Assignment 2 – Group 47

Tutors: Kelvin Hsu, Anthony Tompkins

Group members:

Jikang Wang – jwan7835,

Yi Huang – huyi2823,

Kwon Nung Choi – kcho7166

Abstract

Our report investigates on adult data set to predict whether one’s annual income is over 50k or less than 50k using logistic regression, decision tree, random forest, XGBoost and ensembled model. The result shows that XGBoost has highest accuracy with decent running time. Ensembled model also performs well. We conclude that XGBoost has more robust power in binary prediction problem over random forest and logistic regression algorithms.

Introduction

By using the adult data set from 1994 Census database, we aim to predict whether one’s annual income is over 50k or less than 50k. 14 attributes are provided to make the binary prediction. This work is important as technically, the adult data set is an ideal data set for training and comparing classification algorithms. Numerous past papers have tested algorithms on this data set so our work on different algorithms will be comparable and convincing. From a commercial and social perspective, the work is meaningful and insightful for assisting work in income check for loan assessment, insurance underwriting, social benefits, pensions assessment, etc.

Methods

**Computer hardware/software specifications**

The working environment is using Google Colab, the VM used for Collaboratory equipped with 13GB RAM and 2 vCPU 2-core Xeon 2.3GHz 13GB RAM 33GB HDD.

Instruction on how to run code

The code file is a ipynb file, which is created using Google Colab. So, it can be run in Colab or Jupyter Notebook environment.

It should be noted that the data is imported through URL, so there is no need to download and store data manually. Just running the whole chunks of ipynb file is enough. The total running time for whole notebook is less than 10 minutes.

**Previous work on Adults dataset**

The Adults dataset is a popular dataset for classification tasks, whereby numerous research papers have used it to investigate and evaluate the performance of classifiers. For instance, Zadrozny(2004) used the Adults dataset to investigate how some well-known classifier learning methods such as Naive Bayes, logistic regression, C4.5—an algorithm by Ross Quinlan for generating a decision tree—and SVMLight—an implementation of Support Vector Machines (SVM) in the programming language C—are affected by sample selection bias problem by artificially created by stimulating biasedness[[1]](#endnote-1). Similarly, Keerthi and Lin (2003) used the same dataset to analyse the behaviour of SVM classifier when its hyperparameters such as the penalty parameter *C* and kernel width *σ* take extremely small or large values[[2]](#endnote-2). Furthermore, Cerquides and de Màntaras (2003) applied a combination of Tree Augmented Naïve Bayes (TAN) with maximum a posteriori (MAP) probability on the Adults dataset to show improvement in accuracy and error rate when compared to TAN based classifiers with same complexity[[3]](#endnote-3). Therefore, the popularity of the Adults dataset for various classification tasks also deem it an appropriate dataset for the purpose of the present paper.

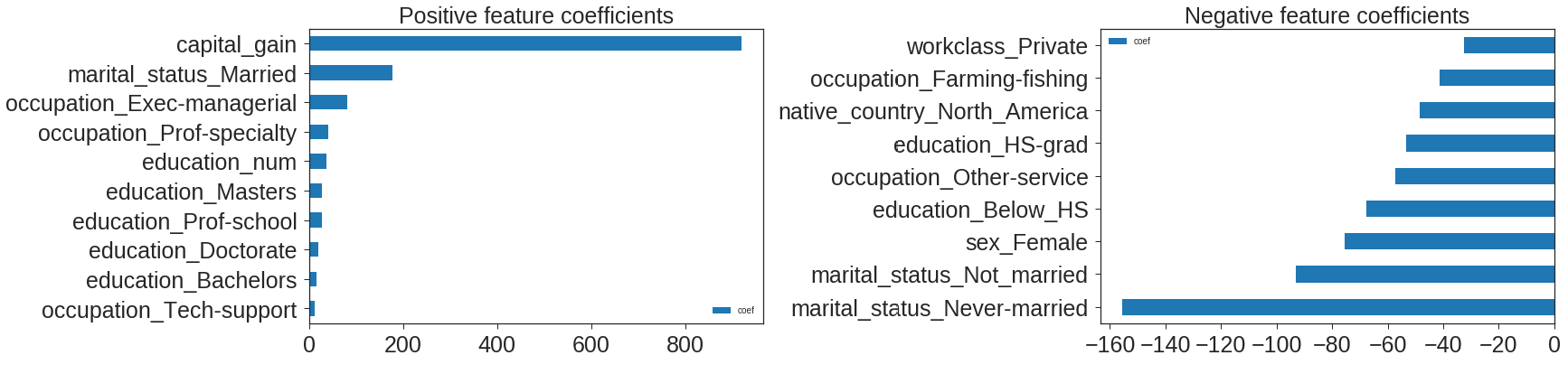
Data preprocessing

The dataset is relatively clean as there is already some data cleaning done by previous people. However, some refining works are done, including adding column labels, drop nans, transforming categorical data to numeric data or dummy values.

