JOURNAL OF
EMERGING INVESTIGATORS

Article

Comparison of the ease of use and accuracy of two machine learning algorithms – forestry case study

Bhavya Bhatia[1*], Jakub Michalski[2*], Jegath Shebin[3]
[1]Nord Anglia International School, Dubai, United Arab Emirates
[2]Dubai College, Dubai, United Arab Emirates
[3]NTT Data Services Chennai
*these authors contributed equally to this work

SUMMARY

With the availability of massive amounts of data and cheap computing, machine learning has become increasingly viable to create extensive multivariate mathematical models of natural phenomena to help predict accurate future trends that would have been impossible for humans to accomplish by themselves. There is a wide variety of different machine learning algorithms available, and it is not always known which one will perform best for a given dataset. This can be determined after training and evaluating the different models and comparing them. In this case study, logistic regression and random forest models were compared in terms of accuracy and ease of use. We hypothesized that logistic regression would yield a higher accuracy and be easier to set up in a comparable scenario for a given dataset compared to random forest. Both algorithms used the same forestry dataset to see which one would outperform the other. Initially, logistic regression looked like the better choice, however, after a variety of comparisons, random forest yielded higher performance in both accuracy measurements (accuracy=0.9722, Fβ=0.9722 for random forest vs. accuracy=0.7141, Fβ=0.6990 for logistic regression) and did not require as much detailed tuning as logistic regression did.

INTRODUCTION

Machine learning can be defined in many different ways. Arthur Samuel defined it as an ability of a computer to learn without telling it precisely what to do (1). However, a more modern definition can be that of Tom Mitchell. He defined it as "the study of computer algorithms that allow computer programs to automatically improve through experience" (2). Machine learning algorithms look for patterns and correlations between different independent variables and how they affect the dependent variable. The resulting trained model will then attempt to predict the outcome of a dependent variable in a new scenario based on independent variables put into it. Machine learning promises to analyze large quantities of data in an exponentially shorter time than a human would while finding patterns that may not have been obvious to any human operator. This is the main reason for the rapid interest in and growth of machine learning, demonstrated by the spending increase from $1.58 billion in 2017 to a predicted $20.83 billion in 2024 (3).

Data scientists must choose a suitable model from a variety of available methods, which is not always an easy choice to make. The advances made in machine learning in recent years have made it easier for the average user to try out different machine learning models. Currently, one of the most popular and easy-to-use frameworks for machine learning is scikit-learn, which is based on the Python programming language (4). It offers a variety of models ready to be used and trained, including logistic regression and random forest. Its high customizability proved to be useful in this investigation, as it allows for exploratory hyperparameter tuning. This pair of models were selected for this investigation in particular as these two methods have fundamentally different approaches to predicting the values of the dependent variables. Logistic regression tries to fit a straight-line separating data points in a dataset and minimizes the cost of the function by reducing incorrect predictions (5). Random forest makes predictions based on multiple decision trees, so the method is not limited by the constraints of a straight line and can be more versatile (6). Clearly, a random forest model is completely different from logistic regression. Furthermore, logistic regression is one of the oldest and simplest machine learning algorithms. Other models were also considered, such as support vector machines, but they do not differ from logistic regression as greatly as random forests do.

This presents the question of which of the two is easier to use and more accurate. We defined ease of use as to how time-consuming and work-intensive it was to set the model up. Based on our previous knowledge, the hypothesis was that logistic regression would be easier to use than the random forest model because random forest would take too long to develop for a similar accuracy. However, this hypothesis was rejected based on a variety of measures, including accuracy and Fβ scores. Accuracy is the proportion of accurate predictions, while the Fβ score is the harmonic mean of precision (false positive rate) and recall (false negative rate), with a beta parameter specifying the weight of precision or recall. For this investigation, the beta parameter is set to give more weight to precision, which should minimize the rate of false positives.

The dataset is forestry-based, containing numerous environmental variables (7). It was chosen arbitrarily as the focus of this investigation is on the models, not the dataset. The two models will be used to predict cover types for forests based on elevation, aspect, slope, and others.

