Detecting Volcanoes on Venus via Classification

Team Mamamia!:Di Ye (diye2)[1], Hao Wang (haow2)[1], and Hannah Pu (chpu2)[1]

[1]Department of Statistics, University of Illinois at Urbana-Champaign

This version was compiled on December 8, 2018

LASSO Regression, logistic regression, neural network, random forest, and xgboost have been used in this project to identify whether each image in the Kaggle datasets contains volcanoes or not, and if there is any, how many volcanoes are there. The analysis of the images revealed that there are several volcanoes on Venus. Neural network has the best performance to identify whether the images have volcanoes in this project for image classification with an area under the curve 0.9629 and a high F-1 score 0.9747. The comparison between models is also included.

Image Classification | LASSO Regression | Logistic Regression | Marginal Screening | Neural Network | Random Forest | Xgboost

I. INTRODUCTION.

It is well known that Venus is the closest planet to Earth. However, the surface of Venus is obscured by layers of thick cloud cover. It was until NASA's Magellan spacecraft was launched off for a mission to Venus on May 4, 1989 that the first detailed information about the surface of Venus was revealed. The mission mapped surfaces of Venus and image recorded most 98% of Venus surface. Dr. King in his article *Volcanoes on Venus* has mentioned that "volcanic activity is the dominant process for shaping the landscape of Venus, with over 90% of the planet's surface being covered by lava flows and shield volcanoes". Based on the images of volcanoes on Venus, we aim to construct prediction model to identify whether an image contains volcanoes or not, and how many volcanoes are there. **Figure 1** is a simulated color image of the surface of Venus created by NASA using radar topography data acquired by the Magellan spacecraft.

Data Source Information. The data was downloaded from Kaggle, which is originally from NASA's Magellan spacecraft database. (https://www.kaggle.com/amantheroot/finding-volcanoes-on-venus/data)

Data Description. 9734 images were captured by the NASA's Magellan spacecraft in the 1990s and converted to pixels (110 x 110, from 0 to 255), where each image is one row of 12100 columns (all the 110 rows of 110 columns). Images can contain more than one volcanoes or maybe none. The 9000+ images are separated to four datasets (file names : *train_images*, *train_labels*, *test_images*, and *test_labels*):

Image Dataset (*train_images* and *test_images*) *Train_images* : 7000 images as train data with 12100 variables;
Test_images : 2734 images as test data with 12100 variables; All the variables (V1 to V12100) correspond to the pixel image, 110 pixels * 110 pixels = 12100 pixels.

Fig. 1. Volcanoes on Venus

Label Dataset (*train_labels* and *test_labels*) A summary of the variables in both *train_labels* and *test_label* datasets is listed down below:
1. *Volcano?* : If in the image there exists at least one volcano, the label is 1 (Yes). Otherwise, the label is 0 (No). (If Volcano? equals 0, the following three categories would be "NaN"). 2. *Type* : 1 = definitely a volcano,2 = probably, 3 = possibly, 4 = only a pit is visible
3. *Radius* : Is the radius of the volcano in the center of the image, in pixels?
4. *Number Volcanoes* : The number of volcanoes in the image.

Literature Review. In Kaggle, there were 11 kernels regarding data analysis of this dataset, all using Python. The kernels included vivid data visualization and ex-

ploratory data. Most kernels focused on classification prediction on whether there are volcanoes or not. For this classification prediction, different methods have been used to construct prediction model, from simple models such as logistic regression, to advanced models such as Convolutional Neural Network (CNN) and VGG Neural Network for deep learning. Accuracy of models ranged from 84.1% to 97%.

Scientific Goal. For this project, we focus mainly on doing classification predition on whether each image has a volcano or not. In addition, if the classification prediction goes well, we will also construct models to predict the number of volcanoes in the images. We aim in constructing different classification models and choosing the best model to predict whether there exists a valcano in each image. Identifying valcanoe through IT technology would increase the efficency of space exploration and safty of the crews.