In exploring reasonability and redundancy of data, most data show reasonable value and range, except for capital gain column, whose maximum value is capped at 9999. We interpret this as actual capital gain value is too high and for convenience of presentation they are capped at 9999. Considering that we do classification task, this capped value has no actual impact. Since the data is from census data, the column relationship and marital\_status contains similar information, thus column relationship is dropped.

In feature engineering, for some input variables with too many values, aggregation of values is done, e.g. countries names can be combined based on continents.

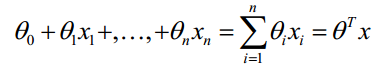
Feature contribution exploration: A simple SGD logistic classifier is run to explore the coefficients of input variables. Positive feature coefficients show that capital\_gain is the highest contributor to income >50k, followed by married marital status, occupation being executive managers and specialty professors, and number of years of education. Negative feature coefficients show that never and not married contributed most to income<50k, followed by being female, education below high school.

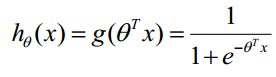


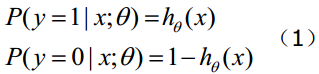
Method 1 - Logistic regression

Previous work: For dichotomous outcome prediction, initially OLS regression was used to handle. However, its shortcoming is obvious in strict assumptions that do not generally fit in binary classification data. in late 1960s logistic regression was introduced and became widely used[[4]](#endnote-4). Its popularity was also enhanced through installation in statistical packages (Peng CY, Lee KL, Ingersoll GM)

As a traditional and simple model, logistic regression is used here, which can also serve as a baseline model. The core in logistic regression is the transfer of linear output value to probabilistic value using sigmoid function. The whole process is defined as h-theta(x).





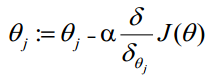


After forward propagation, loss function using cross entropy loss is calculated, which is the negative average of ground truth times log of predicted probability.

http://img.my.csdn.net/uploads/201407/16/1405496926_1023.png

http://img.my.csdn.net/uploads/201407/16/1405496967_1305.png

For backward propagation, the derivative of cost function to parameter is calculated using chain rule. The parameters are then updated with a learning rate.



**Method 2 -** **Decision tree**

Previous work: Considering the number of features are below 15, decision tree would be an ideal method to include inductive reasoning and make reasonable predictions. Also, a major advantage for decision tree is that it can handle both numeric and categorical data. Compared to other black-box algorithms (neural network), decision tree enables logical and interpretable explanation of input variables; the exhaustive split of trees also ensures algorithm convergence[[5]](#endnote-5).

However, decision tree usually does not perform as well in non-linear data and can be influenced by outliers. Also, decision tree is generally more applicable when the data is categorical.

Considering sklearn’s decision tree classifier is capable of tuning multiple parameters, it is used. Under sklearn’s CART algorithm, two classification algorithms are used.

The entropy theory chooses the way of splitting the node when most information gain is achieved. This is by using the entropy before splitting and dividing weighted average entropy after splitting.

Gini:

*: total number of samples;*

Under Gini method, the smaller Gini(D) is, the purer dataset D is. After D is split into D1 and D2, there will be multiple values of F. we choose the F with the least Gini(D, F), so that delta-Gini(F) becomes the largest value.

A major unique method for decision tree to avoid overfitting is pruning[[6]](#endnote-6).

Pre-pruning: prevent decision tree from over-growing if the tree is not improving on its generalization capability. The node will stop growing and be labeled as leaf. This is practically done through:

1. When the depth of tree reaches the max\_depth specified.
2. When the number of nodes reaches max\_leaf\_nodes specified.