Solver (no normalization)	Accuracy	Time needed (s)
newton-cg	0.7096	628
sag	0.6959	45
saga	0.6931	58
lbfgs	0.6223	28
Solver (normalization)	**Accuracy**	**Time needed (s)**
newton-cg	0.7141	635
sag	0.7140	40
saga	0.7139	62
lbfgs	0.7003	27

Table 1. Accuracy table for an equal number of iterations for different solvers. Separate results for solvers with and without normalization of data applied.

Solver	max_iter	Accuracy	Time needed (s)
newton-cg	34	0.7141	132
sag	400	0.7141	128
saga	250	0.7141	126
lbfgs	500	0.7115	127

Table 2. Accuracy results for an equal amount of time for different solvers. The max_iter parameter controlled how long each solver took

RESULTS

The first model tested was logistic regression. The different solvers for it were compared with and without normalization (**Table 1**). Normalization improved the performance of the sag and saga solvers, taking roughly the same physical time (within random variation) for the same accuracy (about 0.71). Normalization also minimally improved the accuracy for the newton-cg solver. Surprisingly, lbfgs benefited the most from normalization. Normalization was kept for the rest of the measurements due to improvements across all solvers.

However, we kept the time supplied to each solver constant, making the results easier to compare as ease of use is one of the criteria. Newton-cg, sag, and saga offer the same accuracy (0.7141) in a very similar time frame (**Table 2**), with lbfgs giving the lowest accuracy (0.7115) for the given time. Thus, it is hard to select the best solver due to the similarity in the accuracies and time needed.

With the solver selected and normalization applied, the number of iterations was increased to see what the highest accuracy that can be obtained is. However, there were no observable improvements in accuracy in the training set after 400 iterations, and the time needed increased rapidly. Other optimizations for logistic regression will be explored depending on the results obtained by the random forest classifier.

The model does not seem to improve with more training examples (**Figure 1**), and the cross-validation score even seems to decrease at some stages. The decrease in accuracy is quite unexpected for the same data set as in general the accuracy should increase with more training examples. This points to the limitations of the model itself, not the number of training examples. After the main parameters were selected, the final accuracy achieved was 0.7141, and the Fβ score was 0.6990. We next investigated random forest in order to compare the results to this model.

Criterion	Time needed (s)	Accuracy
Gini	102	0.9598
entropy	102	0.9625

Table 3: Accuracy results for the different criteria for the same amount of time for a fair comparison.

Cover	Accuracy	Number of samples
1	0.9696	211840
2	0.9808	283301
3	0.9740	35754
4	0.8998	2747
5	0.8750	9493
6	0.9317	17367
7	0.9667	20510

Table 4: Accuracy results for each cover type as well as the number of samples for each cover type. The table presents a clear relationship between the accuracy and number of samples

The first decision for creating a model was the selection of the random forest criterion. The accuracies obtained only differ by 0.025 between the two criteria, Gini and entropy, (**Table 3**). So, the entropy criterion was used for the final model as even such a small change may result in a higher final accuracy for the model.

The learning curves show that when more data is supplied, the accuracy is increased (**Figure 2**). The training score was compared to the cross-validation score. As expected, the cross-validation scores were lower than the training scores. Hence, using more data points would increase accuracy, but it is not a valuable use of time because it could be used for other purposes (e.g., exploring alternative models or more hyperparameter tuning).

The accuracy consistently increased with more decision trees (**Figure 3**) up until about 50 trees, where it leveled off. Using more than a hundred decision trees provides negligible improvements, with no improvements observed after 400 decision trees.

Next, hyperparameter tuning was performed to see its benefits. This was done as random forest already had better baseline results than logistic regression, so, a decision was made to try and improve it with hyperparameter tuning. As a baseline, a RandomForestClassifier was run several times

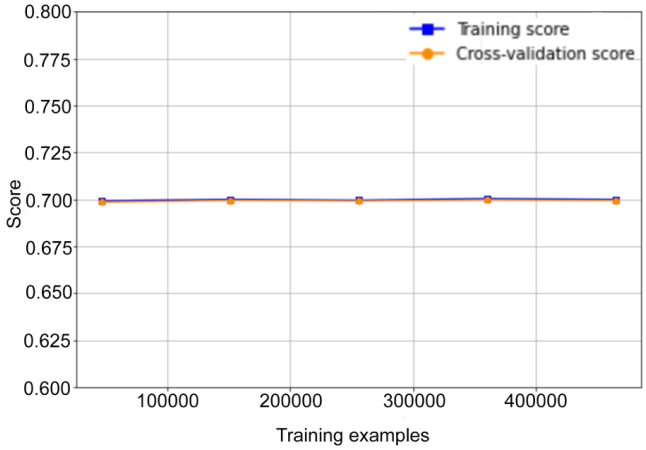

Figure 1: Learning curve for the sag solver (logistic regression). Accuracy plotted against the number of training examples supplied, which shows how the model can become more accurate with more data provided to it.

