HarvardX PH125.9x Data Science Capstone: Dermatology

Xiao Guo

7/28/2021

1 Introduction

1.1 Introduction: Background

The Dermatology dataset was originally created by a research team from Bilkent University in 1998. The dataset was collected from patients with known diagnosis of erythemato-squamous diseases (skin conditions). The six types of erythemato-squamous diseases are psoriasis, seborrheic dermatitis, lichen planus, pityriasis rosea, chronic dermatitis, and pityriasis rubra pilaris. Twelve clinical features were collected by questioning or visually examining the patients. Twenty two histological features were collected by taking a skin biopsy and examining it under the microscope. These diseases are difficult to accurately diagnose because many share the same clinical and histological features. In addition, certain features are only present in a fraction of the patients of a specific disease. In the paper "Learning differential diagnosis of erythemato-squamous diseases using voting feature intervals" the scientists analyzed the dataset with an algorithm called voting feature intervals, which is an ensemble method based on the naive Bayes algorithm. Since that time, more machine learning methods have become available for classification tasks, some of which may offer improved efficiency or accuracy.

1.2 Introduction: Dataset and Goals

The research group that collected the dermatology data created two files. One file is the dataset composed of a matrix, where the rows are the patients and the columns are the tests and the diagnosis classifications. The other file contains a brief description of the dataset, as well as the column names and the diagnosis class names. These two files are combined to create a dataset with column names.

This dataset has thirty five columns (listed below). The first thirty four columns are the predictors (referred to as Attributes). These predictors are divided into two types, clinical predictors whose names start with "C" (observed in a clinic or questionnaire), and histological predictors whose names start with "H" (observed in a lab with a microscope). All the predictors except two have linear data that only take on the values of 0, 1, 2, and 3. The data type of the predictor Family History is nominal with values 0, and 1, while the data type of the predictor Age is linear and has a much greater range of values. The predictor Age also has missing values in some observations. The final column (D0_diagnosis) is the outcome. There are six types of outcomes (referred to as Class) corresponding to the six types of disease diagnoses.

```
## Classes 'data.table' and 'data.frame':   366 obs. of  35 variables:
##  $ C01_erythema                          : int  2 3 2 2 2 2 2 2 2 2 ...
##  $ C02_scaling                           : int  2 3 1 2 3 3 1 2 2 2 ...
##  $ C03_definite_borders                  : int  0 3 2 2 2 2 0 3 1 1 ...
##  $ C04_itching                           : int  3 2 3 0 2 0 2 3 0 0 ...
##  $ C05_koebner_phenomenon                : int  0 1 1 0 2 0 0 3 2 1 ...
##  $ C06_polygonal_papules                 : int  0 0 3 0 2 0 0 3 0 0 ...
```

```
##  $ C07_follicular_papules                 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ C08_oral_mucosal_involvement           : int  0 0 3 0 2 0 0 2 0 0 ...
##  $ C09_knee_and_elbow_involvement         : int  1 1 0 3 0 0 0 0 0 0 ...
##  $ C10_scalp_involvement                  : int  0 1 0 2 0 0 0 0 0 0 ...
##  $ C11_family_history                     : int  0 1 0 0 0 0 0 0 0 0 ...
##  $ H12_melanin_incontinence               : int  0 0 1 0 1 0 0 2 0 0 ...
##  $ H13_eosinophils_in_infiltrate          : int  0 0 0 0 0 2 0 0 0 0 ...
##  $ H14_PNL_infiltrate                     : int  0 1 0 3 0 1 0 0 0 0 ...
##  $ H15_papillary_dermis_fibrosis          : int  0 0 0 0 0 0 3 0 0 0 ...
##  $ H16_exocytosis                         : int  3 1 1 0 1 2 1 2 2 3 ...
##  $ H17_acanthosis                         : int  2 2 2 2 2 2 3 3 1 2 ...
##  $ H18_hyperkeratosis                     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ H19_parakeratosis                      : int  0 2 2 3 0 2 0 0 1 2 ...
##  $ H20_rete_ridges_clubbing               : int  0 2 0 2 0 0 0 0 0 0 ...
##  $ H21_rete_ridges_elongation             : int  0 2 0 2 0 0 2 0 0 0 ...
##  $ H22_suprapapillary_epidermis_thinning  : int  0 2 0 2 0 0 0 0 0 0 ...
##  $ H23_spongiform_pustule                 : int  0 2 0 2 0 1 0 0 0 0 ...
##  $ H24_munro_microabcess                  : int  0 1 0 0 0 0 0 0 0 0 ...
##  $ H25_focal_hypergranulosis              : int  0 0 2 0 2 0 0 0 0 0 ...
##  $ H26_granular_layer_disappearance       : int  0 0 0 3 2 0 0 2 0 0 ...
##  $ H27_basal_layer_vacuolisation_and_damage: int  0 0 2 0 3 0 0 2 0 0 ...
##  $ H28_spongiosis                         : int  3 0 3 0 2 2 0 3 2 2 ...
##  $ H29_retes_sawtooth_appearance          : int  0 0 2 0 3 0 0 2 0 0 ...
##  $ H30_follicular_horn_plug               : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ H31_perifollicular_parakeratosis       : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ H32_inflammatory_monoluclear_inflitrate : int  1 1 2 3 2 1 2 3 2 2 ...
##  $ H33_band_like_infiltrate               : int  0 0 3 0 3 0 0 3 0 0 ...
##  $ C34_age                                : chr  "55" "8" "26" "40" ...
##  $ D0_diagnosis                           : int  2 1 3 1 3 2 5 3 4 4 ...
```

The primary goal of the machine learning task is to use the thirty four predictors and classify the observations as one of the six possible diagnoses classes. The secondary goal is to compare the performance of several different classification algorithms.

1.3 Introduction: Overview

First, the data is divided into two sets, the training set and the testing set. The training set is used to build and tune the models. The test set is the hold-out set that is not used to build the models but reserved to evaluate the performance of each tuned model.

Second, both datasets are cleaned to ensure that the predictor variables have the appropriate data types, and the missing data are filled in.

Third, histograms are used to visualize the distribution of values for each predictor, and boxplots are used to visualize the differentiation of the classes for each predictor.

Fourth, using the raw data from the training set, four models are built with four tree based classification algorithms: decision tree, bagged tree, random forest, and gradient boost. The same four algorithms are used to construct four additional models utilizing up-sampled training data, which has balanced diagnosis classes. The result is eight tuned models.

Fifth, these eight models are used to make predictions on the test set.

Finally, the accuracy and other statistical metrics for the eight models are examined.

2 Methods

2.1 Methods: Preprocessing

Preprocessing involves three steps: 1) converting the predictors to the appropriate data types, 2) splitting the dataset into training and testing sets, and 3) imputing the missing values in the data.

The outcome column (D0_diagnosis) is converted to factor data type since it is nominal data. The predictor Family History (C11_family_history) is converted to logical data type since it is Boolean data. Since the predictor Age (C34_age) is numerical data, the missing values are converted to NA, and the data type is converted to numeric. All the other columns have the appropriate data types.

The dataset is split into the training set and the test set. The training set is used to train and tune the models. The test set is only used to evaluate the already trained models and not for building the models. Since the total number of observations is relatively small, 80% of the data is used for training and 20% is used for testing, to balance model building and model evaluation. If the dataset is much larger, then a 90% by 10% split can be used. The split is made with stratified sampling to ensure the proportion of the diagnosis classes in the training set and the testing set are the same.

Imputing the missing values requires using the entire dataset as predictors to estimate the missing values. To prevent data leakage, (where data from the test set contributes to building the models), this step is performed after splitting the original dataset into the training and the testing sets. This process requires creating dummy variables, calculating the missing values, and inserting the missing values.

2.2 Methods: Data Exploration

There are 290 observations (patients) in the training set. There are 6 distinct outcome (diagnosis) classes, where the number of observations in each class are not balanced. The number of observations for each class are 89 in psoriasis, 48 in seborrheic dermatitis, 57 in lichen planus, 39 in pityriasis rosea, 41 in chronic dermatitis, and 16 in pityriasis rubra pilaris. The range for all the predictors with three exceptions, is from 0 to 3. The range for Family History is 0 to 1, Eosinophils in Infiltrate is 0 to 2, and Age is 0 to 75.

