - accuracy: 91.29194194627964

Results

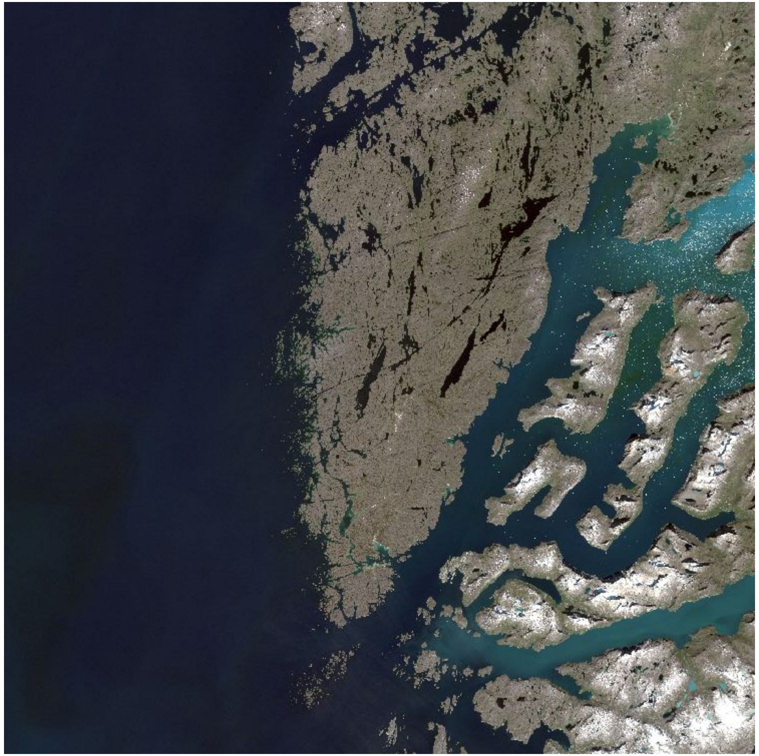
- All measured on the 150.000 sample test-set

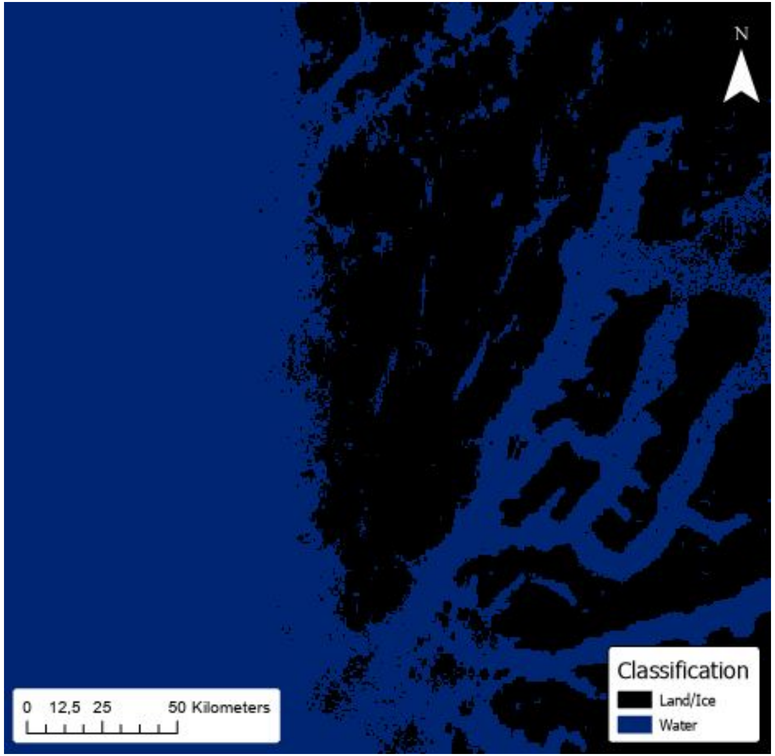




Model	LGBM	XGB	RF	DTC	AdaB	Low_NN	NN	SVM
Precision Multiclass	99.30%	99.28%	99.13%	97.34%	99.01%	99.06%	99.04%	Still running....
Precision Binary	99.46%	99.35%	99.26%	98.88%	99.36%	99.16%	99.07%	Kernel died...