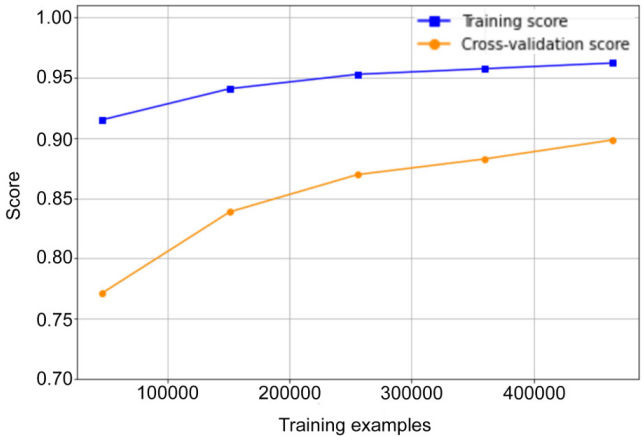

Figure 2: Learning curve for random forest. Accuracy plotted against the number of training examples supplied, which shows how the model can become more accurate with more data provided to it.

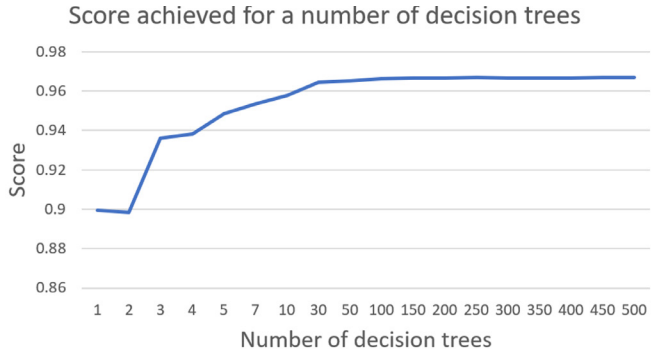

Figure 3: Accuracy plotted against a given number of decision trees. More decision trees should result in a higher accuracy as it reduces the variance by averaging the results from several trees, preventing overfitting.

with only the random state set to 0 and no other parameters set, and its accuracy was on average 0.9635. This result was then compared to the results from the hyperparameter tuning.

The final accuracy was on average 0.9720, which is only a marginal improvement over the baseline; however, it is still an improvement as all of the final accuracy tests yielded higher results than any of the baseline tests. For a more objective measure of accuracy, the Fβ score was used to assess the final model, as it accounts for any imbalances in the dataset, which is the case in this investigation. An accuracy value of 0.97217 was achieved. This means that random forest was more accurate than logistic regression in both measures.

When compared to the distribution of the cover types in the dataset (**Table 4**), a clear trend can be seen: the more training examples there were for a cover type, the higher the accuracy for that cover type was.

Overall, the baseline random forest model (0.9635) had significantly greater accuracy than the logistic regression model (0.7141) even with the solvers investigated. Furthermore, hyperparameter tuning provided a significant increase in accuracy from the baseline model (0.97217). To increase the performance of logistic regression, significantly more effort would have to be put into the model and the data itself, which goes against the ease of use criterion set out in this investigation while comparing the two models. The results lead to the rejection of the hypothesis.

DISCUSSION

When faced with a new dataset, such as the forestry one in this investigation, it is often difficult to decide which model to use. Although each dataset is unique, the basic use of models for multivariate classification is roughly the same for each case. The aim of this investigation was to determine which method would be easier to use while also yielding high accuracy. In the beginning, higher accuracy results were expected from logistic regression, however, it massively underperformed in comparison to random forest in both of the criteria that were set (accuracy and ease of use).