The names and the range of values for all the predictors are shown below:

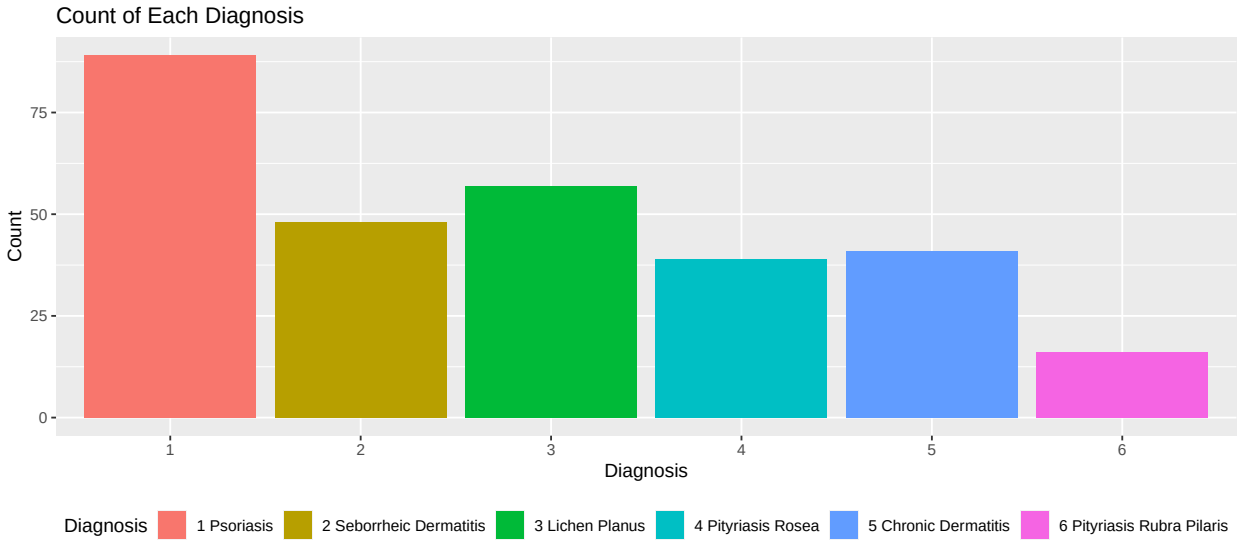
```
##                                      [,1]          [,2]
## .. C01_erythema                     "Min.   :0  "  "Max.   :3  "
## .. C02_scaling                      "Min.   :0  "  "Max.   :3  "
## ..C03_definite_borders              "Min.   :0  "  "Max.   :3  "
## .. C04_itching                      "Min.   :0  "  "Max.   :3  "
## ..C05_koebner_phenomenon            "Min.   :0  "  "Max.   :3  "
## ..C06_polygonal_papules             "Min.   :0  "  "Max.   :3  "
## ..C07_follicular_papules            "Min.   :0  "  "Max.   :3  "
## ..C08_oral_mucosal_involvement      "Min.   :0  "  "Max.   :3  "
## ..C09_knee_and_elbow_involvement    "Min.   :0  "  "Max.   :3  "
## ..C10_scalp_involvement             "Min.   :0  "  "Max.   :3  "
## ..C11_family_history                "Min.   :0  "  "Max.   :1  "
## ..H12_melanin_incontinence          "Min.   :0  "  "Max.   :3  "
## ..H13_eosinophils_in_infiltrate     "Min.   :0  "  "Max.   :2  "
## ..H14_PNL_infiltrate                "Min.   :0  "  "Max.   :3  "
## ..H15_papillary_dermis_fibrosis     "Min.   :0  "  "Max.   :3  "
## ..H16_exocytosis                    "Min.   :0  "  "Max.   :3  "
## ..H17_acanthosis                    "Min.   :0  "  "Max.   :3  "
## ..H18_hyperkeratosis                "Min.   :0  "  "Max.   :3  "
## ..H19_parakeratosis                 "Min.   :0  "  "Max.   :3  "
```

```
## ..H20_rete_ridges_clubbing                "Min.   :0  "  "Max.   :3  "
## ..H21_rete_ridges_elongation              "Min.   :0  "  "Max.   :3  "
## ..H22_suprapapillary_epidermis_thinning   "Min.   :0  "  "Max.   :3  "
## ..H23_spongiform_pustule                  "Min.   :0  "  "Max.   :3  "
## ..H24_munro_microabcess                   "Min.   :0  "  "Max.   :3  "
## ..H25_focal_hypergranulosis               "Min.   :0  "  "Max.   :3  "
## ..H26_granular_layer_disappearance        "Min.   :0  "  "Max.   :3  "
## ..H27_basal_layer_vacuolisation_and_damage "Min.   :0  "  "Max.   :3  "
## ..H28_spongiosis                          "Min.   :0  "  "Max.   :3  "
## ..H29_retes_sawtooth_appearance           "Min.   :0  "  "Max.   :3  "
## ..H30_follicular_horn_plug                "Min.   :0  "  "Max.   :3  "
## ..H31_perifollicular_parakeratosis        "Min.   :0  "  "Max.   :3  "
## ..H32_inflammatory_monoluclear_inflitrate "Min.   :0  "  "Max.   :3  "
## ..H33_band_like_infiltrate                "Min.   :0  "  "Max.   :3  "
## ..   C34_age                              "Min.   : 0  " "Max.   :75  "
```

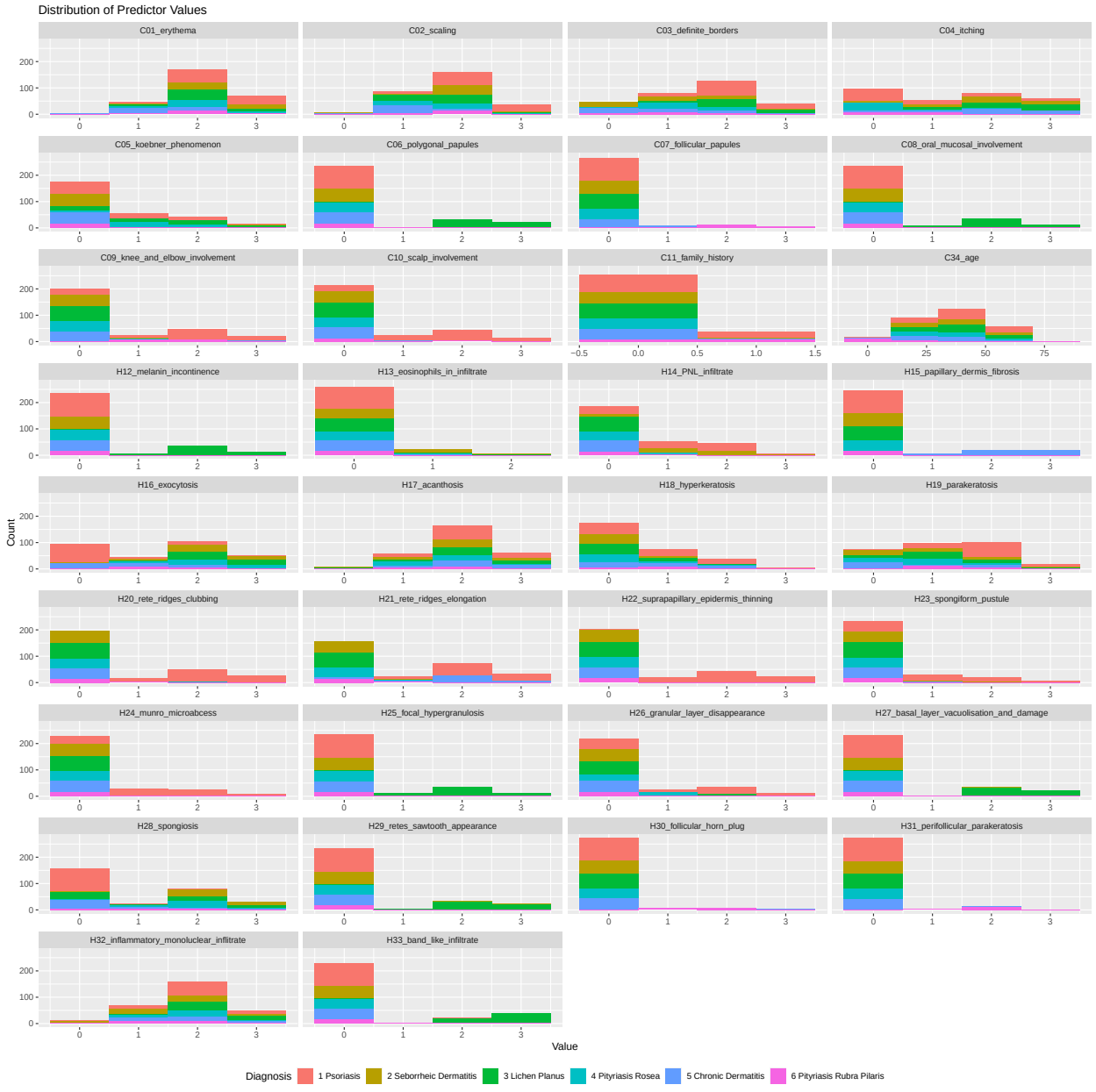
2.3 Methods: Data Visualization

The predictors are plotted to visualize the distribution of the diagnosis classes. Each color represents a different diagnosis class (skin condition).

2.3.1 Count of Each Diagnosis The Count of Each Diagnosis plot represents the number of patients in each diagnosis class. It shows the distribution and the prevalence of the six diagnoses. The prevalence of seborrheic dermatitis, lichen planus, pityriasis rosea, and chronic dermatitis are similar. However, the prevalence of psoriasis is significantly higher, and pityriasis rubra pilaris is significantly lower than the others. Since the classes are not balanced, the ideal model should take prevalence into account.



2.3.2 Distribution of Predictor Values The Distribution of Predictor Values plot has thirty four individual histograms, one for each predictor. Every plot shows the distribution of predictor values for each of the six classes. These plots show how the distribution of one predictor differs from another predictor. Some predictors (such as C03_definite_borders) have a distribution that is spread out among all the values, and all the diagnosis classes have a similar distribution shape. These will likely be poor predictors for diagnosis classification. Many predictors (such as H22_suprapapillary_epidermis_thinning) have a distribution where the majority of the patients have a value of zero and only a few patients have non-zero values. These will likely be good predictors for diagnosis classification. More importantly, these plots compare how the distribution for a specific predictor differs among the six diagnoses. For example, some predictors (such as C08_oral_mucosal_involement) only has one diagnosis with non-zero values. These will be the most useful predictors for differentiating between different classes



2.3.3 Distribution of Diagnosis Classes The Distribution of Diagnosis Classes plot also has thirty four individual box plots, one for each predictor. Every plot shows the average value of each diagnosis. These plots provide some clues on which predictor can be useful for differentiating between different diagnoses. Some predictors (such as C01_erythema) have similar average values for all the diagnoses. These features are equally likely to be observed in most or all of the diagnoses, and will not have significant impact in differentiating between diagnoses. Some predictors (such as H33_band_like_infiltrate) only have one or two diagnoses with non-zero averages, while all the other diagnoses have an average of zero. These features that are unique to only one or two of the diagnoses, and will have a significant impact in differentiating between diagnoses.