Best prediction mapped: LightGBM on binary classes



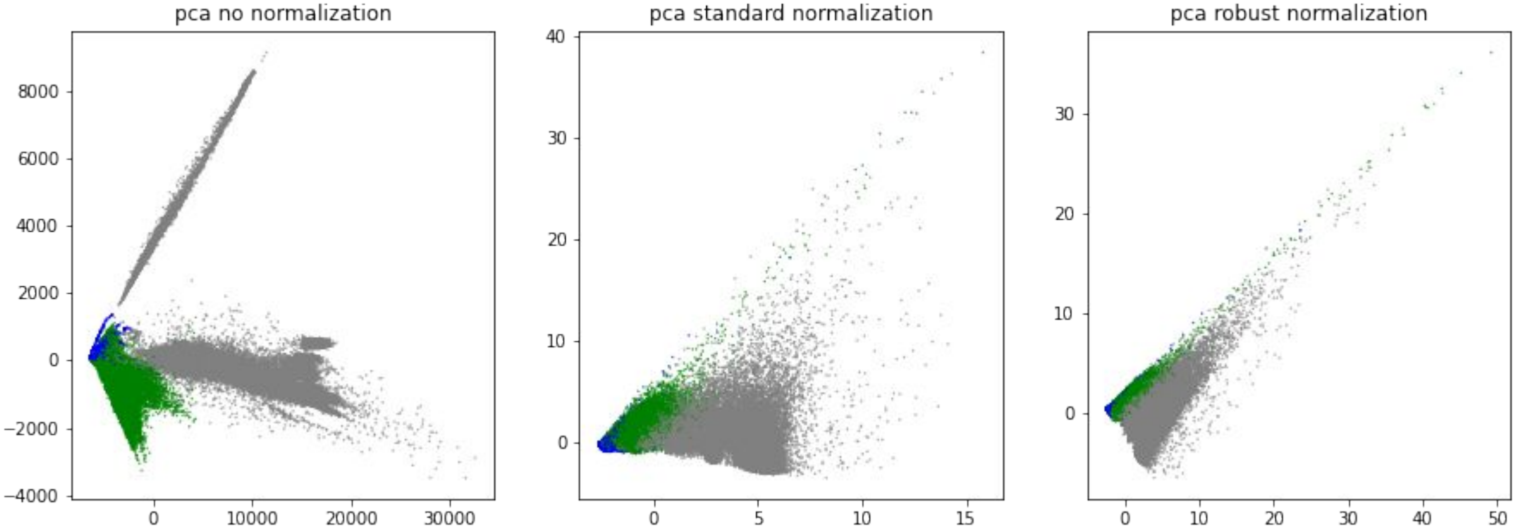


Final thoughts

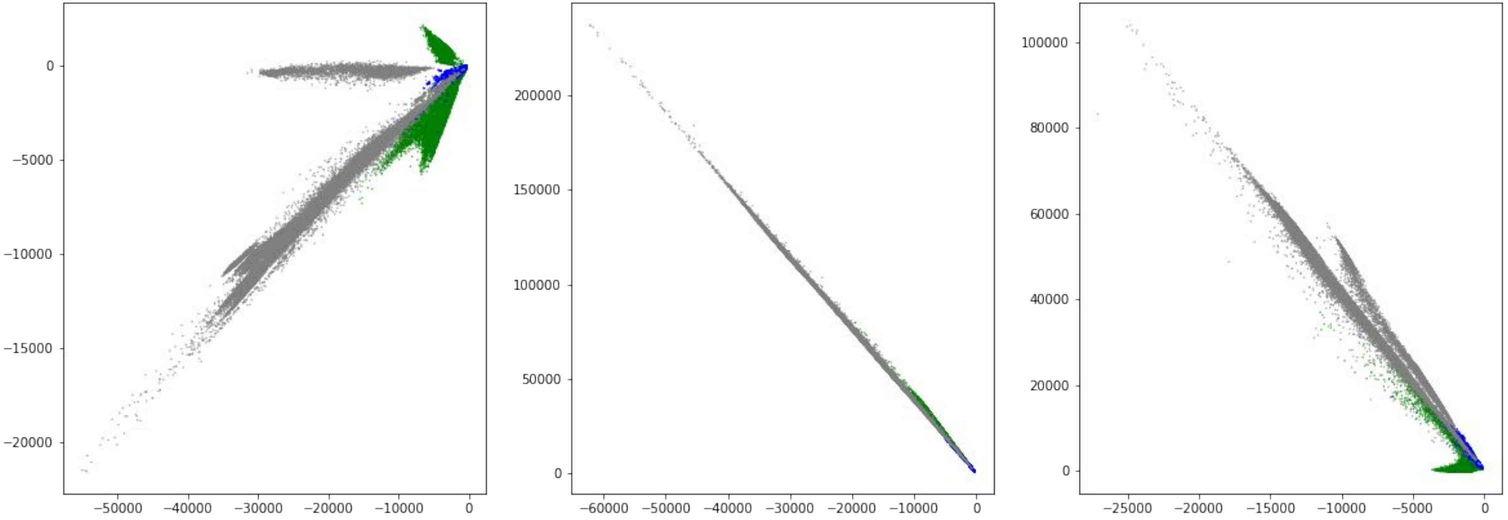
- Using Machine learning on full images
  - Convolutional Neural network
  - Other image processing techniques
- When precision is high, minor improvements are still impactful.
- Adapting to the nature of the given problem.