Post- pruning: prune unnecessary nodes after the tree is completely trained, using majority class criterion to replace sub trees with nodes. Done through:

1. Reduced error pruning: use a validate set to test the error of simplifying original tree. And return the simplified tree with most reduced error.
2. Prune tree based on cost-complexity function (used by CART algorithm)

**Method 3 -** **Random forest**

Previous work: Random forest algorithm is a meta estimator. Developed by Breiman and Cutler, the random forest algorithm is a meta estimator[[7]](#endnote-7). It is a way of averaging multiple deep decision trees, trained on different parts of the same training set, with the goal of reducing the variance[[8]](#endnote-8). Random Forests have proven to be extremely flexible while producing very high accuracy scores, while overcoming the overfitting and high variance limitations of single decision trees. As a result, it has been widely used in classification tasks where high precision and recall was crucial, such as phishing email detection[[9]](#endnote-9), predicting long-term cognitive impairment following breast cancer[[10]](#endnote-10), and diagnosing Alzheimer’s disease and Lewy body dementia[[11]](#endnote-11). Given its reliability and reputation, random forest is also used for the present Adults dataset.

Subtree created by randomly sampled the data with replacement. Each subtree will be split based on tree learning algorithm. For each node the number of features will be randomly selected (by doing this the correlation between estimator is reduced), information gain method will be used to select the feature to perform binary split at each step. After training on each subtree, the new prediction will be made by majority-voting for classification problem.



Figure 1 - Random forest algorithm

**Method 4 - XGBoost**

Available as an open source package, extreme gradient boosting (XGBoost) is a scalable tree boosting system proposed in 2016 by Chen and Guestrin (2016)[[12]](#endnote-12). It is a scalable and accurate implementation of gradient boosting developed by the Distributed Machine Learning Community.

**Previous Work:** The impact of XGBoost has been widely recognised in numerous machine learning and data mining challenges. As noted by Chen and Guestrin (2016), 17 of the 29 winning solutions of a machine learning competition in 2015 called Kaggle were reported to have used XGBoost. In the same year, XGBoost was used by all top-10 winning teams of the annual Data Mining and Knowledge Discovery competition called KDD Cup. Such success of XGBoost is also reflected through its use for classification tasks such as web text classification, malware classification, Chinese Gaofen-3 satellite polarimetric SAR image classification[[13]](#endnote-13) and patient epilepsy classification[[14]](#endnote-14). For these reasons, XGBoost was selected for the present classification task on the Adult income dataset.

The base learning model of XGBoost is the decision tree. Through ensemble strategies such as bagging and gradient boosting, XGBoost attempts to solve the overfitting, low bias and high variance limitation of single decision trees caused by the tree growing very deep to learn highly irregular patterns.

Also known as Bootstrap Aggregation, the bagging algorithm increases classification accuracy by training various randomly sampled subsets of the training dataset with a single learning algorithm[[15]](#endnote-15). Random sampling with replacement results in subsets that are similar in size to the original dataset, but with duplicates and/or exclusions of the original data. As the same learning algorithm is used for training these different subsets, each training produces a new base classifier[[16]](#endnote-16). Classifier predictions under the new subset of data are combined through majority voting, and the final prediction for the ensemble is produced.

While bagging depends on randomness to obtain better performance, gradient boosting more actively obtains better performance. Gradient boostingis an example of ensemble technique, where predictors are made sequentially rather than independently. New classification is predicted after each iteration in an additive way as shown in Figure 2, so that predictions are derived from weak classifiers that successively improve over the error of previous classifiers. As explained by Georganos et al. (2018), higher weights are assigned to incorrectly classified items at the following step so that the classifier focuses on its performance in the subsequent iterations[[17]](#endnote-17). The final step then includes the aggregated improvement of all the previous modelled trees. However, one limitation of gradient boosting is the absence of a robust regularisation to guard against overfitting. XGBoost addresses this limitation by incorporating a stronger regularisation framework to the objective function.