2.3.4 Overall Insights Diagnoses psoriasis, lichen planus, chronic dermatitis, and pityriasis rubra pilaris all have at least one predictor where it is the only diagnosis class with a non-zero average. These four diagnosis classes should be easy to classify. Diagnoses seborrheic dermatitis, has one predictor where it has

a non-zero average alongside psoriasis. Diagnosis pityriasis rosea does not have any predictors where it has the only non-zero average. These two diagnosis classes should be more difficult to classify.

In theory, a decision tree should be able to differentiate all six diagnoses. The procedure should be classifying the easy to identify diagnoses first. Once those four classes are ruled out, the fifth diagnosis can be identified. Finally, if an observation does not belong to any of the previous five diagnoses, then it must be the sixth one.

2.4 Methods: Model Selection

The dermatology machine learning task requires the classification of patients into one of six diagnosis classes. There are many classification algorithms. However, only a few are suitable for categorizing this dataset.

The logistic regression based algorithms are inflexible and are mainly used for binary classification. The complex relationships in this multi-class dataset will be hard for logistic regression to capture.

The naive Bayes algorithm requires independence of predictors. This is not true for the predictors in this dataset.

The K-nearest-neighbours algorithm requires the number of predictors to be small. The large number of predictors in this dataset will cause models to suffer from the curse of dimensionality.

Discriminant analysis algorithms such as LDA and QDA require the data to be multivariate normal and have a small number of predictors. This data is not always normal and the large number of predictors will require the calculation of a large number of correlations, means, and standard deviations.

Support vector machines can work well for this dataset. However, since they are designed for binary outcomes, this multi-class classification task needs to be divided into six one-vs-rest binary classification tasks.

Tree based methods have several advantages for this type of machine learning task. The number of predictors can be large. The predictors can have different data types. Numerical predictors do not need to be standardized. The predictors do not need to be independent. Assumptions on the shape of the data are not required. Therefore, the tree based methods are chosen for this machine learning task.

2.4.1 Decision Tree The decision tree algorithm splits the set of observations into smaller groups based on specific criteria, where each of the final group is a class.

For this algorithm, the starting point is all the observations in the training set. The algorithm then chooses a predictor to create a decision rule. Usually, the predictor that is most effective at differentiating between the outcome classes is used first. The decision rule divides the group of observations into smaller subgroups based on the value of that predictor for each observation. The members of the same subgroup have similar values for the predictor that was just used. The algorithm then chooses another predictor, and divides each subgroup into smaller subgroups. A predictor can be used once, multiple times, or not at all. The algorithm stop splitting the subgroups once a criteria (such as maximum depth, or minimum members in a subgroup, or specific amount of model improvement) is reached.

The original set of observations is the root node, the terminal subgroups are the leaf nodes, and the intermediate subgroups are the branch nodes. The class for each leaf node is determined by the class of the majority of the observations in that leaf node.

When predicting an observation from the test set, every observation starts at the root node, moves through the branch nodes by following the decision rules, and gets classified once it reaches the leaf node.

Advantages of the decision tree are flexibility and interpretability. A disadvantage of the decision tree model (comparing to other tree models) is a relatively large variance and bias trade off. Deep trees have high variance and low bias, while shallow trees have high bias and low variance. Ensemble methods can overcome this issue. The bagged tree and random forest algorithms train multiple trees in parallel and improve algorithms with high variance and low bias. The gradient boost algorithm trains multiple trees in series and improves algorithms with high bias and low variance.

2.4.2 Bagged Tree The bagged tree algorithm combines multiple decision trees, each produced from a different bootstrapped sample.

This algorithm resamples the dataset with bootstrapping, where observations are drawn with replacement from the original training set. The process is repeated multiple times to create multiple bootstrap samples. Each bootstrapped sample has the same number of observations as the original training set. The bootstrapped samples are different from each other since some observations are absent and some observations are repeated. The algorithm then produces a decision tree from each of the bootstrapped samples. For each bootstrapped sample, the observations that are not used for training (the out of bag samples) are used for cross validation.

When predicting an observation from the test set, the result from each tree is calculated for that observation. The final class is determined through a vote of all the trees.

Comparing to a single tree, an ensemble of multiple trees reduces the variance and the risk of overtraining. An issue that the bagged tree model does not resolve is the possibility of tree correlation, where similar trees are built even though different bootstrapped samples are used. This occurs because the algorithm has access to the same predictors (all the predictors) when building each tree.

2.4.3 Random Forest The random forest algorithm performs the bagged tree algorithm but limits the available predictors at each split to a random subset of predictors.

This algorithm creates several bootstrapped samples from the original training set observations. When constructing the decision tree for the first bootstrapped sample, the algorithm makes only a subset of the predictors available to create the decision rule at the first split. The predictors available for use are selected at random. At each subsequent split, the algorithm randomly chooses another set of predictors to make available for use. All the other trees are constructed the same way, where the predictors available for use are randomly selected at each decision rule.

When predicting an observation from the test set, the result from each tree is calculated for that observation. The final class is determined through a vote of all the trees.

Comparing to the bagged tree, the randomness of the available predictors reduces the correlation between the different trees.

2.4.4 Gradient Boost The gradient boost algorithm combines many shallow trees made in sequence, where each tree considers the errors from the previous tree.

For each observation, this algorithm makes an initial prediction on the probability that this observation belongs to each of the outcome classes. Since this dataset has six classes, every observation has a set of six predictions, one for each class. These predictions are based on the prevalence of the classes in the training set. At this step, all the observations have the same set of predictions. For each observation, the algorithm calculates several residual values, one for each class. The residual is the difference between the predicted probability that a specific observation belongs to a specific class, and the true probability (0 or 1) that this specific observation belongs to that specific class. The calculated residuals are associated with their respective observation. At this time, each observation has a set of (six) predictions and a set of (six) residuals.

A decision tree is constructed to predict the residual values (not the outcome classes). The algorithm uses the current tree and a learning rate parameter to update the predictions on the probability that the observation belongs to each of the classes. At this step, the set of predictions starts to differ between different observations. For each observation, new residual values are calculated based on the new prediction values for each class. The algorithm then constructs another shallow tree, and updates the prediction and the residual for each observation. The process continues until a predetermined number of trees are constructed.

In summary, each observation always has a set of prediction values (probability that this observation belongs to each of the classes) and a set of residual values (the difference between the prediction values and the true

probability that this observation belongs to each of the classes). Each subsequent tree calculates new residual values. The algorithm then updates both the residual and the prediction values for every observation.

When predicting an observation from the test set, the prediction values from the final tree choose the class with the highest probability.

Similar to the random forest, the gradient boost algorithm is a high performance ensemble method. Where random forest is better for reducing the variance, gradient boost is better for reducing the bias.

2.5 Methods: Model Evaluation

Accuracy and kappa are two performance metrics used to evaluate and find the best performing classification models.

Accuracy is the proportion of instances that are classified correctly. However, accuracy does not take prevalence (the a priori distribution of each of the classes) into account. This can be an issue when the prevalence of the classes are not balanced. As an extreme imbalanced example, class 1 has 95% prevalence, and classes 2, 3, 4, 5, and 6 all have 1% prevalence each. In this case, simply guessing class 1 for all instances will produce a 95% accuracy.

Kappa (Cohen's Kappa Coefficient) takes prevalence into account. Since the distribution of the classes in this dataset is not balanced (psoriasis is significantly higher and pityriasis rubra pilaris is significantly lower than the other classes), kappa is the better metric to maximize in the algorithms. One downside of kappa is that it is not as interpretable as accuracy. Therefore, the accuracy from the model that has the best kappa is recorded to estimate the performance of the model.

2.6 Methods: Cross Validation

Cross validation is used to estimate the model performance (accuracy and kappa), and to find the optimal parameters for each algorithm. Five-fold cross validation (split the training set into five non-overlapping sets) is used so that 20% of the training data is used to estimate the model performance in each iteration. If more observations are available in the dataset, then a higher fold cross validation can be used. The cross validation is repeated five times to reduce variation and not be computationally expensive. If a more powerful computer is available, then more repeats can be performed. The model with the optimal parameters is saved for predicting the test set.

2.7 Methods: Model Training and Tuning

All the models are built by training the algorithm on the training set, and cross validated to estimate the performance metric on the training set (not the test set).

The decision tree, random forest, and gradient boost algorithms have tuning parameters. The bagged tree algorithm does not have any tuning parameters.

For algorithms with tuning parameters, the algorithm is first trained with the default tuning parameter values. Then, the kappa values produced from the default tuning parameter values are examined as a reference to choose several custom values for each tuning parameter. These tuning parameter values are placed in a tune grid and applied to the algorithm. The algorithm is trained again with each new combination of the tuning parameter values. This train object (containing all the trained models) is saved. Then, the kappa values for all combinations of the tuning parameter values are calculated and plotted to find the model that maximizes the kappa. Finally, the model with the optimal kappa is saved.

For algorithms without any tuning parameters, the algorithm is trained and the train object (containing the trained model) is saved.

The final step for both types of algorithms is to record the performance metrics in a summary table.