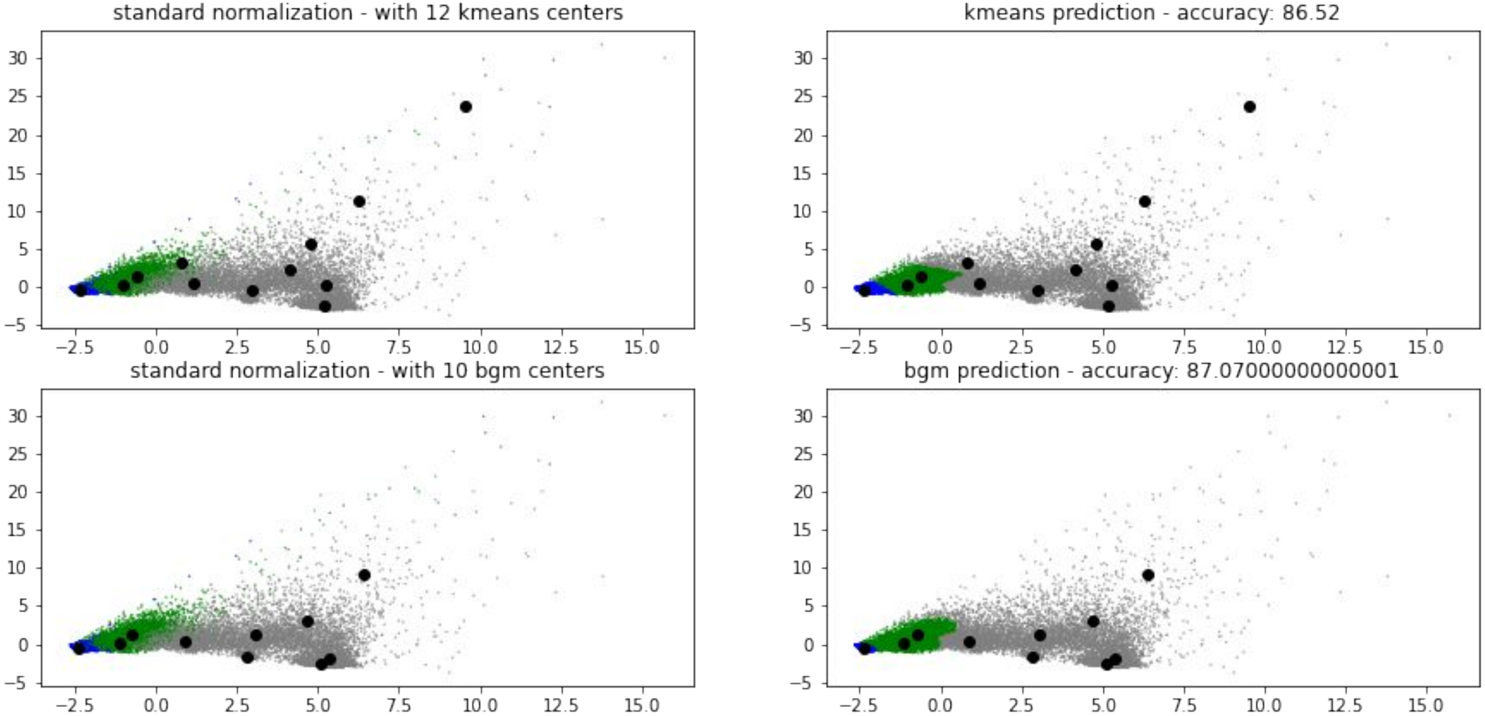
Thanks!

Unsupervised learning results (PCA)



Unsupervised learning results (AE)