In our investigation, random forest needed less pre-processing of the data for it to yield high accuracy. A logistic regression model would need much more manual data pre-processing, such as feature selection including multicollinearity analysis to reach comparable accuracy levels (8). As a random forest uses a subset of the available features for each tree, it may remove features that are highly correlated to each other, increasing the overall accuracy. For a logistic regression model, this correlation analysis would have to be done manually and before the training. Not doing that may make the learning of the model slower or introduce harmful bias. Using a random forest prevents these issues. Furthermore, logistic regression attempts to fit a linear model to the data. A model may perform better if it is a polynomial model, which can be done by squaring one of the features, for instance. All these issues must be resolved manually for logistic regression and they do not concern random forest.

The dataset used in this investigation was relatively large with a small number of features. The accuracies for the different solvers (**Table 1**), considering the number of iterations, accuracy, and time needed, can then easily be explained for this dataset. In the case of newton-cg, it uses an exact Hessian matrix to optimize the model, which explains

	Elevation	Aspect	Slope	Horizontal hydrology	Vertical hydrology	Horizontal roadway	Hillshade 9am	Hillshade noon	Hillshade 3pm	Horizontal fire	Cover	Soil Type	Area
Elevation	1.000000	0.015735	-0.242697	0.306229	0.093306	0.365559	0.112179	0.205887	0.059148	0.148022	-0.269554	0.689848	-0.315590
Aspect	0.015735	1.000000	0.078728	0.017376	0.070305	0.025121	-0.579273	0.336103	0.646944	-0.109172	0.017080	0.007149	0.138703
Slope	-0.242697	0.078728	1.000000	-0.010607	0.274976	-0.215914	-0.327199	-0.526911	-0.175854	-0.185662	0.148285	-0.105571	0.286178
Horizontal hydrology	0.306229	0.017376	-0.010607	1.000000	0.606236	0.072030	-0.027088	0.046790	0.052330	0.051874	-0.020317	0.194685	0.056339
Vertical hydrology	0.093306	0.070305	0.274976	0.606236	1.000000	-0.046372	-0.166333	-0.110957	0.034902	-0.069913	0.081664	0.071313	0.189085
Horizontal roadway	0.365559	0.025121	-0.215914	0.072030	-0.046372	1.000000	0.034349	0.189461	0.106119	0.331580	-0.153450	0.240829	-0.445922
Hillshade 9am	0.112179	-0.579273	-0.327199	-0.027088	-0.166333	0.034349	1.000000	0.010037	-0.780296	0.132669	-0.035415	0.040877	-0.233455
Hillshade noon	0.205887	0.336103	-0.526911	0.046790	-0.110957	0.189461	0.010037	1.000000	0.594274	0.057329	-0.096426	0.013662	-0.080767
Hillshade 3pm	0.059148	0.646944	-0.175854	0.052330	0.034902	0.106119	-0.780296	0.594274	1.000000	-0.047981	-0.048290	-0.001122	0.105050
Horizontal fire	0.148022	-0.109172	-0.185662	0.051874	-0.069913	0.331580	0.132669	0.057329	-0.047981	1.000000	-0.108936	0.100952	-0.416830
Cover	-0.269554	0.017080	0.148285	-0.020317	0.081664	-0.153450	-0.035415	-0.096426	-0.048290	-0.108936	1.000000	-0.164337	0.275464
Soil Type	0.689848	0.007149	-0.105571	0.194685	0.071313	0.240829	0.040877	0.013662	-0.001122	0.100952	-0.164337	1.000000	-0.271883
Area	-0.315590	0.138703	0.286178	0.056339	0.189085	-0.445922	-0.233455	-0.080767	0.105050	-0.416830	0.275464	-0.271883	1.000000

Figure 4: Correlation table for the variables included in the dataset. The correlation between any two variables was calculated to determine which variables contribute the most towards the dependent variable and which variables are collinear.

the small number of iterations needed to converge as well as the high accuracy; however, it also makes the calculations more computationally expensive. The key difference between newton-cg and lbfgs is that lbfgs uses an approximate Hessian matrix, making it less computationally expensive. However, this explains the need for more iterations. It does not generally perform well on larger datasets.

Sag works differently to newton-cg and lbfgs. It minimizes finite sums of convex functions and it is one of the fastest solvers for large datasets (number of data points and the number of features is both large). As our dataset does not have many features, this might be the reason for the lower-than-expected performance. However, the dataset had many data points, meaning it performed better than lbfgs. Saga works similarly to sag: it is a variation of sag which supports L1 regularization. It supports sparse datasets (the one in the case study was not), as well as very large data sets (the one in the case study was not). Although secondary sources seem to suggest that saga should, in general, be faster than sag, the empirical evidence does not support such a conclusion for the case study, which may be due to the reasons outlined.