2.8 Methods: Balance Unbalanced Data

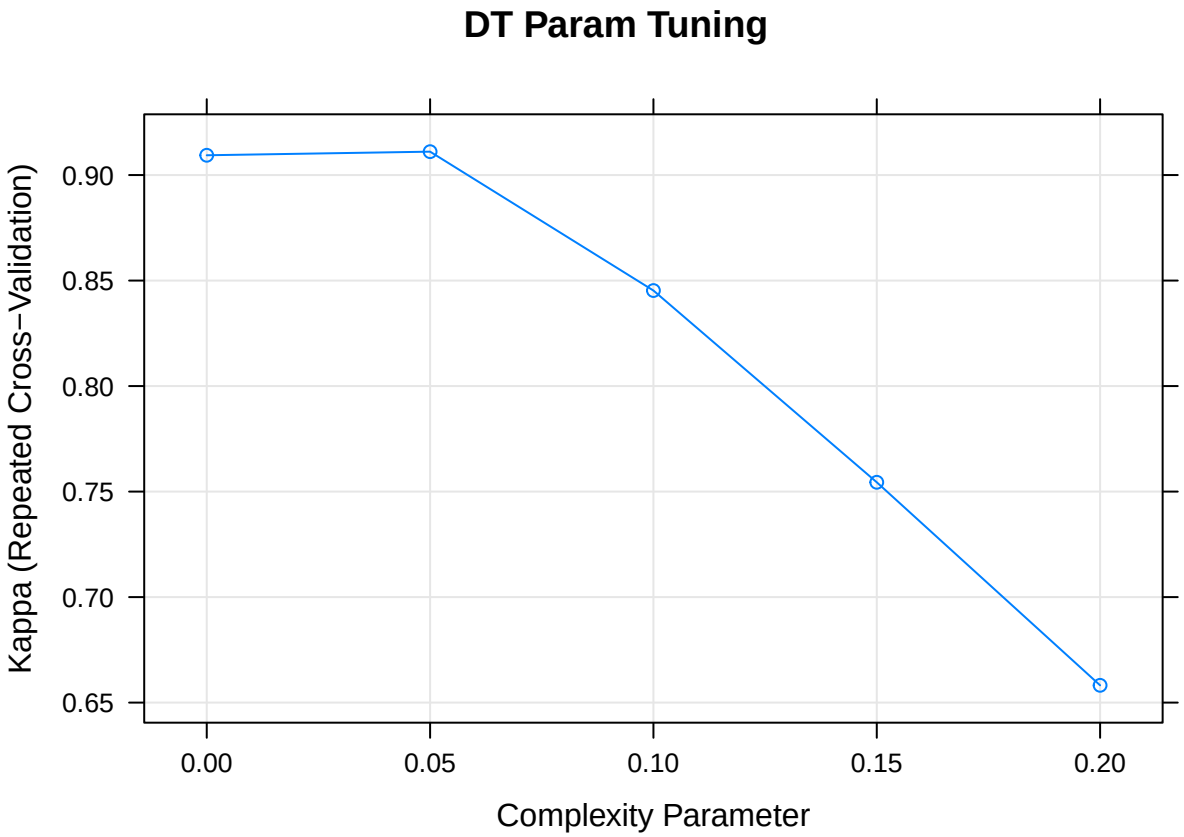
The data is unbalanced because the prevalence of the different diagnosis classes differ significantly (section 2.3.1). Even though the algorithms are maximizing kappa instead of overall accuracy, this imbalance in prevalence may still impose a limit on the achievable performance of any algorithm using this dataset. Higher performance may be achieved if the prevalence of the outcomes are balanced, where each class (diagnosis) has a similar number of observations (patients). A balanced dataset can be constructed with up-sampling, where the data from each class are sampled with replacement to create a new dataset where the class distributions are equal. Using this method, a balanced dataset is created from the training set. After all the algorithms are trained on the original dataset, all the algorithms are trained again using the up-sampled dataset to examine the impact of using a balanced dataset.

3 Results

3.1 Results: Models Built From Existing Training Data

The algorithms are trained on the training dataset. The performance of each model at predicting observations in the training set (not test set) are examined.

3.1.1 Decision Tree The available tuning parameter for the decision tree model is cp (complexity parameter). Cp controls the size of the tree by requiring a minimum improvement in prediction to split another node. Low cp values create larger trees to capture the complexity of the data. High cp values create smaller trees to prevent overfitting. The default cp values showed kappa decreased as cp increased from 0.20 to 0.27. Therefore, cp values 0, 0.05, 0.1, 0.15, and 0.2 are tested.




```
##     cp
## 2 0.05
```

	Model	Accuracy	Kappa
2	DT	0.9289883	0.9110892

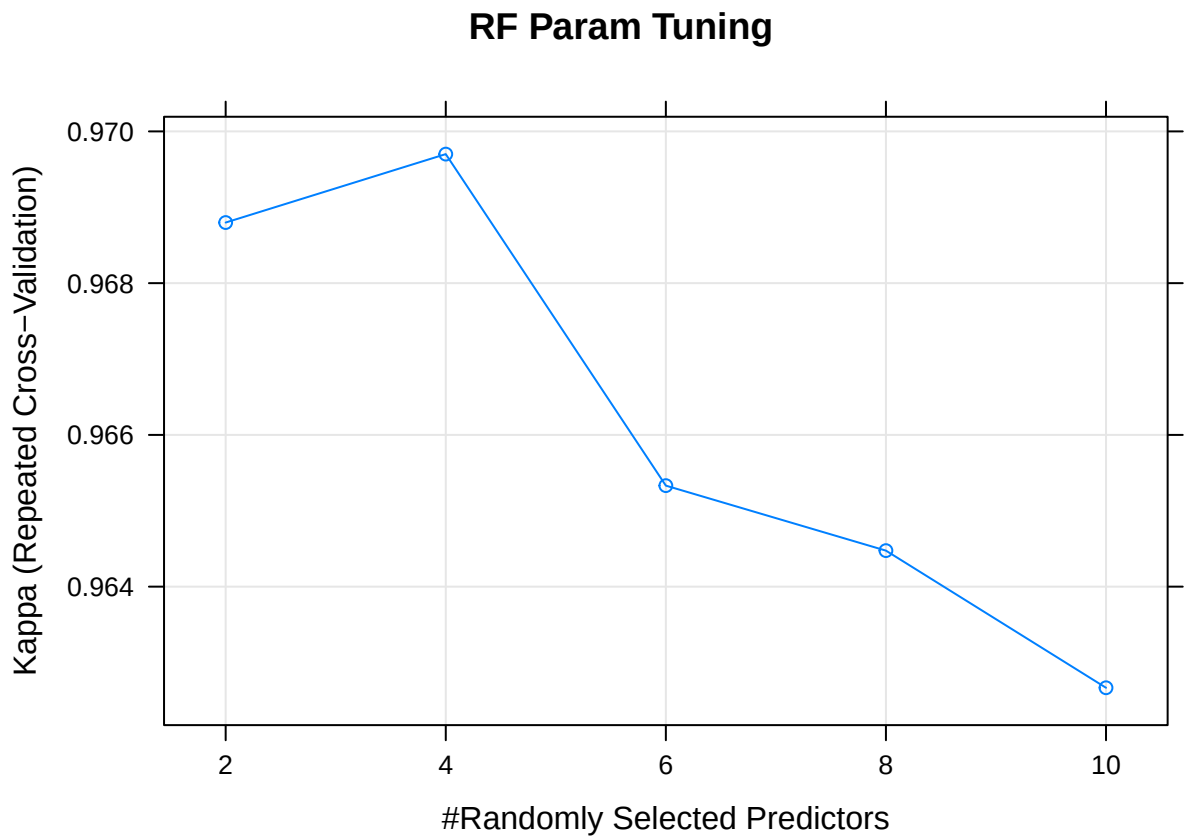
The decision tree model with the max kappa has a cp of 0.05 and an accuracy of 0.9289883, a good base performance value.

3.1.2 Bagged Tree There are no available tuning parameters for the bagged tree model.

	Model	Accuracy	Kappa
2	DT	0.9289883	0.9110892
1	BT	0.9564242	0.9453897

The bagged tree model with the max kappa has an accuracy of 0.9564242, an improvement on the decision tree performance.

3.1.3 Random Forest The available tuning parameter for the random forest model is mtry. Mtry determines the number of predictors to choose from for each split of a node. Low mtry values reduce the correlation between trees by decreasing the chance that different trees are created with the same predictors. High mtry values increase the model flexibility by decreasing the chance that only non-important variables are used at a node. Reasonable mtry values for classification tasks are around the square root of the number of predictors. Since 6 is close to the square root of 34, mtry values of 2, 4, 6, 8, and 10 are tested.




```
##   mtry
## 2    4
```

	Model	Accuracy	Kappa
2	DT	0.9289883	0.9110892
1	BT	0.9564242	0.9453897

	Model	Accuracy	Kappa
21	RF	0.9758334	0.9697007

The random forest model with the max kappa has a mtry of 4 and an accuracy of 0.9758334, an improvement on the bagged tree performance.

3.1.4 Gradient Boost The available tuning parameters for the gradient boost model are n.tree, interaction.depth, shrinkage, and n.minobsinnode.