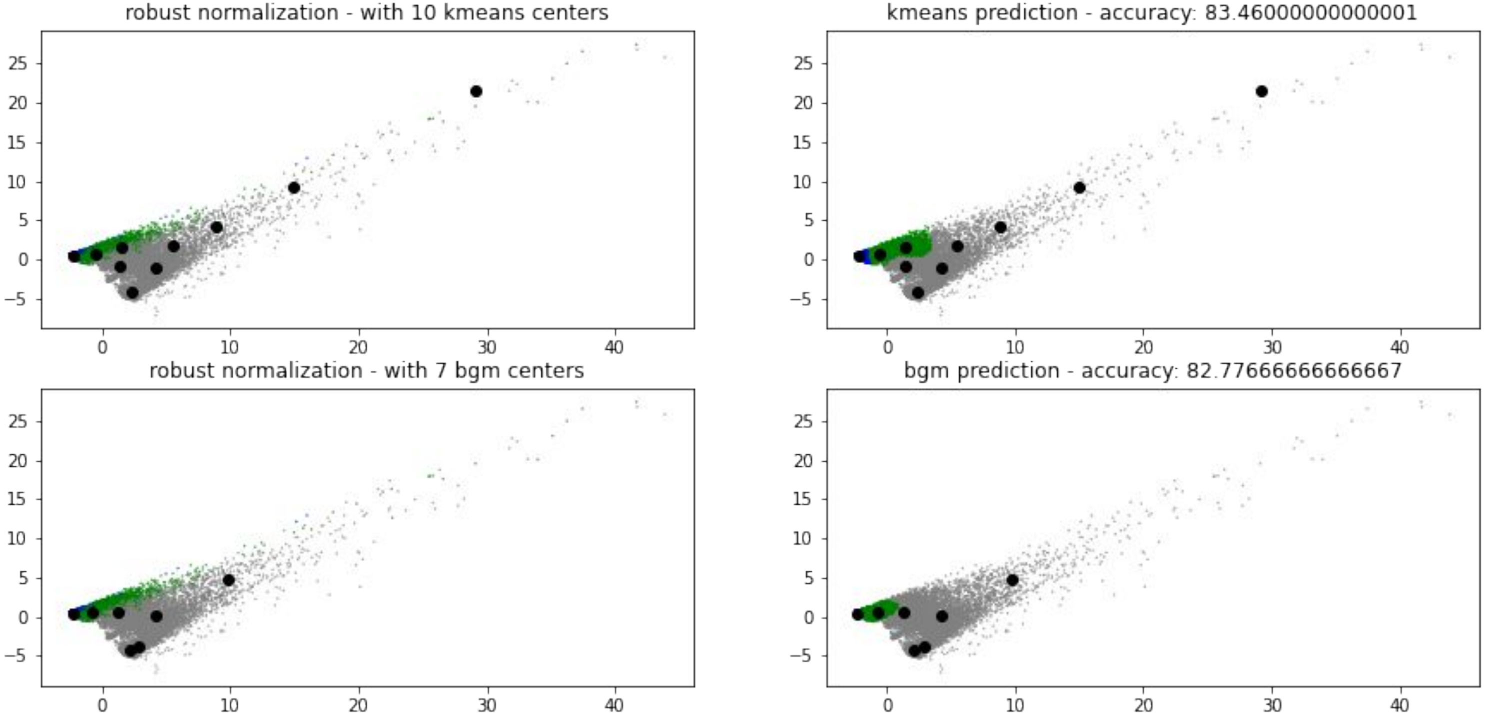


standard normalization - with 12 kmeans centers

kmeans prediction - accuracy: 86.52

standard normalization - with 10 bgm centers

bgm prediction - accuracy: 87.07000000000001

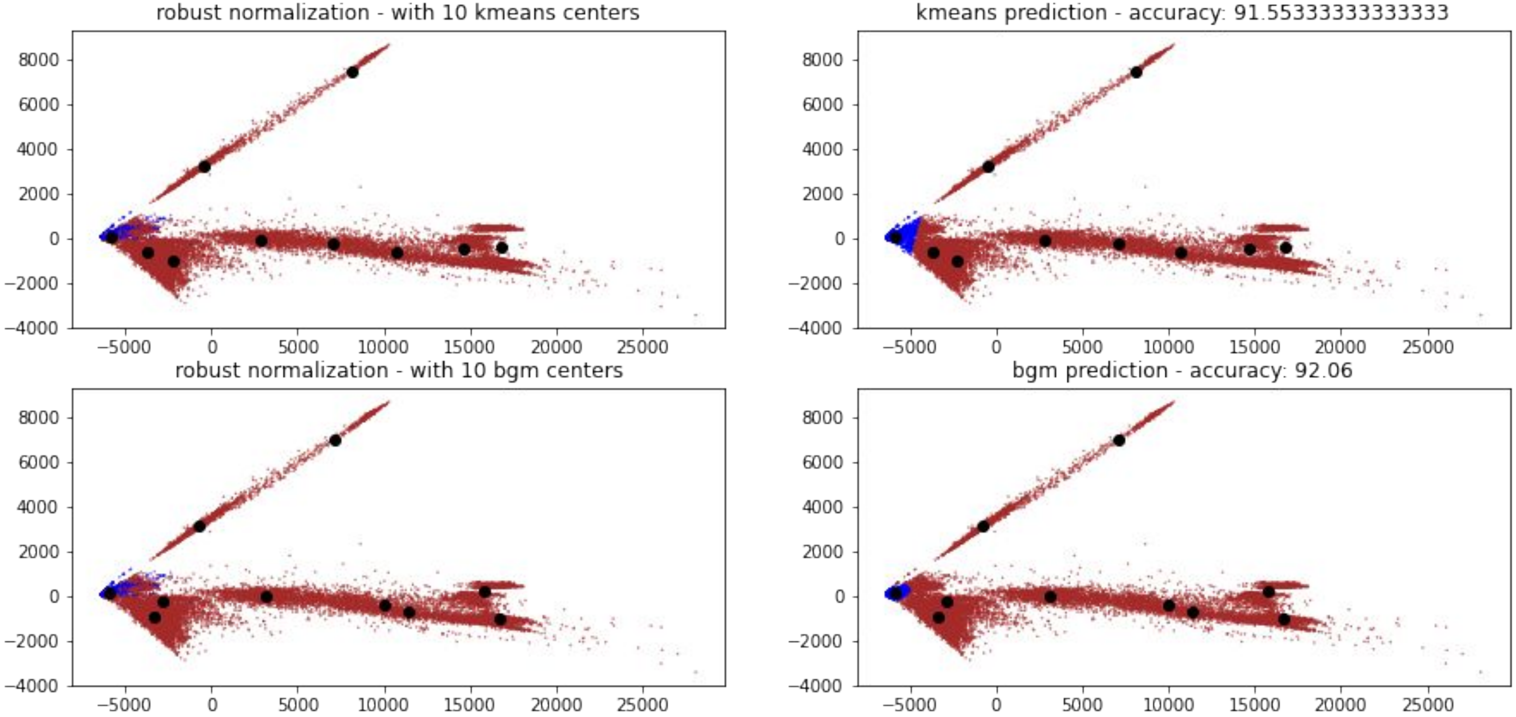


robust normalization - with 10 kmeans centers

kmeans prediction - accuracy: 83.46000000000001

robust normalization - with 7 bgm centers