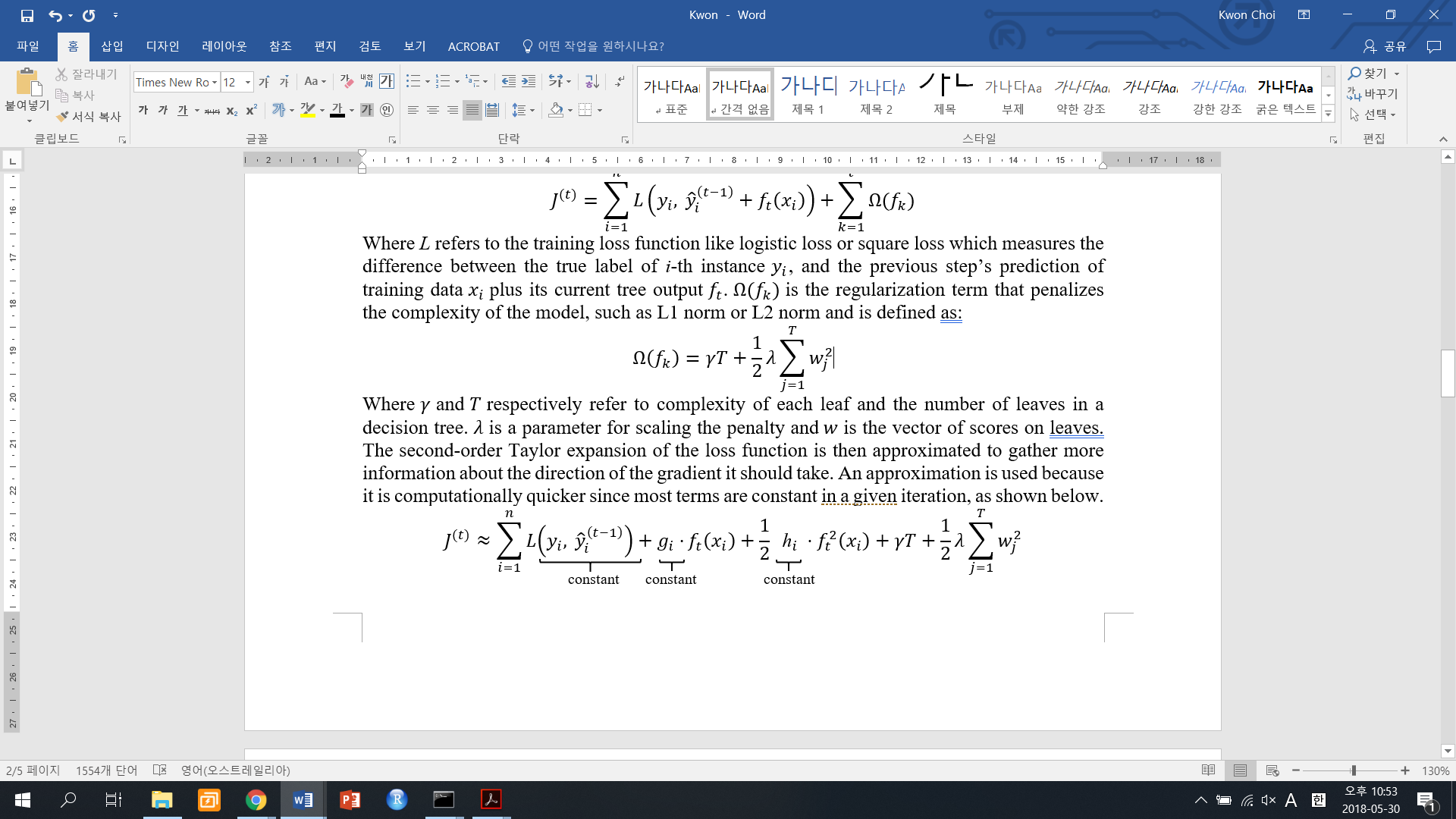


**Figure 2 -** Flow chart of gradient boosting

Given the explanations above, we now follow the explanation of Zhang et al. (2018) to mathematically define the objective function *J* in XGBoost as[[18]](#endnote-18):

Where *L* refers to the training loss function like logistic loss or square loss which measures the difference between the true label of *i*-th instance , and the previous step’s prediction of training data plus its current tree output . is the regularization term that penalizes the complexity of the model, such as L1 norm or L2 norm and is defined as:

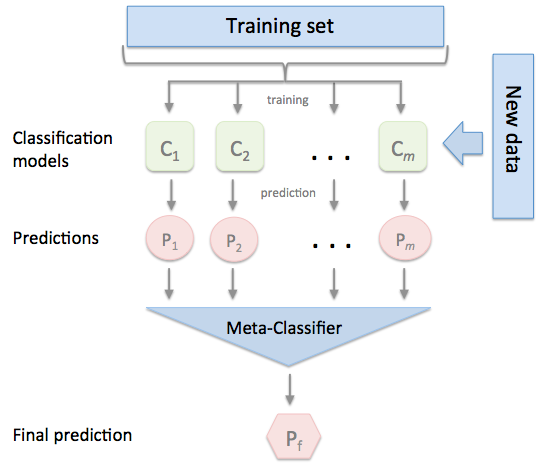
where and respectively refer to complexity of each leaf and the number of leaves in a decision tree. is a parameter for scaling the penalty and is the vector of scores on leaves. The second-order Taylor expansion of the loss function is then approximated to gather more information about the direction of the gradient it should take. An approximation is used because it is computationally quicker since most terms are constant in a given iteration, as shown below.



Where and refer to the first and second derivatives of the loss function. Therefore, only and the regularization term need to be computed. This means that changes in model performance can be evaluated using the objective function when a certain node split occurs in the decision tree. The change will be adopted only when the decision tree model performance is better after the current node split. A predictive classifier is guarded against overfitting due to regularization during objective function optimization.

**Method 5 - Ensemble modelling**

Considering combining the result to potentially build up a better model, an ensemble learning method is used through Stacking classifier. The stacking algorithm firstly trains the multiple models based on training set. Then test data is involved to make a prediction. The prediction result for each model is then combined based on input variables. The combined prediction result and test set’s ground truth forms a new data set. Under new data set, a meta classifier uses input from new data set to predict final results. For meta classifier, logistic regression classifier is used.



**Figure 3 -** Flow chart of ensemble modelling

Experiments

Method 1 - Logistic regression

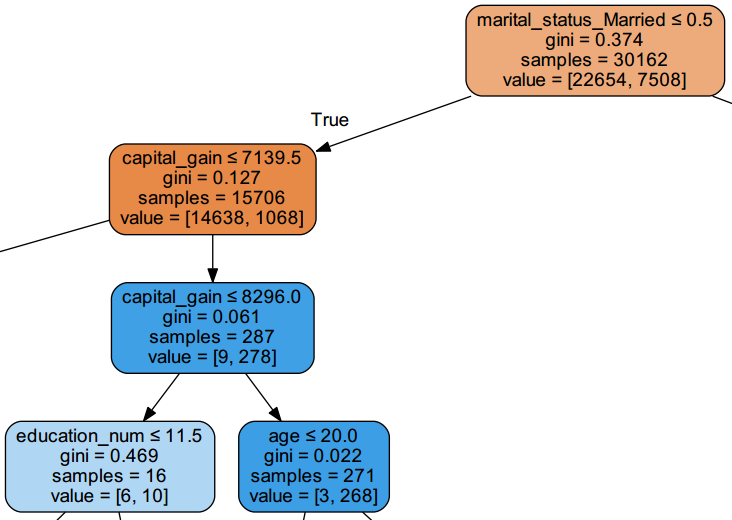
Logistic regression: as a baseline model, logistic takes a simple form without much parameter tuning. Using default settings with regularization power c = 0.01 (indicating a high penalization), logistic regression has an accuracy of 08423 and time is 0.249s, which represents an efficient and decent result.

Method 2 - Decision tree

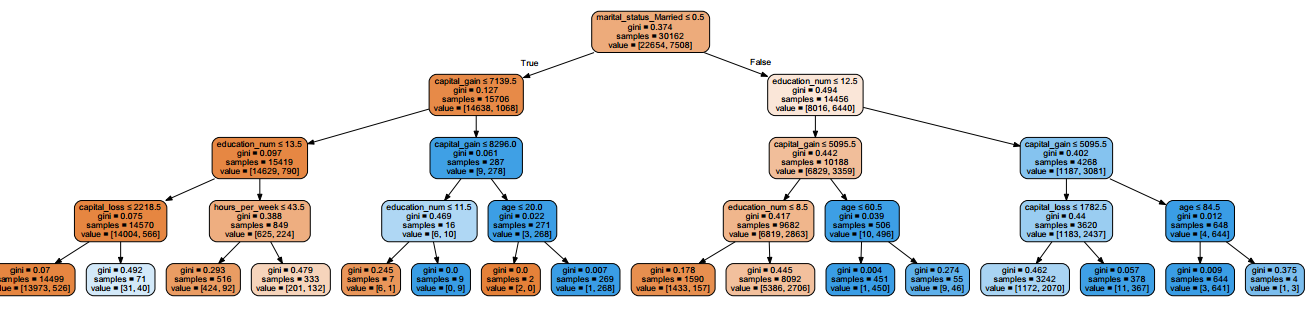
Firstly, for convenience of visualization, a tree with maximum depth of 4 is created.