II. EXPLORATORY DATA. The first 6 observations of *train_labels*

```
#    Volcano. Type Radius Number.Volcanoes
#  1        1    3  17.46                1
#  2        0  NaN    NaN              NaN
#  3        0  NaN    NaN              NaN
#  4        0  NaN    NaN              NaN
#  5        0  NaN    NaN              NaN
#  6        0  NaN    NaN              NaN
```

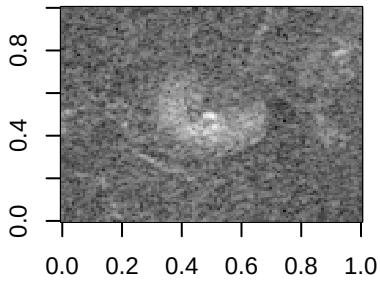

Fig. 2. Obs 10 Type: 1 Radius: 22.02 Number Volcanoes: 1

The first 6 observations of *test_labels*

```
#    Volcano. Type Radius Number.Volcanoes
#  1        0  NaN    NaN              NaN
#  2        0  NaN    NaN              NaN
#  3        1    1  17.00                1
#  4        0  NaN    NaN              NaN
#  5        1    3  15.13                1
#  6        0  NaN    NaN              NaN
```

After exploring the datasets, we found only labels have NaNs. The NaNs in the label dataset occurs only when there is no Volcanoe in the image observation, then `Type`, `Radius` and `Number.Volcanoes` would be NaNs as there are no volcanoes. We have set the all these NaNs to 0.

Data Visualization. 6 observations are picked to demonstrate how the images got labeled. **Figure 2 - 5** show how well the volcanoes can be seen from the images (*Type* : 1 = definitely a volcano,2 = probably, 3 = possibly, 4 = only a pit is visible). **Figure 6** contains 2 volcanoes, and Figure 6 contains no volcano at all. We can tell from the images that a bright white dot indicates a potential volcano, while the white dot might not be clear enough to see in the image, which means that it is hard to identify whether there is a volcano or not.

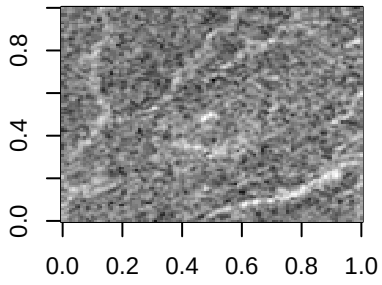

Fig. 3. Obs 39 Type: 2 Radius: 19.31 Number Volcanoes: 1

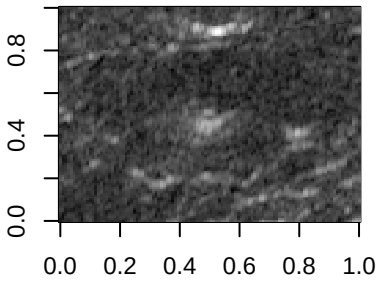

Fig. 6. Obs 289 Type: 1 Radius: 11.05 Number Volcanoes: 2

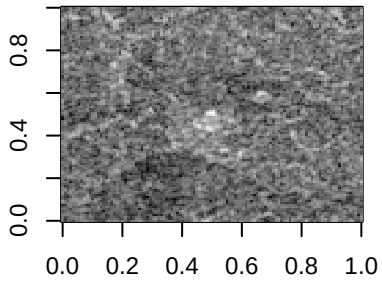

Fig. 4. Obs 1 Type: 3 Radius: 17.46 Number of Volcanoes: 1

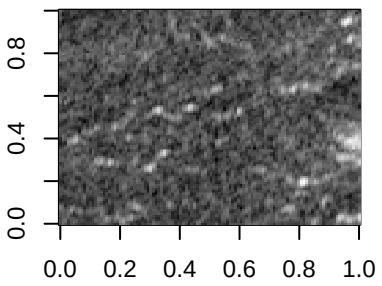

Fig. 7. Obs 2 No Volcano

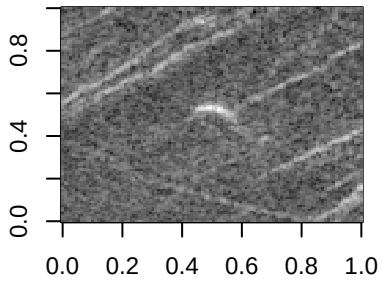

Fig. 5. Obs 30 Type: 4 Radius: 6.40 Number Volcanoes: 1

Data Summary. To facilitate the analysis process and explore the data, we have summarized the data. The `ggplot` has been used to visualize the training set labels. **Figure 8** shows that only 1000 images in the training dataset have volcanoes. **Figure 9** shows that within 1000 images that have volcanoes, how clearly we can identify the volcanoes. **Figure 10** shows how many volcanoes there are in each image. The number of volcanoes ranges from 0 to 5.