bgm prediction - accuracy: 82.77666666666667

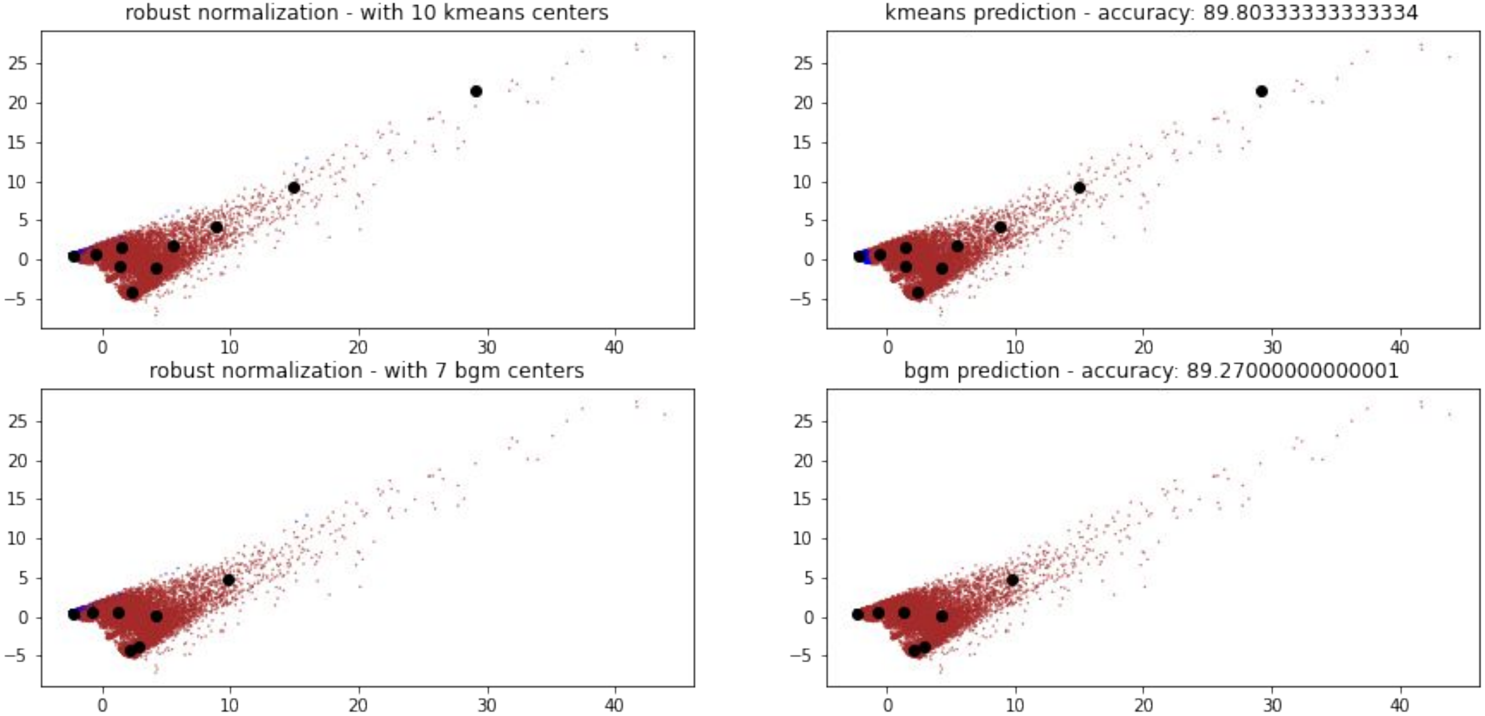


robust normalization - with 10 kmeans centers

kmeans prediction - accuracy: 91.55333333333333

robust normalization - with 10 bgm centers

bgm prediction - accuracy: 92.06

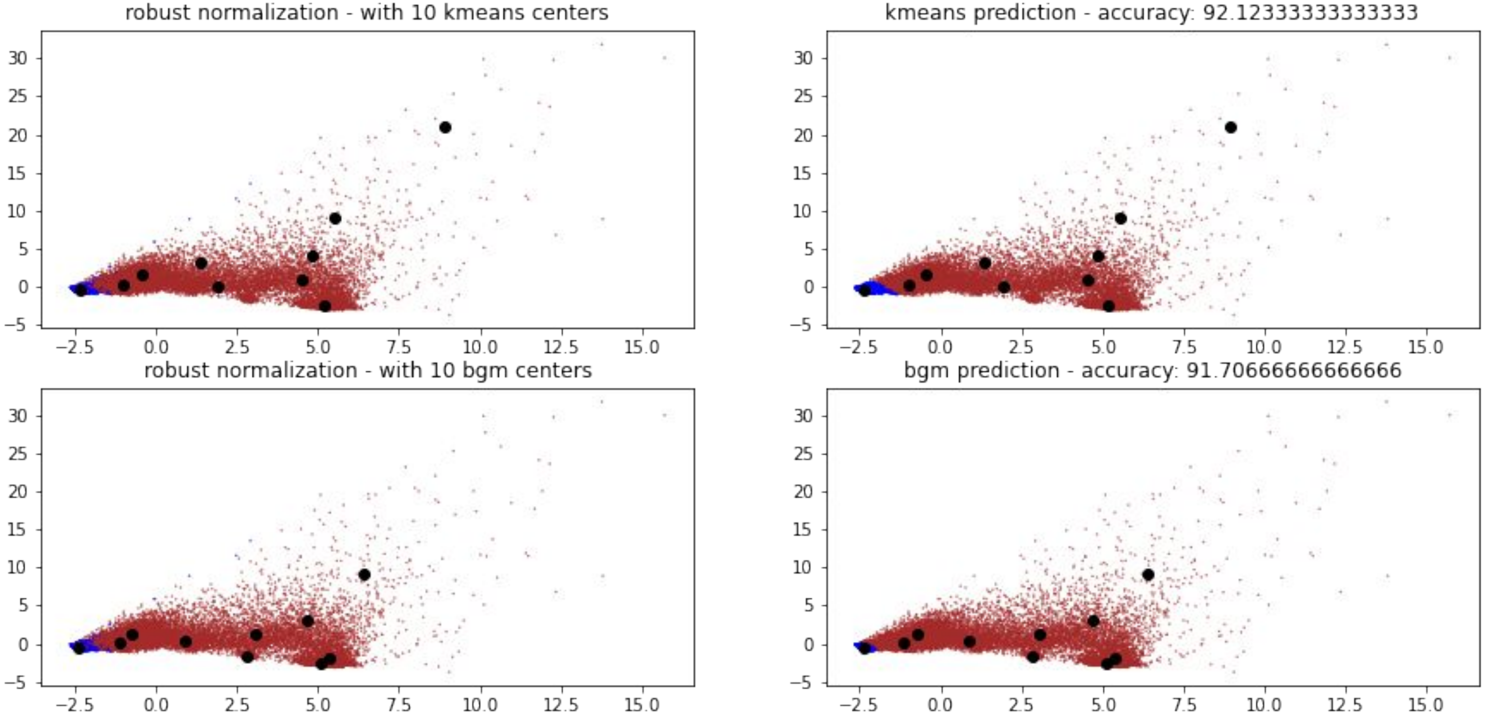


robust normalization - with 10 kmeans centers

kmeans prediction - accuracy: 89.80333333333334