Interpretation: from top of the tree, the value [22654, 7508] means 22654 less than 50k, and 7508 samples >50k; for any node that goes to the left, it means on this node it is predicted to be < 50k, to the right is > 50k.

From the top node, if the person is not married, i.e. < =0.5, then it is lead to the left leaf (i.e. < 50k). by checking the actual distribution on left leaf, it has value of [14638, 1068], i.e. high proportion of <50k (with a relatively low gini = 0.127). This indicates a powerful split. The tree keeps on being split until maximum depth is reached. This presentation is in general aligned with common knowledge.

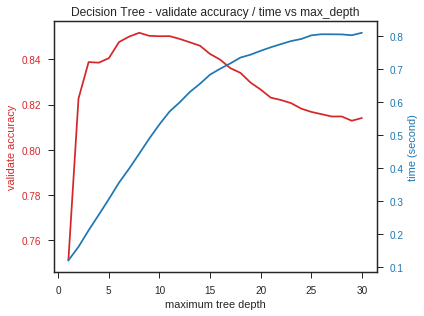


**Figure 4 –** illustration of part of decision tree



**Figure 5 –** illustration of decision tree

Although there are multiple decision tree algorithms: ID3, C5.0, CART, etc., sklearn uses an optimised version of the CART algorithm. For a single decision tree, major parameters that could be gridded include ‘max\_depth’ (maximum depth of the tree), and loss function. The ‘max\_depth’ limits the maximum depth of the tree, which prevents overfitting. Considering that our data has 13 features, with certain level of interdependence on input variables, we decide on grid search on number of features (1-30).



**Figure 6 –** decision tree accuracy vs max tree depth

The result shows that while gini/cross\_entropy makes little difference, max\_depth is important parameter. Along with increase in maximum tree depth, the time increases in a log like manner, and the accuracy goes up, reaches peak and then goes down, which represents overfitting. When max\_depth is 8, the accuracy reaches top at 85.16% while running time is moderate at 0.44s.

Method 3 - Random forest

Current Random forest in Scikit-learn is based on work done by Willianm Koehrsen[[19]](#endnote-19). The most optimal hyper parameters found by using **RandomGridSearch** is {'n\_estimators':50, 'criterion': 'entropy', 'max\_features': 15,}

With these parameters the 10-fold cross validation we enabled the multi-processing which reduced the run time by about 10%.

|  |  |  |
| --- | --- | --- |
| 10-fold cross validation | Accuracy | Time seconds |
| Enable Multi processing | 86.13% | 120.66 |
| Disable Multi processing | 86.15% | 137.47 |

**Table 1** - 10-fold cross validation result

We specifically looked at these parameters one by one:

**n\_estimators** is the number of subtrees in the forest. Keeping rest parameters unchanged we tested different number of subtrees ‘running time and accuracy

The number of subtrees increase will increase the runtime, while the accuracy did not increase much.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | |
| Tree\_size | | Accuracy | Time(s) |
| 5 | | 85.77% | 0.33 |
| 10 | | 85.71% | 0.63 |
| 20 | | 85.93% | 1.23 |
| 50 | | 86.06% | 4.16 |
| 100 | | 86.00% | 6.27 |
| 200 | | 86.00% | 12.41 |

**Table 2** - random forest – tree size vs accuracy/time

**Gini and Entropy**

Random forest consists of a number of decision trees. Each feature is used as a condition to split the dataset, the similar dependent variable will be classified as the same class. In Scikit-learn implementation, the node split criterion is Gini and entropy. The Gini measure the Gini impurity, the entropy is for information gain.

From test metrics result the difference between two split criteria is negligible This is also confirmed from Raileanu’s work[[20]](#endnote-20), there are about 2% difference in most cases. The Entropy method is slightly slower because it is taking the logarithm.

|  |  |  |
| --- | --- | --- |
| Criterion | Accuracy | Time(s) |
| gini | 85.93% | 3.12 |
| entropy | 85.94% | 3.30 |

**Table 3** - random forest – loss function vs accuracy/time

**Max\_features** is the number of features to consider when looking for best split,the max\_feature and are randomly selected from existing predictors and usually less than number of features m[[21]](#endnote-21).