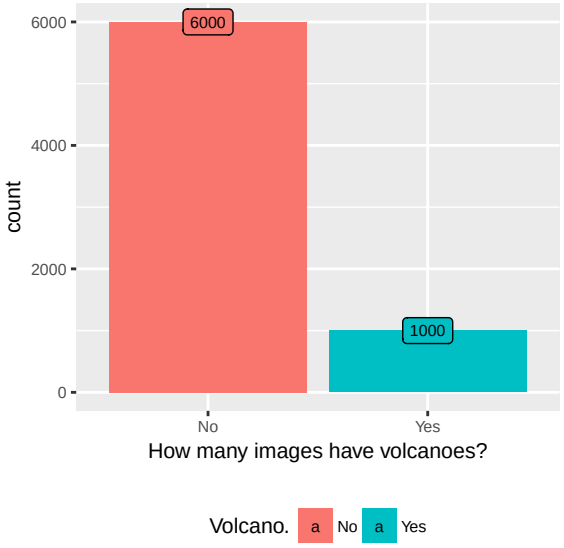

Fig. 8. How many images have volcanoes?

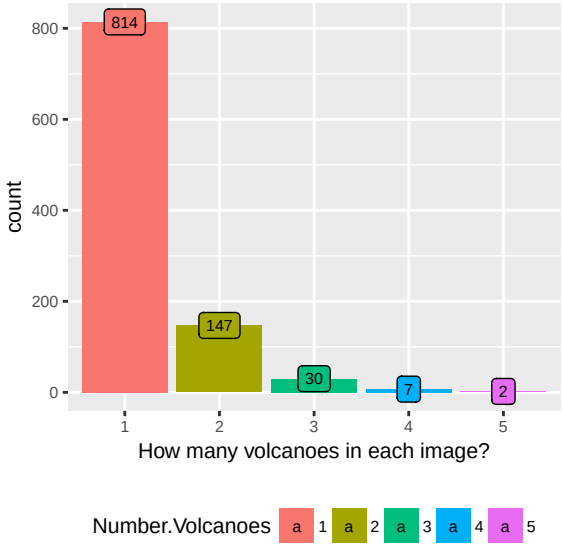

Fig. 10. How many volcanoes are there in each image?

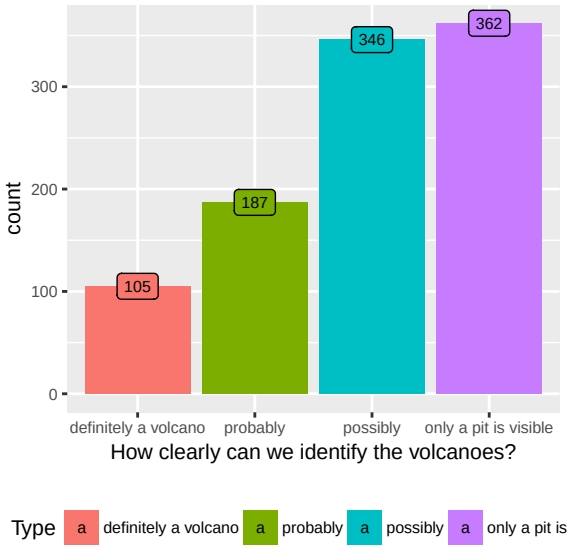

Fig. 9. How clealy can we identify the volcanoes?

III. METHODOLOGY & ANALYSIS.

Lasso Regression. To apply Lasso Regression,`glmnet` package was implemented. We first selected reasonable interval for tuning parameter λ by examing cross-validation error. After that, we did the model selection by using glmnet and use the "lambda.min" as final tuning parameter.

We ran two Lasso Regressions. One for classification: whether there is a volcano or not; one for Poisson distribution: how many volcanoes are there. By using the cross-validation error data, we identified two intervals, (-6.5, -5) and (-5, -4.5). Because of the different interval, we have different selected variable numbers: 470 and 114. The prediction using those pixels yielded a good start, a high prediction accuracy, for our following analysis.