robust normalization - with 7 bgm centers

bgm prediction - accuracy: 89.27000000000001

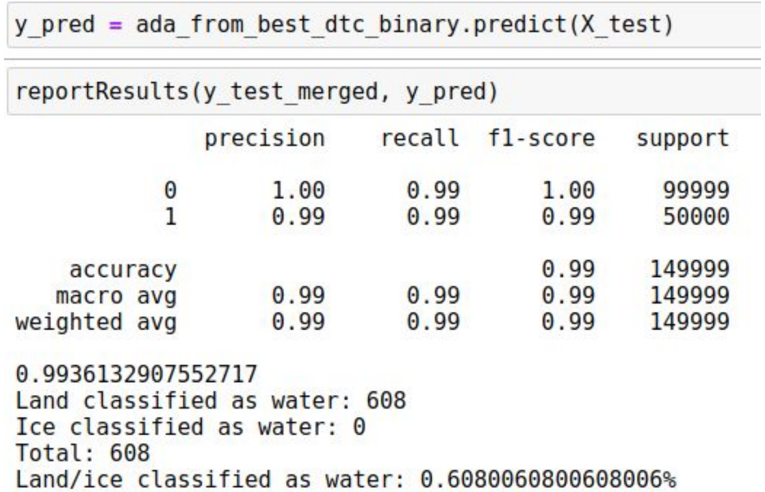


robust normalization - with 10 kmeans centers

kmeans prediction - accuracy: 92.12333333333333

robust normalization - with 10 bgm centers

bgm prediction - accuracy: 91.70666666666666



```
y_pred = ada_from_best_dtc_binary.predict(X_test)
```