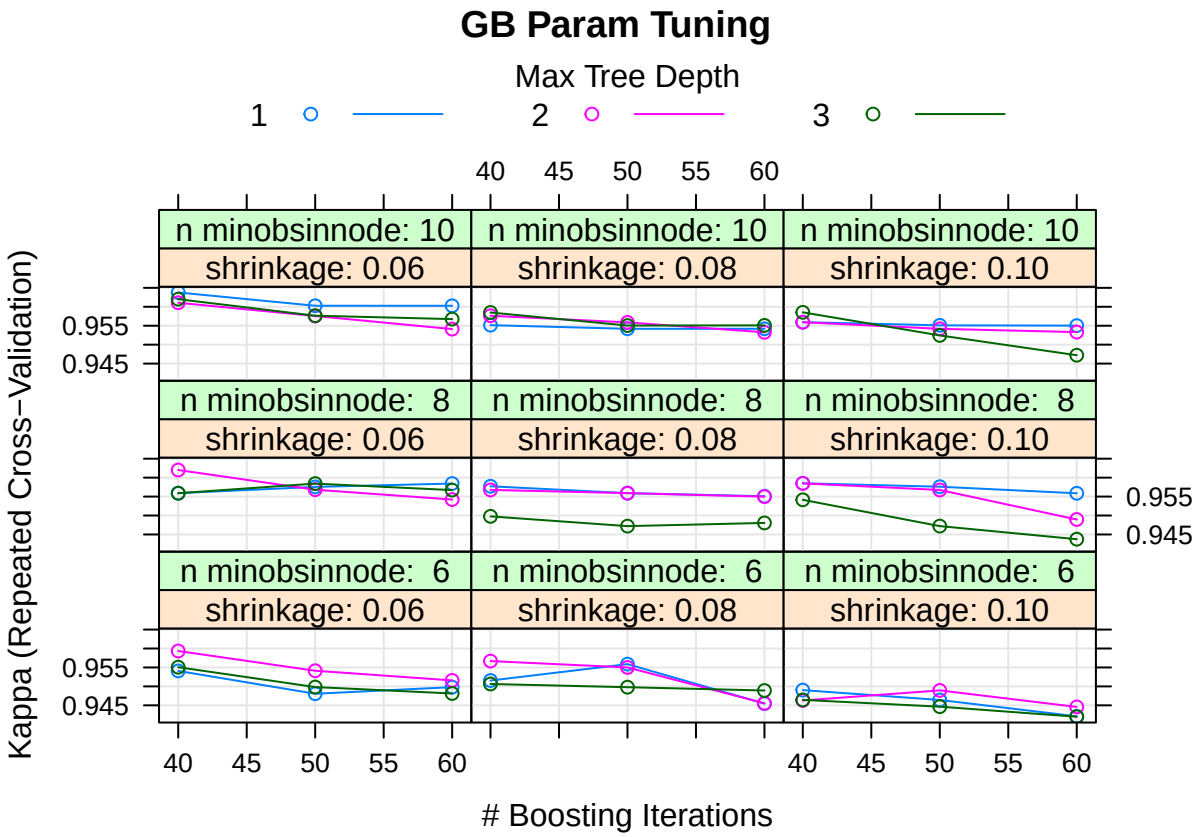
N.tree controls the number of boosting iterations (or the number of trees). Low n.tree values produce less trees, which prevents overfitting. High n.tree values produce more trees, which increases model flexibility. In the models built with default parameters, n.tree values of 50 had higher kappa than values of 100 and 150 in general. Therefore, values of 40, 50, and 60 are tested as they are close to 50.

Interaction.depth controls the number of splits performed on a tree. Low interaction.depth values produce less splits, which prevents overfitting. High interaction.depth values produce more splits, which increases model flexibility. In the models built with default parameters, lower interaction.depth values produced higher kappa. Therefore, the lowest possible values of 1, 2, and 3 are tested.

Shrinkage controls the impact of each tree, and is referred to as the learning rate. Low shrinkage values take small incremental steps to minimize the impact of faulty boosting iterations. High shrinkage values take large steps to speed up the runtime of the algorithm. In the models built with default parameters, the shrinkage parameter was held constant at 0.1. Therefore, values of 0.06, 0.08, and 0.1 are tested to increase model performance

N.minobsinnode is the minimum number of observations allowed in the terminal nodes. Low n.minobsinnode values allow the splitting of observations to smaller groups, which increases model flexibility. High n.minobsinnode values prevent the splitting of observations to smaller groups, which prevents overfitting. In the models built with default parameters, the n.minobsinnode parameter was held constant at 10. Therefore, values of 6, 8, and 10 are tested to increase model flexibility.

The interactions between the parameters are quite complex. For example, slower learning rate (shrinkage) requires more trees (n.tree). As another example, less trees (n.tree) requires larger trees (interaction.depth). Therefore, the impact of each tuning parameter is difficult to view in isolation, and a grid of many combinations of these tuning parameters is required.




```
##   n.trees interaction.depth shrinkage n.minobsinnode
## 7      40                 1      0.06             10
```

	Model	Accuracy	Kappa
2	DT	0.9289883	0.9110892
1	BT	0.9564242	0.9453897
21	RF	0.9758334	0.9697007
7	GB	0.9710408	0.9637713

The gradient boost model with the max kappa has n.trees of 40, interaction.depth of 1, shrinkage of 0.06, n.minobsinnode of 10 and an accuracy of 0.9710408, similar to the random forest performance.

3.1.5 Compare Models The summary table (above) lists the algorithm name, the accuracy achieved by the optimal model, and the highest achieved kappa. The value in the left-most column is the model number from the train function that resulted in the highest kappa (this value has no meaning and can be ignored). These accuracy values measure each model's performance on the training set (not test set). Therefore, these are not the true accuracy values. Instead they only offer an idea on the relative performance between the models.

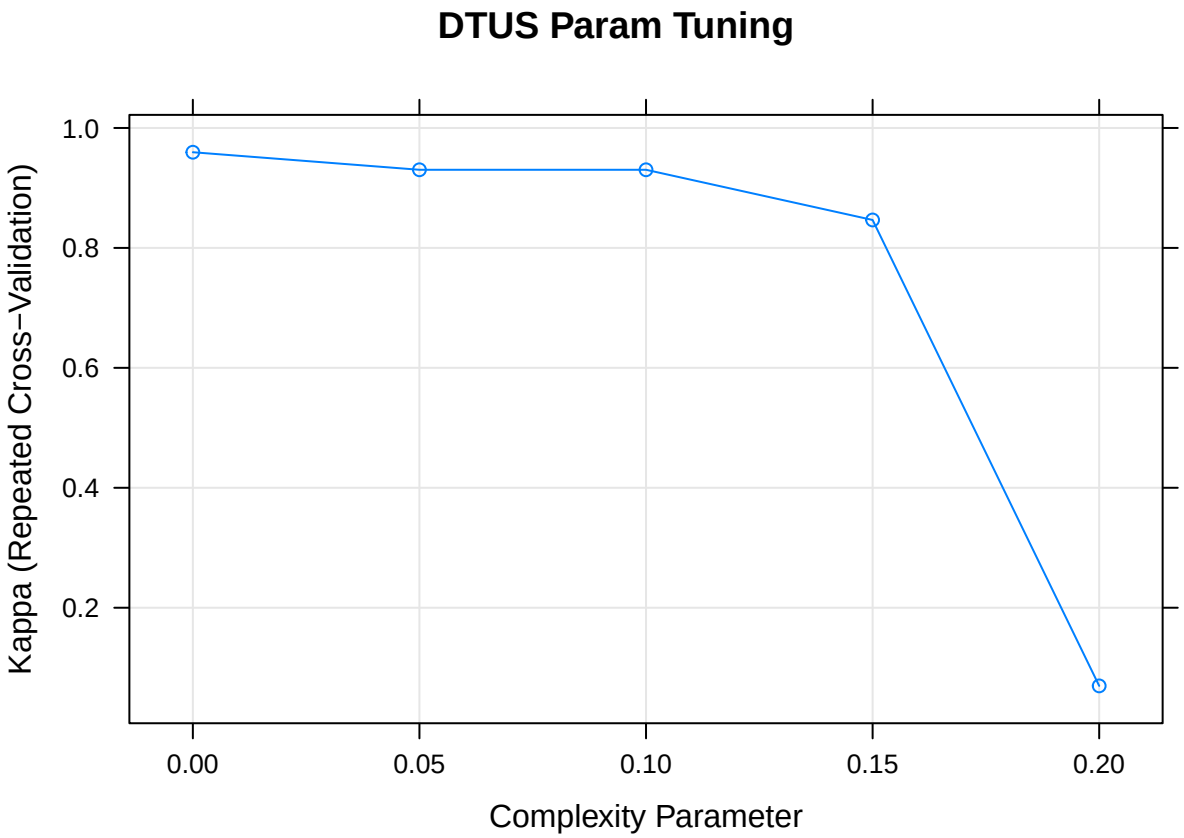
The single decision tree model achieved an accuracy in the low nineties. It is quite good as a baseline performance. This is not surprising, since the decision tree is already a highly flexible algorithm, and the exploratory analysis already showed that most of the classes can be differentiated from each other in theory. The bagged tree model reduced the variance, and improved the accuracy to the mid nineties. The random

forest model reduced the correlation between trees, and improved the accuracy to the high nineties. The gradient boost model had a similar accuracy value as the random forest model. The performance of most of these model (especially the gradient boost model, since gradient boost has much more tuning parameters than the other algorithms) may be increased by further adjusting their tuning parameters.

3.2 Results: Models Built From Up-Sampled Training Data

The algorithms are trained on the up-sampled training dataset. The performance of each model at predicting observations in the up-sampled training set (not test set) are examined.