Logistic Regression. Two Logistic Regressions were applied for classification and Poisson prediction. Before performing logistic regression, we used two variable selection methods to reduce dimension. 400 and 250 pixels were selected by using Marginal Screening, respectively; 470 and 114 pixels were selected by using above Lasso Regression. Then, we applyed logistic regressions on those 4 sets of selected pixels repectively. We then recorded their F1 score and AUC for classification, and RMSE for Poisson prediction.

Neural Network. Convolutional neural networks (CNNs) have been applied widely in image and video recognition and classification. CNNs are made up of neurons

with learnable weights and bias. Therefore, CNNs were used in the project for seeking better classfication results. Some pooling layers were added into the models to progressively reduce the spatial size and the amount of parameters and computation in the network. The raw data *train_images* and *test_images* was converted into numeric variables and reshaped into 3D array with dimension 7000 × 110 × 110 and 2734 × 110 × 110 respectively. The pixels were shrinked from 0-255 to 0-1 by dividing each pixel by 255. The target label (Volcano?) was converted into categorical variable for classification. The neural network models yielded efficient and satisfying classification results. The loss function is relatively low, indicating the models are good.

Two neural network models were run in this project. **Figure 12** shows the result of the first neural network model which is designed to classify whether an image contains a volcano. The model used 20% of the training data as validation data. **Figure 13** shows the result of the second neural network model which is designed to classify whether an image contains at least one volcano. The model, similar to the first one, used 20% of the training data as validation data.

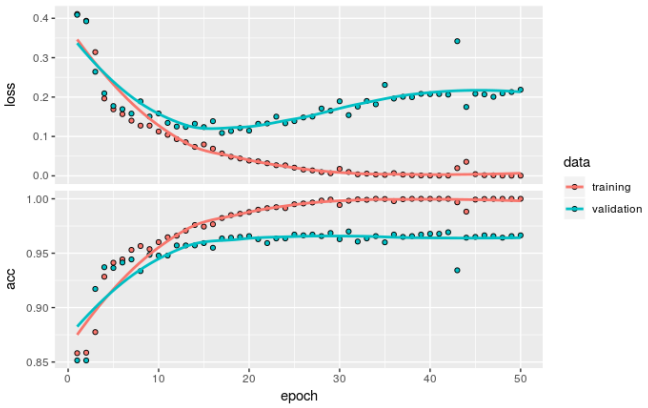

Fig. 11. Neural Network to classify whether an image contains volcanoes

The loss function and accuracy:

```
#  $loss
#  [1] 0.3018637
#
#  $acc
#  [1] 0.9568398
```

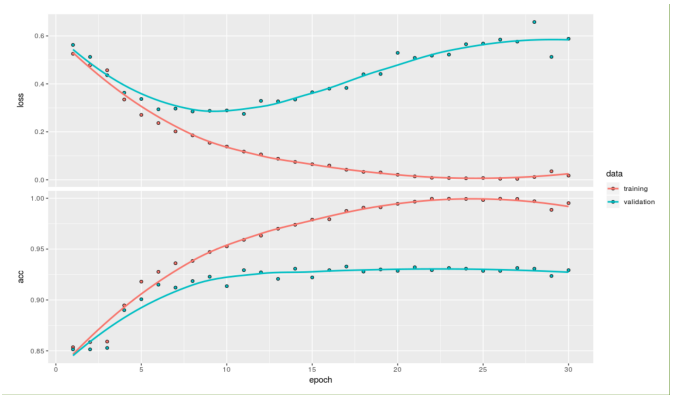

Fig. 12. Neural Network to classify how many volcanoes in each image

The loss function and accuracy:

```
#  $loss
#  [1] 0.554538
#
#  $acc
#  [1] 0.9297732
```

Random Forest. Random Forest was implemented using the `randomForest` package. Originally, n.tree was set to 200 and all other parameters were set as default. However, this took over than 10 hours to run. As it took too much time, it was terminated before it finished running. The final model used in this project was by setting n.tree to 200, mtry to 80 and nodesize to 70. Variable importance were recorded to compare with variable selection in other methods. Variable importance from random forest are shown in **Figure 11**. The yellow represents the important ones while black means those pixels are less meaningful for our prediction.