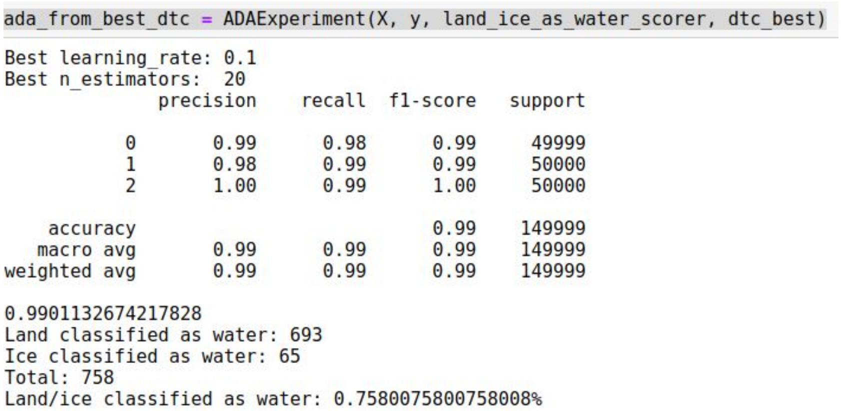
```
reportResults(y_test_merged, y_pred)
```

```
              precision    recall  f1-score   support

           0       1.00      0.99      1.00     99999
           1       0.99      0.99      0.99     50000

    accuracy                           0.99    149999
   macro avg       0.99      0.99      0.99    149999
weighted avg       0.99      0.99      0.99    149999

0.9936132907552717
Land classified as water: 608
Ice classified as water: 0
Total: 608
Land/ice classified as water: 0.6080060800608006%
```



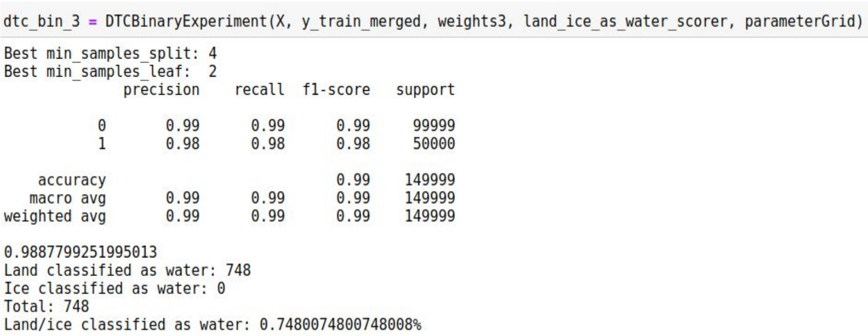
```
ada_from_best_dtc = ADAExperiment(X, y, land_ice_as_water_scorer, dtc_best)
```

```
Best learning_rate: 0.1
Best n_estimators:  20
              precision    recall  f1-score   support

           0       0.99      0.98      0.99     49999
           1       0.98      0.99      0.99     50000
           2       1.00      0.99      1.00     50000

    accuracy                           0.99    149999
   macro avg       0.99      0.99      0.99    149999
weighted avg       0.99      0.99      0.99    149999

0.9901132674217828
Land classified as water: 693
Ice classified as water: 65
Total: 758
Land/ice classified as water: 0.7580075800758008%
```