A random forest without any hyperparameter tuning was already much more accurate than any of the above solvers, and so that was the model with more tuning put into it. This is why random forest was focused on more during the end of the investigation.

The models tested in this investigation were chosen because they are fundamentally different. However, a similar investigation can be carried out on any two other models, such as support vector machines and neural networks, which are also two very different approaches. Neural networks promise to increase the accuracy obtained, as they can combine many features to produce more features by themselves, which also means that they can be easy to use. However, this discussion is beyond the scope of this investigation.

METHODS

Dataset

For this investigation, we used a forestry dataset (7). This dataset was used to predict cover types for forests using cartographical variables. 30x30 meter blocks of these forests are used for these cover types as this can be more easily controlled. These 30x30 meter areas also represent forests with minimal human-caused disturbances, so that existing forest cover types are more a result of ecological processes rather than forest management practices. The independent variables were derived from data originally obtained from the US Geological Survey (USGS) and US Forest Service (USFS) data. Data is not scaled and contains binary columns for qualitative independent variables (e.g., wilderness areas and soil types).

The variables included in the dataset included information such as: Elevation, Aspect, Slope, Horizontal and vertical distance to hydrology, Horizontal distance to roadways, Hillshade indexes at different times of the day, Horizontal distance to fire points, Wilderness area (4 binary columns), Soil type (40 binary columns), Cover type (7 types). More detailed descriptions of each variable are included in the dataset (9).

Data pre-processing and pre-analysis

The data was supplied in a comma-delimited data format. Columns were labeled with their corresponding names, and

values were identified as either continuous or discrete. The file was then exported into a .csv format to conform with Pandas (Python data framework library).

Soil type columns were merged into a single column with 40 different categories possible, labeled from 0 to 39. This was done via a short script which first detected the number of the column selected (which indicated which soil type the area has) and then entered that number into a new column. The same process was repeated with the wilderness area: merging 4 binary columns into one label column.

The correlation values between all of the variables in the dataset were calculated (**Figure 4**). This allowed us to determine the variables that most significantly impacted the cover type, which were: elevation, slope, horizontal roadway, horizontal fire, soil type, and area. The highest correlation value was 0.2755 between cover and area, which is not a high value, implying that none of the independent variables are highly correlated to cover. For training, all of the independent variables were used, as there was a small number of them, and none were highly correlated to cover. Reducing the number of independent variables could affect the methodology due to underfitting or having a lower accuracy in general.

The data is not equally distributed between the cover types, with type 2 being the most common and type 1 being slightly behind. The frequency of type 1 is two orders of magnitude larger than of type 4 (211840 vs 2747 data points), which skewed the results. So, this needed to be accounted for in the models to get the fairest and most accurate results as possible.

The data was randomly split into training and test data sets in a ratio of 70:30. This ratio was decided based on the size of the dataset (m=581013), which allowed for a large proportion of the examples to be left for testing purposes. Both models used the same sets of data to ensure that the variance was low, and any anomalies were shared across both, so neither would have an advantage over the other during testing. Using training data to assess the performance of the algorithms would not yield reliable results as it would not provide an accurate assessment of performance on unseen data.

Multivariate logistic regression

To test our hypothesis, we need to determine both how accurate and easy to use each model is for predicting the cover type in the forestry dataset. The first model tested was logistic regression and how different solvers for it performed. The tests were implemented to find the accuracy of each solver, while also noting the time taken for the accuracy to reach a certain threshold for a fair comparison. This was done by first inputting the train and test data to find the respective accuracies with a limited number of iterations (7). Then the accuracies were measured after a certain time to determine which solver yielded better accuracies given the time constraints. These limitations were set to keep variables constant and to check for the efficiency of each solver with the data set, which ultimately concluded which would be more accurate and easier to use.

Each model in scikit-learn has a random state parameter that is used as the seed for the random number generator. Throughout the whole investigation, it was set to the same value. This ensures that any improvements observed are not due to random variation, but only due to the changes made to the model.