3.2.1 Up-Sampled Decision Tree The decision tree algorithm is trained on the up-sampled training set.




```
##   cp
## 1  0
```

	Model	Accuracy	Kappa
2	DT	0.9289883	0.9110892
1	BT	0.9564242	0.9453897
21	RF	0.9758334	0.9697007
7	GB	0.9710408	0.9637713
11	DTUS	0.9662858	0.9595397

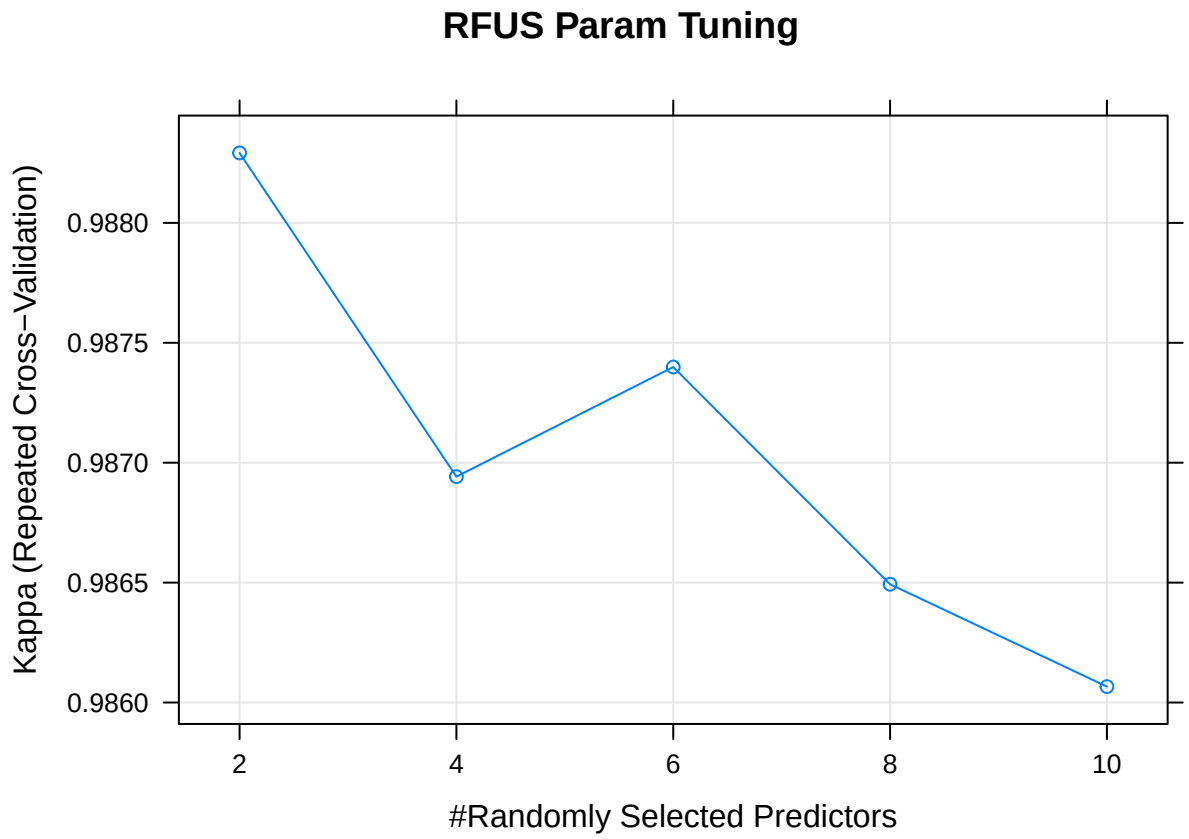
The decision tree model with the max kappa has a cp of 0 and an accuracy of 0.9662858, an improvement on the algorithm that trained on the unbalanced data.

3.2.2 Up-Sampled Bagged Tree The bagged tree algorithm is trained on the up-sampled training set.

	Model	Accuracy	Kappa
2	DT	0.9289883	0.9110892
1	BT	0.9564242	0.9453897
21	RF	0.9758334	0.9697007
7	GB	0.9710408	0.9637713
11	DTUS	0.9662858	0.9595397
12	BTUS	0.9872576	0.9847082

The bagged tree model with the max kappa has an accuracy of 0.9872576, an improvement on the algorithm that trained on the unbalanced data.

3.2.3 Up-Sampled Random Forest The random forest algorithm is trained on the up-sampled training set.



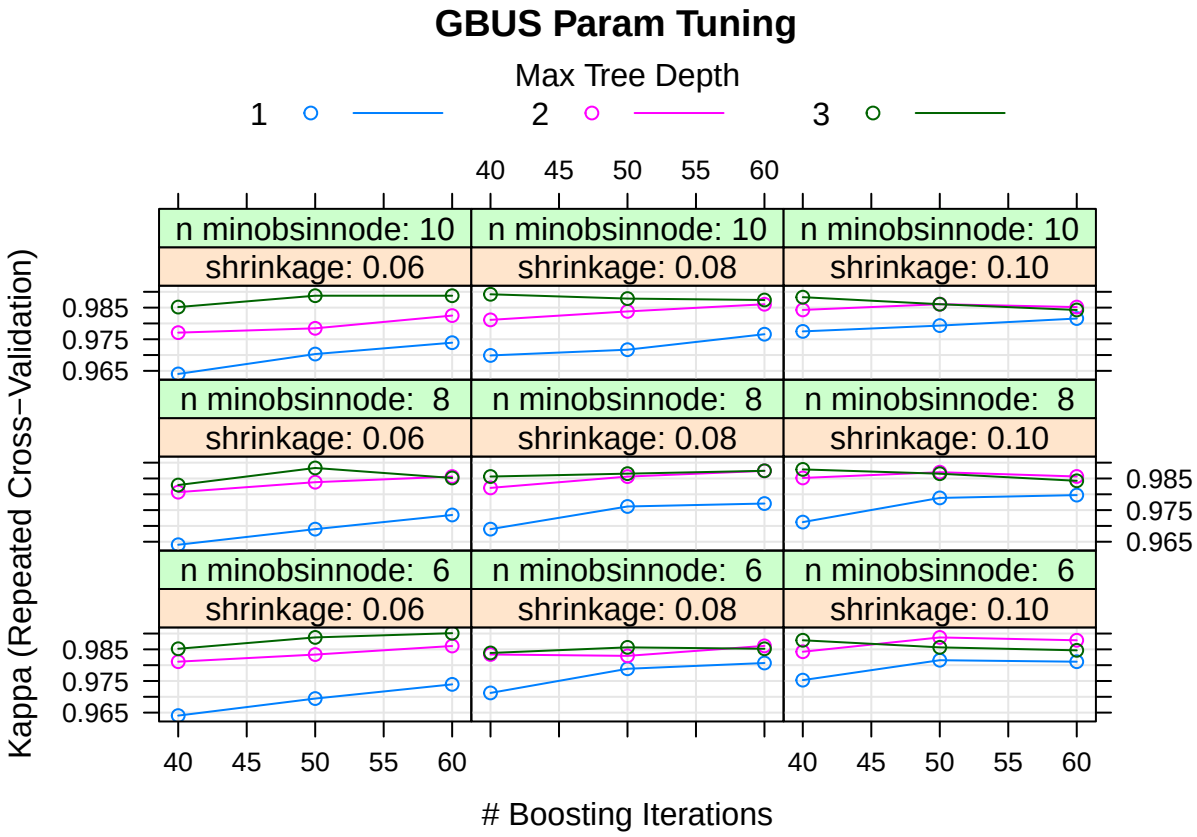

```
##   mtry
## 1    2
```

	Model	Accuracy	Kappa
2	DT	0.9289883	0.9110892
1	BT	0.9564242	0.9453897

	Model	Accuracy	Kappa
21	RF	0.9758334	0.9697007
7	GB	0.9710408	0.9637713
11	DTUS	0.9662858	0.9595397
12	BTUS	0.9872576	0.9847082
13	RFUS	0.9902442	0.9882916

The random forest model with the max kappa has a mtry of 2 and an accuracy of 0.9902442, an improvement on the algorithm that trained on the unbalanced data.

3.2.4 Up-Sampled Gradient Boost The gradient boost algorithm is trained on the up-sampled training set.




```
##    n.trees interaction.depth shrinkage n.minobsinnode
## 21      60                 3      0.06              6
```

	Model	Accuracy	Kappa
2	DT	0.9289883	0.9110892
1	BT	0.9564242	0.9453897
21	RF	0.9758334	0.9697007
7	GB	0.9710408	0.9637713
11	DTUS	0.9662858	0.9595397

	Model	Accuracy	Kappa
12	BTUS	0.9872576	0.9847082
13	RFUS	0.9902442	0.9882916
211	GBUS	0.9917684	0.9901214