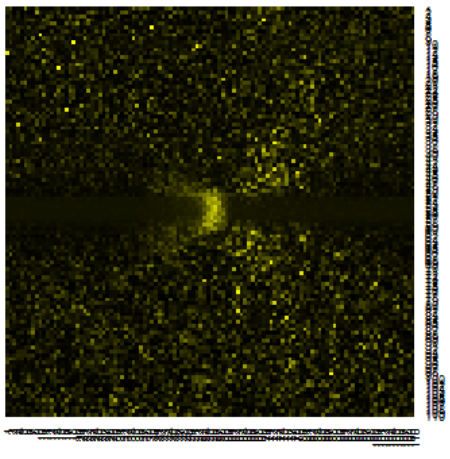

Fig. 13. The Random Forest variable importance heatmap

Xgboost. As xgboost is nowadays a popular algorithm used in machine learning and Kaggle competitions, this algorithm was also used in this project to see its performance. Xgboost was implemented using the `xgboost` package. Tuning was performed by using `xgb.cv` function with nfold = 5, parameters that were tuned are eta (learning rate), max_depth (maximum depth of a tree), min_child_weight (minimum sum of instance weight needed in a child), subsample (subsample ratio of the training instance), colsample_bytree (subsample ratio of columns when constructing each tree). The tuning process took over 12 hours and was terminated before it finished all the combinations. The parameter combination with the lowest error was used. From the tuning process, parameterts used for this model are set as following: eta = 0.1, max_depth = 5, min_child_weight =5, subsample = 0.65, colsample_bytree = 0.8, nrounds = 200, objective = 'binary:logistic'. To model how many volcanoes there are, objective = 'count:possion' was used with same parameters as before.

Variable selection methods.

Shrinking method (Lasso) By applying Lasso Regression, we were able to shrink highly correlated pixels to 0 and only had important ones left. With the tuning parameter λ's interval we selected by examing the cross validation result, we selected 470 important pixels for classification, 250 pixels for Poisson prediction, and apply those to lasso and logistic regression respectively.

Marginal Screening To apply Marginal Screening, we use for loop to model each pixels with outcome by using glm function, logistic regression. After sorting the p values from smallest to largest, we selected the top 400 pixels for classification and 250 for poisson prediction. The reason why we select top 400 and 250 pixels is that, after trying different number of top pixels, we found 400 pixels gave us highest prediction accuracy.

Variable selection methods comparison We originally only applied Marginal Screening to logistic regression. But after realizing Lasso helped us selected variables as well, we tried both methods on logistic regression.

By looking at the pixels' index, **figure 14** and **figure 16**, we found that the pixels selected by Marginal Screening focused between 4000-7000 for classification and between 5200 - 7000 for poisson prediction, which corresponds to the center of pictures; while pixels selected by lasso regression were more spread, as shown in **figure 15** and **figure 17**.

By applying those two different selection methods, we found the accuracy of Lasso selection is higher for classification; while for Poisson prediction, Marginal Screening selection performs better.