Python's sklearn logistic regression model has a variety of different algorithms for gradient descent. In the documentation, it is recommended to use newton-cg, sag, saga, or lbfgs (liblinear is possible, however, limited to one-versus-rest models). The different solvers were benchmarked against each other to compare their performance in terms of speed and accuracy. To provide reliable results, the number of iterations was limited to a fixed number of 100 epochs. The measurements were then repeated after normalizing the data, as sag and saga require normalization for optimal performance (5). The results for each solver are summarized in **Table 1**.

For the same number of iterations, the time taken for the optimization to complete varied greatly, and even though these results give some insight as to which solver is the most appropriate one for this case study, the control variable chosen was not giving comparable results because more time (and so computational power) to optimize the data naturally results in higher accuracy. This is why the control variable was switched to the time needed (around 127 seconds given for each solver, chosen arbitrarily) as it would give a better measure to compare the solvers.

No solver had an obvious advantage over the others in terms of accuracy, so sag was used as the solver for the final model due to the scalability for large datasets that it promises. There is no need to use saga in this investigation due to the limited number of features and no missing data points.

The train and test scores for a different number of training examples were very close to each other or even the same in some cases (**Figure 1**). Cross-validation was performed over 10 different splits of the part of the dataset under consideration (dependent on the number of training examples for each point), with 80% of the dataset going towards training and 20% towards testing for each split, in contrast to the training-testing split. As the dataset is large, more data points can be used for training in this case, and less for testing. This smoothed out any variation in the data supplied to the model and provided a more reliable method of assessing the accuracy of the model. The cross-validation score was calculated from the data that the model had not seen before, which better reflects a real-life scenario.

Random forest classifier

A random forest classifier is a collection of decision trees that are fit on sub-samples of the whole dataset and averages their results to improve accuracy. Each tree continuously splits its data into smaller and smaller groups according to the values of their parameters, until a result is obtained from

a leaf node. A random forest was used instead of a decision tree due to how easily a decision tree can overfit the data (high variance), yielding a higher accuracy while training but a lower accuracy in test data. A random forest negates this tendency by training a specified number of decision trees, which 'spreads out' the results to several different trees. Like logistic regression, the random forest model was created using sklearn's ensemble package.

The Gini coefficient measures the inequality among values of a frequency distribution. It can also be replaced by an entropy measure, which is the information gain. So, the impact of the criterion on the given accuracy for a set time was measured.

The accuracy for a different number of training examples was then investigated (**Figure 2**), which showed promising results. Cross-validation scores were calculated in the same way as in the learning curves for logistic regression.

As a random forest consists of several decision trees, the relationship of the number of trees to the accuracy was graphed (**Figure 3**), which showed an increase in accuracy up to a certain number of trees. This helped determine the value of the number of estimators parameter in hyperparameter tuning.

Hyperparameter tuning

As random forest was used for the final model, the values of the parameters available were tweaked (6):