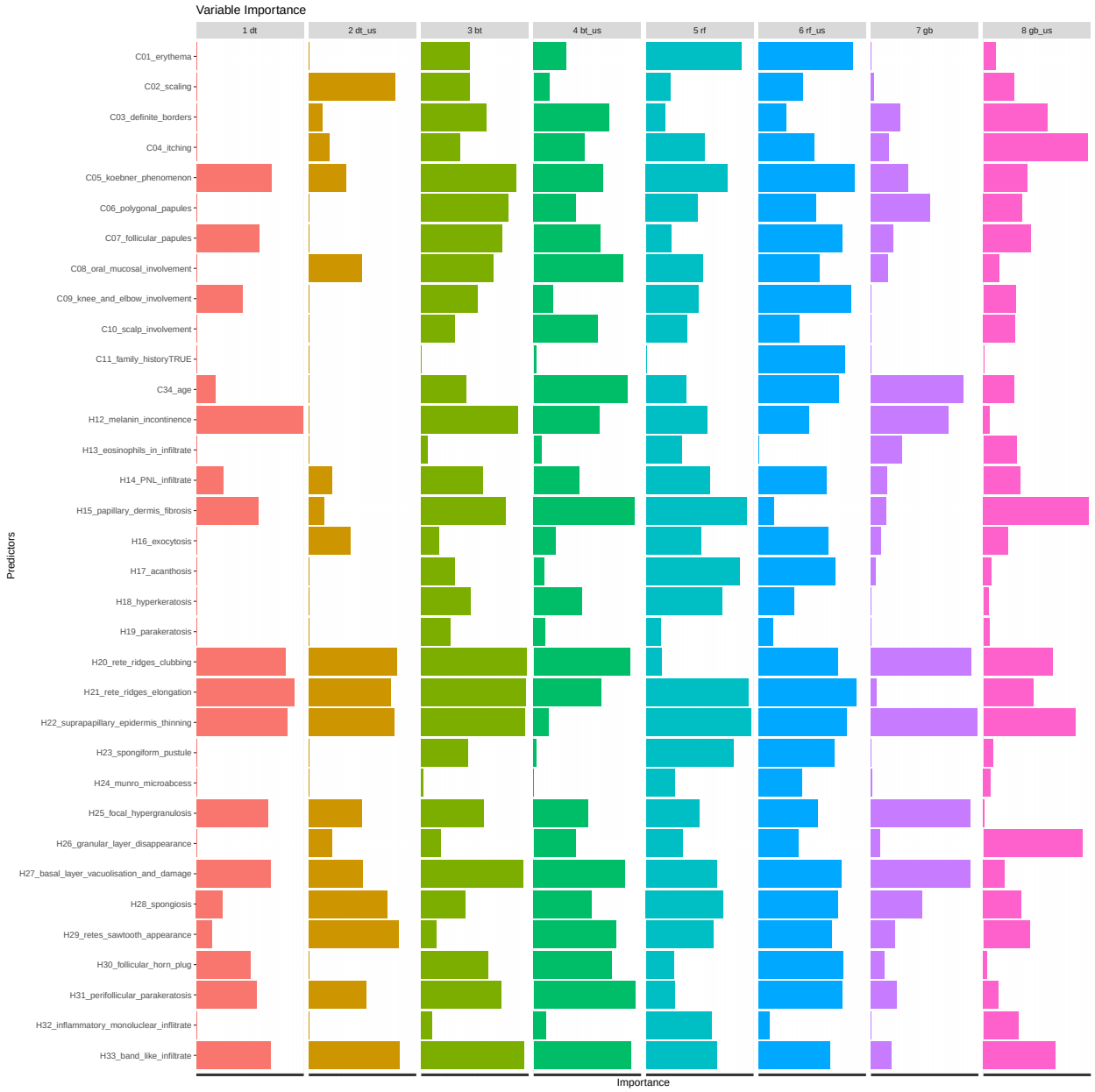
The gradient boost model with the max kappa has n.trees of 60, interaction.depth of 3, shrinkage of 0.06, n.minobsinnode of 6 and an accuracy of 0.9917684, an improvement on the algorithm that trained on the unbalanced data.

3.2.5 Compare Up-Sampled Models The performance of the model using the up-sampled dataset had the same pattern as before, where the performance of the decision tree is the lowest, the performance of the bagged tree is somewhat higher, and the performances of the random forest and gradient boost are the highest. In general, the performance of all the up-sampled models are better than that of the respective unbalanced models. This suggests that balancing the classes does improve the performance of the algorithms.

3.3 Results: Variable Importance

Variable importance shows how useful each predictor is at differentiating between the different diagnoses classes.

The variable importance plot is eight individual bar plots, one for each model. Each bar plot shows the relative importance of the thirty four predictors in their respective model.



The decision tree models have several predictors with a importance value of zero. This makes sense, since a single tree cannot incorporate all the predictors. Most of the predictors in the other models have no zero-importance values. There is no obvious pattern on which predictors are important and which predictors are unimportant that applies to all the models.

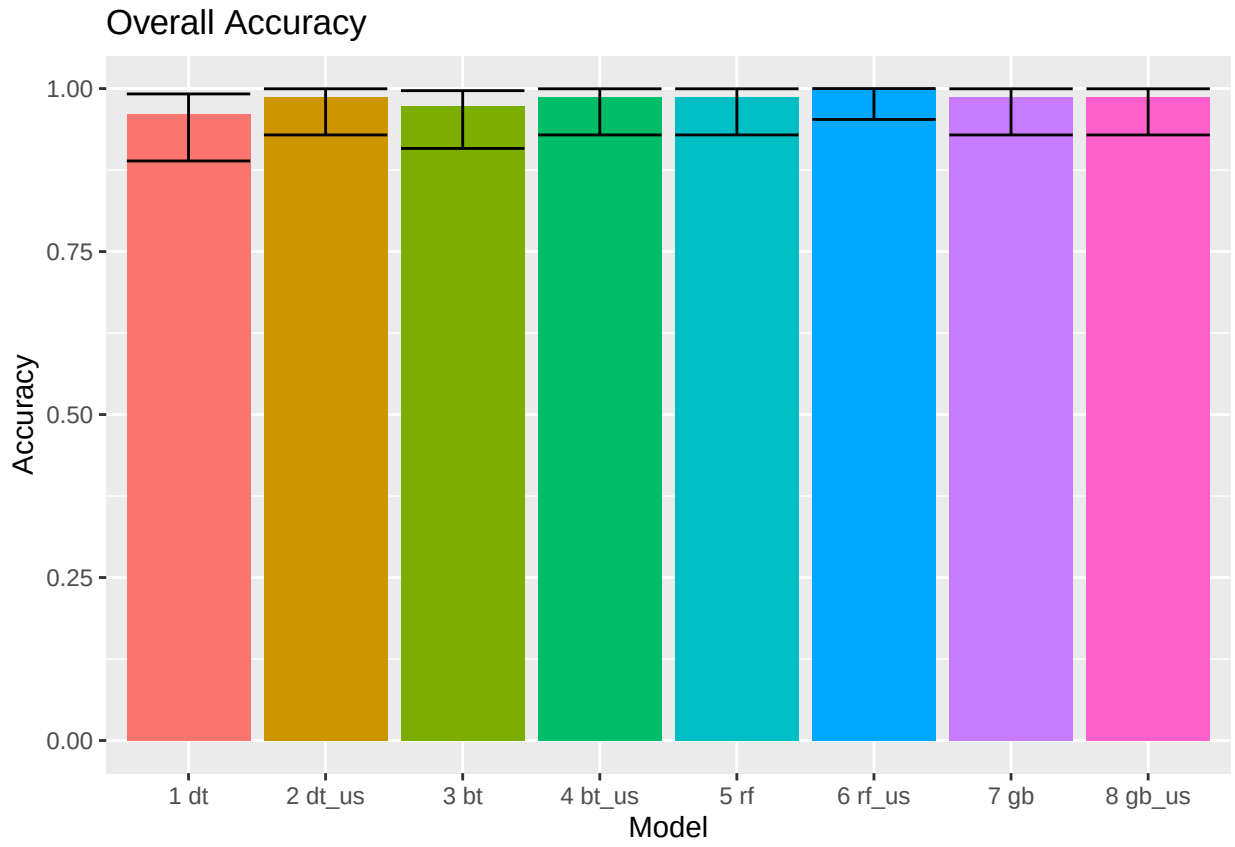
Some predictors (such as C11 family history) are universally unimportant. Some predictors (such as H20 rete ridges clubbing) are highly important in the majority of models. Some predictors (such as H12 melanin incontinence) are relatively important in some models and relatively unimportant in other models. There are rarely any predictors that are universally important in all models. This suggests that several predictors can differentiate between different diagnoses equally efficiently. This also suggests that some predictors may be collinear and interchangeable with other predictors. These hypotheses are supported by the boxplots from the initial exploratory analysis (section 2.3.3).

3.4 Results: Predictions on the Hold-Out Test Set

The tuned final models produced from each of the algorithms are used to predict the diagnosis classes in the test set. A confusion matrix is then constructed by comparing the predicted diagnosis and the actual diagnosis for each observation in the test set. The performance achieved with the test set is the model's true performance, since the test set is not used to train or tune any of the models.

3.4.1 Overall Accuracy The overall accuracy values for all the models are shown below.

models	Accuracy	Kappa	AccuracyLower	AccuracyUpper
1 dt	0.9605263	0.9505423	0.8889467	0.9917844
3 bt	0.9736842	0.9670496	0.9081505	0.9967970
5 rf	0.9868421	0.9835248	0.9288563	0.9996669
7 gb	0.9868421	0.9835248	0.9288563	0.9996669
2 dt_us	0.9868421	0.9835711	0.9288563	0.9996669
4 bt_us	0.9868421	0.9835711	0.9288563	0.9996669
6 rf_us	1.0000000	1.0000000	0.9526212	1.0000000
8 gb_us	0.9868421	0.9835248	0.9288563	0.9996669



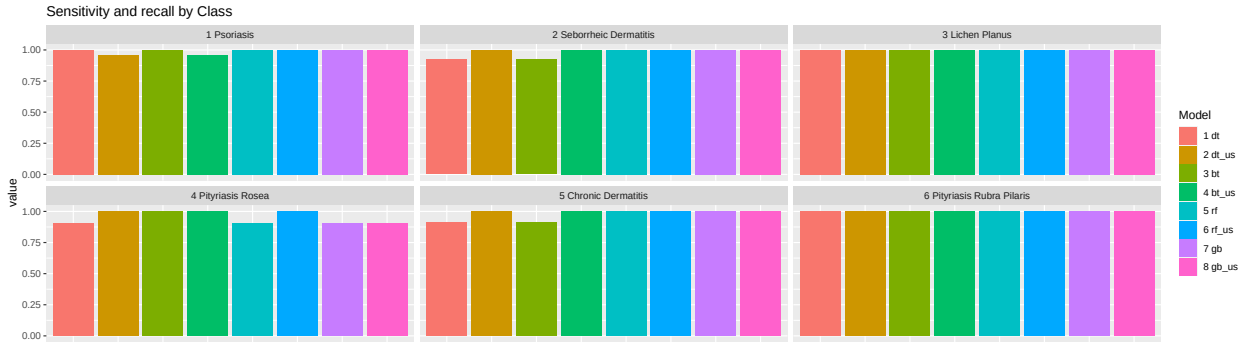
The overall accuracy of the predictions on the test set is very similar between all the models. The up-sampled random forest model has perfect accuracy. The accuracy of most of the other models are near perfect with the exception of the decision tree and the bagged tree models which are slightly lower. Although the test set is 20% of the data, the number of observations in the set is still relatively low. Therefore, the accuracy values may change if a larger sample is available.