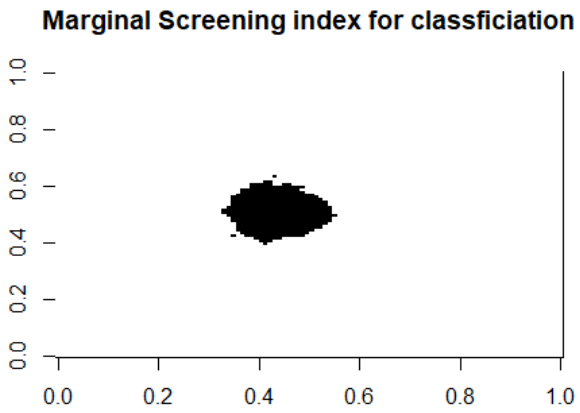

Fig. 14. The Marginal Screening index for classification

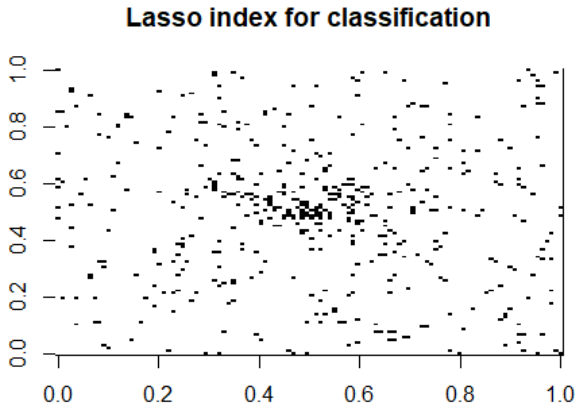

Fig. 15. The Lasso index for classification

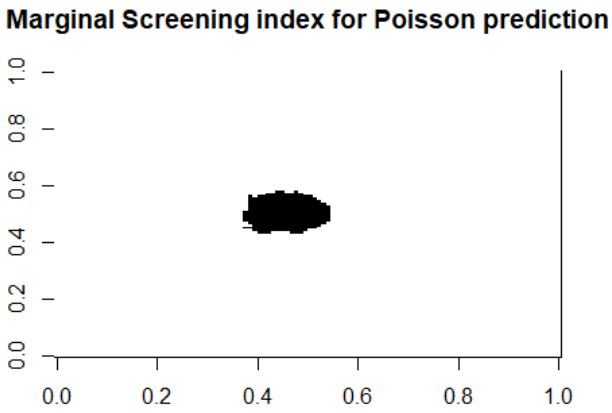

Fig. 16. The Marginal Screening index for Poisson prediction

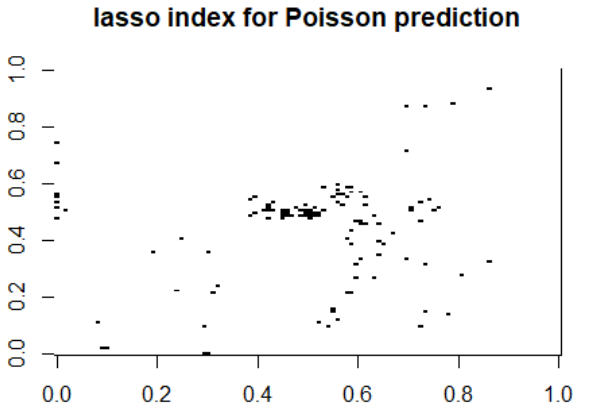

Fig. 17. The Lasso index for Poisson prediction

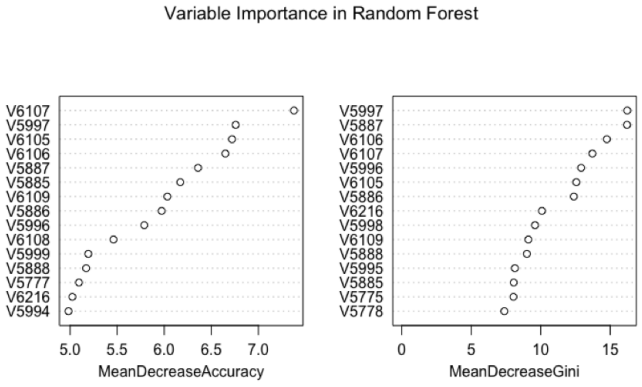

Fig. 18. Variable Importance in Random Forest

Hence, for classification, lasso' more spread pixels made more sense because Marginal Screening's centered selection might ignore some important pixels at the edges. For Poisson prediction, we prefer the Marginal Screening since the smaller number of pixels selected by Lasso regression would omit some important information.

Overall, we could conclude the variable selection criteria for this project is how much information the index can tell us without overfitting model.

IV. CONCLUSION & DISCUSSION.

Our model prediction results are summarized in table below.

Compared to structured data, the images used in this project are unstructured as volcanoes can occur in any pixels, this makes featuring difficult. Models such as lasso, logistic, random forest, require more of predefined features. In this project, convolution neural network works best as

Future Improvements. The limit of the models in this project is that, due to large parameter and sample size. Tuning advanced models such as random forest, xgboost, and neural network takes a lot of time. The tuning done in this project could be more fined tuned in the future. When predicting the number of volcanoes, parameters in xgboost and neural network were directly used from the previous model, therefore these models could also be further tuned in the future.

Table 1: Test dataset prediction result on whether there is volcanoe or not

Model	AUC	F1 Score
Neural Network	0.9629	0.9747
xgboost	0.9680	0.9628
Lasso	0.9347	0.9574
Random Forest	0.9367	0.9399
Logistic (Marginal Screening)	0.8429	0.9386
Logistic (Lasso Selection)	0.8672	0.9527

Table 2: Test dataset prediction result on how many volcanoes there are in the images

Model	RMSE
Logistic	0.5073
xgboost	0.0613
Lasso	0.4613

Neural Network

Since neural network models yielded the best results, we have concluded the classification table down below.