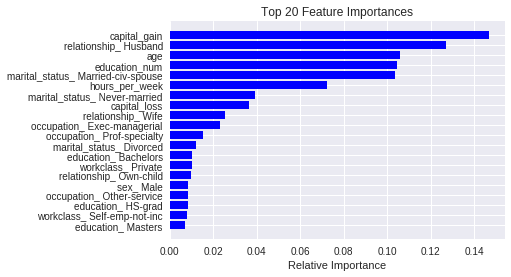
The suggested values are and , which are 5, 10, 20 in our case, these valued being used in grid search to find optimal max\_features number.

|  |  |  |
| --- | --- | --- |
| Max\_feature | Accuracy | Time(s) |
| 1 | 81.63% | 0.66 |
| 5 | 85.40% | 1.43 |
| 10 | 85.94% | 2.33 |
| 20 | 85.88% | 3.84 |

**Table 4** - random forest – max features vs accuracy/time

The result indicates the run time will increase when use more features, and the Max\_features number 10 will give good balance between the accuracy and runtime.

One advantage of using the tree-based algorithm is the ability to provide the most important features. The below figures show the top 20 most important features.



**Figure 7 –** random forest top 20 feature importances

Conclusion: After grid search hyper parameters tuning random forest is the chosen model used, random forest’s run time is relative fast,10-fold cross validation on training data set time is 35.47 seconds with accuracy 86.12%. The prediction accuracy on test data is 86.06% with 1.57 seconds. It is also providing good indication of feature importance. Even though for this data we have cleaned the data, random forest requires least data preparation and can deal with the missing data very well.

Method 4 - XGBoost

*Design:* Google’s Colaboratory—which is based on the open source project Jupyter—was used for the ease of prototyping as the XGBoost package was already installed and available. No further pre-processing techniques on the dataset were required for XGBoost. In order to choose an appropriate model and complexity, the number of trees, tree-specific parameters, regularization parameters and learning rate of XGBoost will be tuned across 10-fold cross validation using its built-in cross validation on the training set with area under the Receiver Operating Characteristic (ROC) curve (AUC) as the criterion. The description of parameters is shown in Table 5.

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter Type** | **Parameter** | **Default Value** | **Description** |
| Tree-specific | *max\_depth* | 6 | maximum depth of a tree |
| *min\_child\_weight* | 1 | minimum sum of instance weight (hessian) needed in a child. If the tree partition step results in a leaf node with the sum of instance weight less than *min\_child\_weight*, then the building process will give up further partitioning |
| *gamma* | 0 | minimum loss reduction required to make a further partition on a leaf node of the tree. The larger, the more conservative the algorithm. |
| *subsample* | 1 | subsample ratio of the training instance. 0.5 means that XGBoost randomly collected half of the data instances to grow trees and this will prevent overfitting. |
| *colsample\_bytree* | 1 | subsample ratio of columns for each split, in each level |
| Regularization | *alpha* | 0 | L1 regularization term on weights |

**Table 5 -** Description of the parameters in XGBoost, as described in the Python package

AUC is used because Bradley (1996) reported there to be a good agreement between accuracy and AUC, with AUC being one of the best ways to evaluate a classifier's performance with single evaluation[[22]](#endnote-22). The built-in cross validation is advantageous as it is implemented at each iteration of the boosting process, making it easy to obtain the exact optimum number of boosting iterations in a single run.

No further parallel processing was required as XGBoost already implements parallel processing. This is achieved by swapping the order of the two loops in sequential building process of a decision tree. Respectively, these are the outer loop to enumerate the leaf nodes, and the inner loop for enumerating features. Swapping their order results in parallelizing the split finding for different features on the same level. The changed order of the loop also allows sorting instances in each node to be avoided. Therefore, instances can be globally sorted first, and this sorted result can be used at each level[[23]](#endnote-23).

Initially, XGBoost was carried out using the default tree-specific and regularization parameters, with the default learning rate of 0.3 for comparison with the tuned parameters.