3.4.2 Accuracy by Diagnosis Class To view the performance of the models for each individual diagnosis class, several performance metrics are extracted from the confusion matrices. The metrics are balanced accuracy, specificity, F1 score, precision, and sensitivity & recall.

All the classes have high balanced accuracy values in most of the models. Only pityriasis rosea has slightly lower values. This suggests that all the models are pretty good at differentiating between the different classes. However, there are still some misclassification, especially by the decision tree and the bagged tree models.

The F1 score takes prevalence into account. The F1 plots are similar to the balanced accuracy plots. The major difference is the lower values for pityriasis rubra pilaris when using the unbalanced decision tree and the bagged tree models. This is expected since the class pityriasis rubra pilaris has the lowest prevalence. This plot also suggests that balancing the classes has the greatest impact when training the decision tree or the bagged tree algorithms.

The sensitivity values in half the models are quite low for pityriasis rosea. This suggests that some patients with pityriasis rosea are mislabeled as another diagnosis. In addition, the unbalanced decision tree and bagged tree models have relatively low values for seborrheic dermatitis and chronic dermatitis. This reinforces that these two models benefit from up-sampling the data.



The specificity values for all classes in all models are very high. This suggests that the diseases that are misclassified by the models may have a low prevalence.

The precision plots look very similar to the F1 plots. The values for seborrheic dermatitis is slightly lower than the other classes in all models. This suggests that patients diagnosed with seborrheic dermatitis may actually have a different disease. In addition, the unbalanced decision tree and the bagged tree models have relatively low values for pityriasis rubra pilaris. This reinforces that these two models benefit from up-sampling the data.

3.4.3 Prediction Errors The specific prediction errors made on the test set can be extracted from the confusion matrix. In these tables, 1 is psoriasis, 2 is seborrheic dermatitis, 3 is lichen planus, 4 is pityriasis rosea, 5 is chronic dermatitis, and 6 is pityriasis rubra pilaris. The columns are the true diagnosis classes. The rows are the predicted diagnosis class.

The up-sampled random forest model predicted all the patients corrects. Most of the other models only made one error. The unbalanced decision tree and bagged tree models made three and two errors respectively. The most common errors are mislabeling chronic dermatitis as psoriasis, mislabeling pityriasis rosea as seborrheic dermatitis, or mislabeling seborrheic dermatitis as pityriasis rubra pilaris. This is consistent with the statistical metrics by class (section 3.4.2). The class seborrheic dermatitis seems to have caused the most false positives and false negatives. This is consistent with the boxplots in the initial data visualization (section 2.3.3).

Decision Tree

	1	2	3	4	5	6
1	23	0	0	0	1	0
2	0	12	0	1	0	0
3	0	0	15	0	0	0
4	0	0	0	9	0	0
5	0	0	0	0	10	0
6	0	1	0	0	0	4

Bagged Tree

	1	2	3	4	5	6
1	23	0	0	0	1	0
2	0	12	0	0	0	0
3	0	0	15	0	0	0
4	0	0	0	10	0	0
5	0	0	0	0	10	0
6	0	1	0	0	0	4

Random Forest

	1	2	3	4	5	6
1	23	0	0	0	0	0
2	0	13	0	1	0	0
3	0	0	15	0	0	0
4	0	0	0	9	0	0
5	0	0	0	0	11	0
6	0	0	0	0	0	4

Gradient Boost

	1	2	3	4	5	6
1	23	0	0	0	0	0
2	0	13	0	1	0	0
3	0	0	15	0	0	0
4	0	0	0	9	0	0
5	0	0	0	0	11	0
6	0	0	0	0	0	4

Up-Sampled Decision Tree

	1	2	3	4	5	6
1	22	0	0	0	0	0
2	1	13	0	0	0	0
3	0	0	15	0	0	0
4	0	0	0	10	0	0
5	0	0	0	0	11	0
6	0	0	0	0	0	4

Up-Sampled Bagged Tree

	1	2	3	4	5	6
1	22	0	0	0	0	0
2	1	13	0	0	0	0
3	0	0	15	0	0	0
4	0	0	0	10	0	0
5	0	0	0	0	11	0
6	0	0	0	0	0	4

Up-Sampled Random Forest

	1	2	3	4	5	6
1	23	0	0	0	0	0
2	0	13	0	0	0	0
3	0	0	15	0	0	0
4	0	0	0	10	0	0
5	0	0	0	0	11	0
6	0	0	0	0	0	4

Up-Sampled Gradient Boost

	1	2	3	4	5	6
1	23	0	0	0	0	0
2	0	13	0	1	0	0
3	0	0	15	0	0	0
4	0	0	0	9	0	0
5	0	0	0	0	11	0
6	0	0	0	0	0	4

4 Conclusion

4.1 Conclusion: Summary

The dermatology dataset was collected from patients that have one of six erythemato-squamous diseases (skin conditions). There are thirty four predictors to determine the diagnoses for each patient. Four tree based algorithms are used to build and tune models that can categorize the patients to a diagnoses class. Since the prevalence between the classes are not balanced, both the raw training data and up-sampled training data are used to tune the models. The result is eight tuned models. All eight models are used to classify the patients in the validation dataset.

The random forest and gradient boost models were the most accurate. The decision tree and the bagged tree models made slightly more errors. This makes sense since random forest and gradient boost are the most sophisticated ensemble algorithms, while decision tree and bagged tree are much simpler algorithms. However, the performances for the decision tree and bagged tree algorithms are dramatically improved if the classes in the training data are balanced with up-sampling. The most difficult diagnosis to classify is seborrheic dermatitis, as it caused both false positives and false negatives. To produce the most accurate result, the random forest or gradient boost algorithms should be used. If computational power and time is a constraint, then the decision tree or bagged tree algorithms with up-sampled data should be used.

4.2 Conclusion: Potential Impact

All of these models can quickly and accurately diagnose patients with erythemato-squamous diseases (skin conditions), assuming the prerequisite tests are completed. The variable importance can help identify the most useful tests to order for new patients, which can save time and money without comprising the diagnosis accuracy. Finally, the decision tree can be used as a guide to diagnose patients either as a clinical, or research, or teaching tool.

4.3 Conclusion: Limitations

These models are all trained with all thirty four predictors. Therefore, the accuracy of at least some of the models may change if not all thirty four predictors are available for an undiagnosed patient. All the observations (patients) in the dataset are known to have at least one of the six disease diagnoses. It is not clear how the models will perform on a dataset that includes healthy people with no skin conditions. The number of observations in the original dataset is relatively small, on the scale of hundreds instead of millions. This is understandable, since it is difficult to administer thirty four tests on a large number of people, or even find a large number of people with existing erythemato-squamous diseases. The small size of the dataset may increase the risk of overtraining. When tuning the models, only three to five different values are tested for each tuning parameter. Therefore the final models may not have the most optimal parameter values. Finally, the five-fold cross validation only has five repeats so that the models can be trained in a reasonable time. Therefore, the model performance may be improved with more cross validation iterations.

4.4 Conclusion: Future Works

Model performance may be improved by removing non useful predictors during preprocessing. This can be accomplished using the variable importance plots as a reference. Each of the tuning parameters can be fine tuned further and their value ranges can be increased. This will help find more optimal models especially with the gradient boost algorithm. Increasing the number of cross validation trials can improve model performance on out-of-sample data. It is possible that there are other predictors that are useful in differentiating between different skin conditions. This can be especially valuable for diagnosing seborrheic dermatitis, which does not currently have a good predictor. Other classification algorithms can be tested on

this data. For example, extreme gradient boost is an improved version of gradient boost with more tuning parameters. As another example, support vector machines have been shown to be effective in classification tasks, even though it is more complex to set up. Finally, it will be interesting to test these models on larger datasets, either datasets with more patients or datasets that include healthy individuals.

5 Citation

Boehmke, B. C., & Greenwell, B. (2020). Hands-on machine learning with R. Boca Raton ; London ; New York: CRC Press.

Güvenir, H., Demiröz, G., & Ilter, N. (1998). Learning differential diagnosis of erythemato-squamous diseases using voting feature intervals. Artificial Intelligence in Medicine, 13(3), 147-165. doi:10.1016/s0933-3657(98)00028-1

Ilter, N., Guvenir, H., Dua, D., & Graff, C. (1998, January 1). Dermatology Data Set. Retrieved from https://archive.ics.uci.edu/ml/datasets/Dermatology.

Irizarry, R. A. (2020). Introduction to data science data analysis and prediction algorithms with R. Boca Raton: CRC Press.

Kassambara, A. (2018). Machine learning essentials: Practical guide in R. Scotts Valley, Kalifornien: CreateSpace Independent Publishing Platform.

Kuhn, M., & Johnson, K. (2013). Applied predictive modeling. doi:10.1007