Table 3: Classification results using neural network for identifying volcanoes in the images

	No	Yes
No	2270	88
Yes	30	346

Table 4: Classification results using neural network for identifying the number of volcanoes in the images

	0	1	2	3
0	2272	67	15	2
1	23	266	44	7
2	5	26	4	3

V. APPENDIX.

```r
## code for neural network for number_volcano
train_y_number <- to_categorical(train_y$Number.Volcanoes,
    num_classes = 6)
test_y_number <- to_categorical(test_y$Number.Volcanoes,
    num_classes = 6)

model <- keras_model_sequential() %>% layer_conv_2d(filters = 8,
    kernel_size = c(3, 3), activation = "relu", input_shape = c(110,
        110, 1)) %>% layer_max_pooling_2d(pool_size = c(2,
    2)) %>% layer_conv_2d(filters = 16, kernel_size = c(3,
    3), activation = "relu") %>% layer_max_pooling_2d(pool_size = c(2,
    2)) %>% layer_conv_2d(filters = 16, kernel_size = c(3,
    3), activation = "relu") %>% layer_flatten() %>%
    layer_dense(units = 6, activation = "softmax")

model %>% compile(optimizer = "adam", loss = "categorical_crossentropy",
    metrics = c("accuracy"))

model_ret <- model %>% fit(train_images, train_y_number,
    epochs = 30, batch_size = 32, validation_split = 0.2)
plot(model_ret)
results <- model %>% evaluate(test_images, test_y_number)
pred_results_number <- model %>% predict(test_images)
pred_results_number_class <- model %>% predict_classes(test_images)
acc_loss_number <- model %>% evaluate(test_images,
    test_y_number)
save(pred_results_number, pred_results_number_class,
    acc_loss_number, file = "neural_network_number.RData")
model %>% save_model_hdf5("my_neural_network_model_number.h5")
```

```r
#code for xgboost
params <- list(
  eta = 0.1,
  max_depth = 5, #7
  min_child_weight =5,
  subsample = 0.65,
  colsample_bytree = 0.8,
  silent = 1
)
xgb.fit.final <- xgboost(
  params = params,
  data = as.matrix(train.x),
  label = train.y,
  nrounds = 200,
  objective = 'binary:logistic',
  verbose = 0
)
#code for random forest
rfmodel = randomForest(train.x, y=train.y[,1], ntree = 200, mtry = 80, nodesize = 70)
```

```r
# code for lasso
cv.out = cv.glmnet(x = train.x, y = train.y, alpha = 1,
    family = "binomial")
lam.seq = exp(seq(-6.5, -5, length = 100))
lasmodel = glmnet(x = train.x, y = train.y, alpha = 1,
    family = "binomial", lambda = lam.seq)

# code for marginal screening
cl <- makeCluster(3)
registerDoParallel(cl)
pre = Sys.time()
vol_results <- foreach(j = 1:ncol(train.x), .packages = "lme4",
    .combine = "rbind") %dopar% {
    pix = scale(train.x[, j])
    fit.glm <- summary(glm(as.factor(train.y) ~ pix,
        family = "binomial"))
    fit.glm$coefficients[2, 4]
}
Sys.time() - pre
stopCluster(cl)
# save(vol_results, file = 'vol_result.RData')
rownames(vol_results) = colnames(train.x)
colnames(vol_results) = "glm.p value"
totalpix = 400
sortedresults = as.matrix(vol_results[order(vol_results),
    ])
usepix = names(sortedresults[1:totalpix, ])
top_pix = match(usepix, rownames(vol_results))
# code for logistic regression:
logmodel = glm(as.factor(volcano) ~ ., data = training,
    family = "binomial")
train.error = (predict(logmodel, data.frame(sel.pix),
    type = "response") > 0.5)
log_pred = (predict(logmodel, data.frame(test.pipx),
    type = "response") > 0.5)
table(log_pred, test.y)
```

```r
# code for random forest variable important heatmap
myimport = rfmodel$importance
my_matrix = matrix(myimport[, 4], 110, 110, byrow = T)

library(RColorBrewer)
pal <- colorRampPalette(brewer.pal(11, "RdYlGn"))(100)
mycol <- c("black", "yellow")
pal <- colorRampPalette(mycol)(100)
heatmap(my_matrix, Rowv = NA, Colv = NA, revC = T,
    col = pal)
```