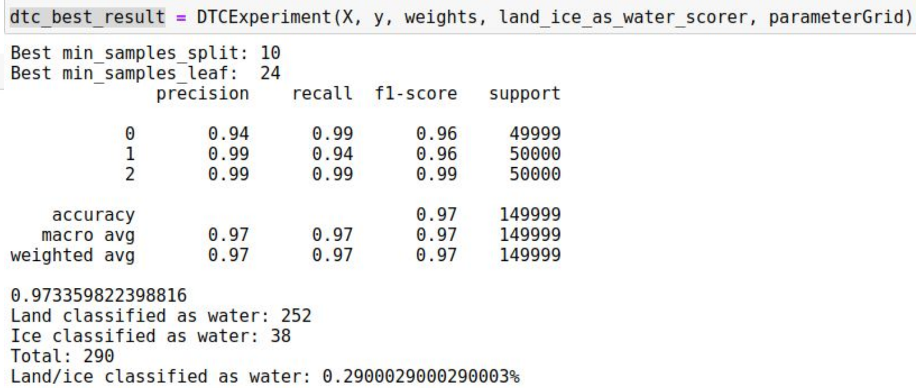
```
dtc_bin_3 = DTCBinaryExperiment(X, y_train_merged, weights3, land_ice_as_water_scorer, parameterGrid)
```

```
Best min_samples_split: 4
Best min_samples_leaf:  2
              precision    recall  f1-score   support

           0       0.99      0.99      0.99     99999
           1       0.98      0.98      0.98     50000

    accuracy                           0.99    149999
   macro avg       0.99      0.99      0.99    149999
weighted avg       0.99      0.99      0.99    149999

0.9887799251995013
Land classified as water: 748
Ice classified as water: 0
Total: 748
Land/ice classified as water: 0.7480074800748008%
```



```
dtc_best_result = DTCExperiment(X, y, weights, land_ice_as_water_scorer, parameterGrid)
```

```
Best min_samples_split: 10
Best min_samples_leaf:  24
              precision    recall  f1-score   support

           0       0.94      0.99      0.96     49999
           1       0.99      0.94      0.96     50000
           2       0.99      0.99      0.99     50000

    accuracy                           0.97    149999
   macro avg       0.97      0.97      0.97    149999
weighted avg       0.97      0.97      0.97    149999

0.973359822398816
Land classified as water: 252
Ice classified as water: 38
Total: 290
Land/ice classified as water: 0.2900029000290003%
```

Tree model hyper parameters

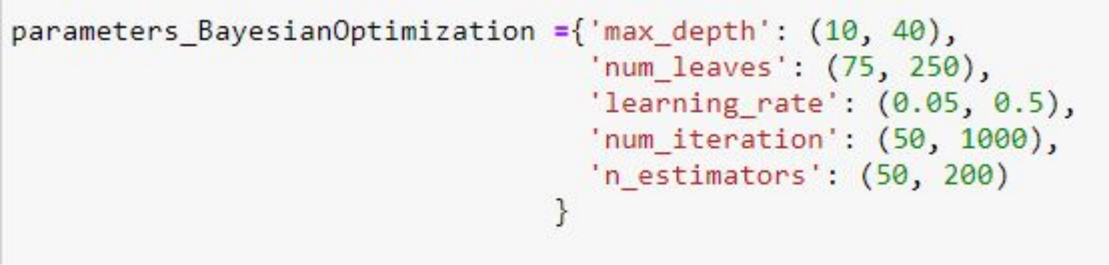
NN Binary classification model

2 classes

- Loss - Binary Cross Entropy Loss (onehot encoding)
- Activation functions - ReLU,sigmoid

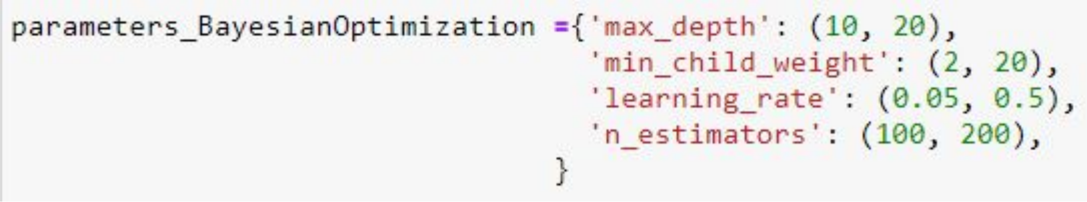
Tree Model Hyperparam ranges

LGBM



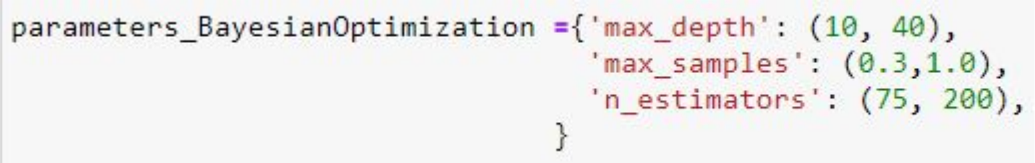
```
parameters_BayesianOptimization ={'max_depth': (10, 40),
                                  'num_leaves': (75, 250),
                                  'learning_rate': (0.05, 0.5),
                                  'num_iteration': (50, 1000),
                                  'n_estimators': (50, 200)
                                 }
```

XGBOOST



```
parameters_BayesianOptimization ={'max_depth': (10, 20),
                                  'min_child_weight': (2, 20),
                                  'learning_rate': (0.05, 0.5),
                                  'n_estimators': (100, 200),
                                 }
```

RF



```
parameters_BayesianOptimization ={'max_depth': (10, 40),
                                  'max_samples': (0.3,1.0),
                                  'n_estimators': (75, 200),
                                 }
```