- n_estimators – It is the number of decision trees that need to be trained. The more, the better; however, it is also more computationally expensive. The selected value was 400, as values above this one gave negligible improvements to the accuracy of the model.
- criterion – It is the function measuring the quality of the split. The data gathered (Table 3) about the performance of each criterion suggests a higher accuracy while using this entropy criterion.
- max_depth – It is the measure of how deep a tree can go before a leaf node is reached. A value of 'None' means the tree will be expanded until the leaves are pure (or have fewer than min_samples_split samples, if specified). This was the optimal value found as it did not produce overfitting in the testing set while giving the model a large depth, preventing underfitting.
- min_samples_split – It is the minimum number of samples needed to split a node. A value of 2 was chosen as this is a good in-between value (and recommended by scikit-learn as the default value). This value lies in the ideal range of 1 to 40, as found by empirical methods (10).
- min_samples_leaf – This is the minimum number of samples needed to be a leaf node. The value of 1 was chosen because a split must result in at least one sample being left. Using any higher values may decrease the accuracy. This is also the value recommended by scikit-learn.
- min_weight_fraction_leaf – This is only applied if the samples are given weights. It comes from the sum of weights of samples at a leaf node and is the minimum weighted fraction of those. For our model, this value was kept at 0.0, as each class has an equal weight.
- max_features – This is the measure of how many features can be considered for a split. It is made to be 'auto' as this value makes the max_features equal to the square root of the number of features as this is a good in-between value.
- max_leaf_nodes – This is the maximum number of leaf nodes a tree can have, which reduces impurity. The value of 'None' has been kept, as this won't limit the number of leaf nodes; it is recommended by scikit-learn.
- min_impurity_decrease – a node can only be split if the split decreases the impurity of a node by that amount or more. It is kept at 0.0 (recommended value), as specifying a minimum decrease in impurity may increase the accuracy; however, it may also cause overfitting.
- bootstrap – This is used in order to determine whether the bootstrap samples need to be used when training a tree. The final model kept this value as 'False', as doing so may increase overall accuracy at the cost of an increase in time needed for training.
- oob_score – This determines whether out-of-bag samples are used to estimate the accuracy. The final model kept this value as 'False', as the final model does not need to estimate the generalization accuracy
- n_jobs – This is the number of jobs that need to run in parallel. A value of -1 was used as this causes all processes to be used, optimizing the model faster.
- random_state – This is used in order to control the randomness when training a tree (keeps random choices of e.g., sampling of features) the same for each optimization of the tree. The final model kept this value as at '0' for a fair comparison between the models.
- verbose – This controls verbosity, The final model kept this value as at the recommended value of 0.
- warm_start – This is used to determine whether to reuse the model from the previous call of the model and add to it. This was not used for the final model, so a value of 'False' was specified.
- class_weight – This is the weight associated with the classes in the data. The final model kept this as not specified, as all our classes have equal weight.
- ccp_alpha – this is used for Minimal Cost-Complexity Pruning, it is the complexity parameter. This value was kept at 0.0 as the final model will not perform pruning.
- max_samples – (if bootstrap is 'True') This is how many samples to get from the training dataset for each estimator. As the bootstrap is kept at 'False', it is not needed.

Final model

The final model included all the parameter values decided in hyperparameter turning. A detailed breakdown of the accuracies obtained is presented in **Table 4**.

ACKNOWLEDGMENTS

We would like to thank the NTT Data Chennai IT team for teaching us machine learning as well as for helping us write the paper.

Received: August 24, 2020
Accepted: February 6, 2021
Published: March 21, 2021

REFERENCES

1. Samuel, A. L. “Some Studies in Machine Learning Using the Game of Checkers.” *IBM Journal of Research and Development*, vol. 3, July 1959, pp. 210–229.
2. Mitchel, T. M. Machine Learning. Vol. 1, *McGraw-Hill*, 1997.
3. Desmond07/23/2019, Michael. “Machine Learning Market to Grow to Nearly $21 Billion by 2024.” Pure AI, 23 July 2019, pureai.com/articles/2019/07/23/nwsdes-machine-learningmarket-growth.aspx, Accessed 16 December 2019.
4. Pedregosa, F. *et al.* "Scikit-learn: Machine Learning in Python". *Journal of Machine Learning Research* 12. (2011): 2825–2830.
5. Documentation of logistic regression: F, Pedregosa. “Sklearn.linear_model.LogisticRegression.” Scikit, *Journal of Machine Learning Research*, Dec. 2011, scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html, Accessed 4 January 2020.
6. Pedregosa, F. “Sklearn.ensemble.RandomForestClassifier.” Scikit, Dec. 2011, scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html, Accessed 4 January 2020.
7. Dua, D. "UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]." *Irvine, CA: University of California, School of Information and Computer Science*. n.p.: n.p., 12 . 22 Jan. 2021, Accessed 17 January 2020.
8. Midi, Habshah, et al. “Collinearity Diagnostics of Binary Logistic Regression Model.” *Journal of Interdisciplinary Mathematics*, vol. 13, no. 3, 2010, pp. 253–267., doi:10.1080/09720502.2010.10700699.
9. Descriptions and types of variables included in the dataset: “Https://Archive.ics.uci.edu/Ml/Machine-LearningDatabases/Covtype/Covtype.info.” Index of /Ml/MachineLearning-Databases/Covtype, archive.ics.uci.edu/ml/ machine-learning-databases/covtype/, Accessed 17 January 2020.
10. Mantovani, Rafael Gomes Mantovani Gome, et al. “An Empirical Study on Hyperparameter Tuning of Decision Trees". 5th *Brazilian Conference on Intelligent System - BRACIS 2016*.”