GridSearchCV was used to tune the parameters. The optimal number of trees (n\_estimators) based on 10-fold cross validation with AUC as the evaluation metric was 380. For tree-specific parameters, the most optimal parameters found were {‘max\_depth’: 4, ‘min\_child\_weight’: 0, ‘gamma’: 0.15, ‘subsample’: 0.9, ‘colsample\_bytree’: 0.55}. Given these parameters, the best learning rate according to grid search was 0.2. Optimizing the parameters resulted in an improvement in AUC and accuracy, but an increase in running time. The corresponding AUC and accuracy of XGBoost under default and optimized parameters on the training set are shown in Table 6. ROC curve of XGBoost using default and tuned parameters for the test set is depicted in Figure 8.

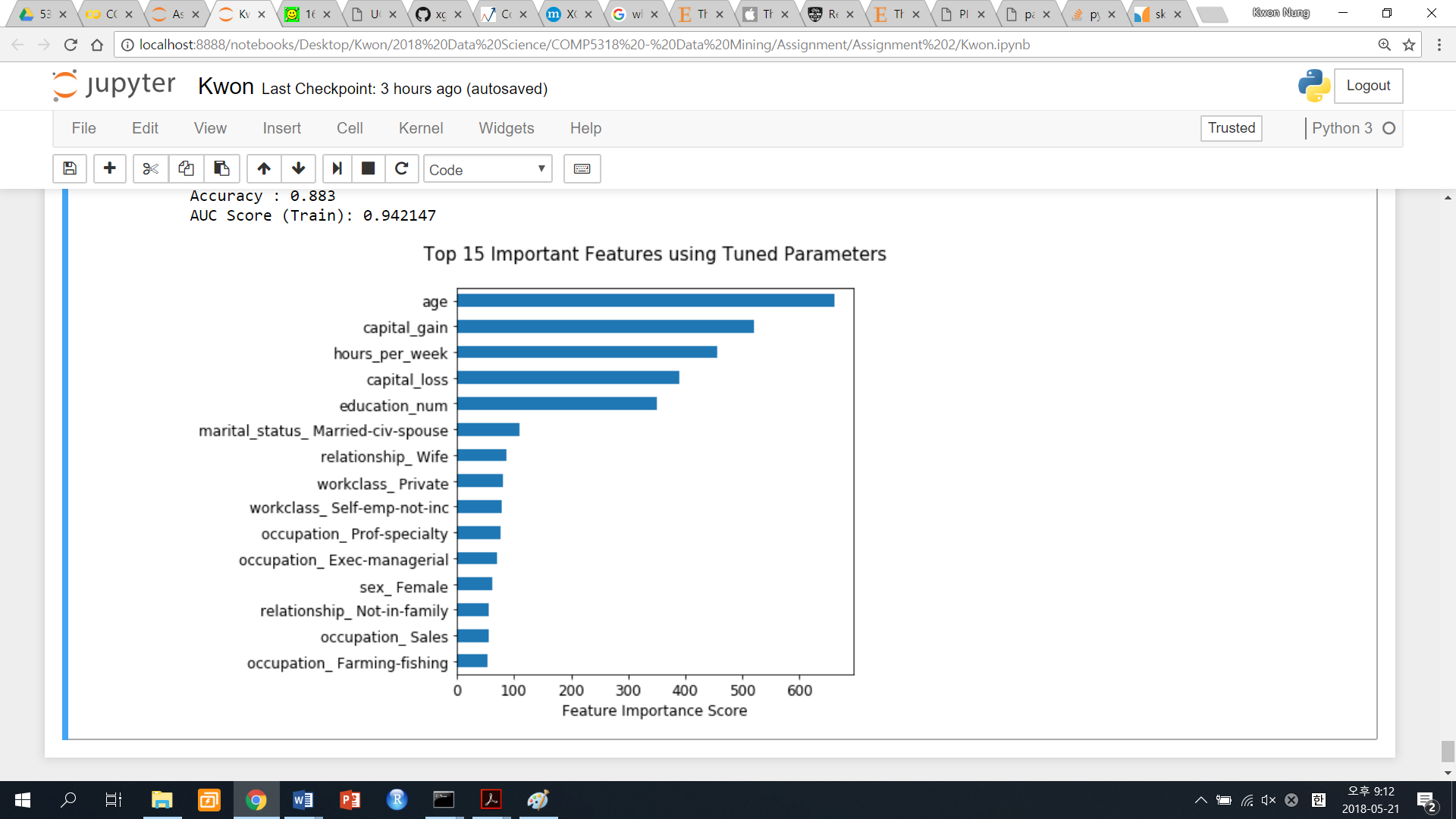
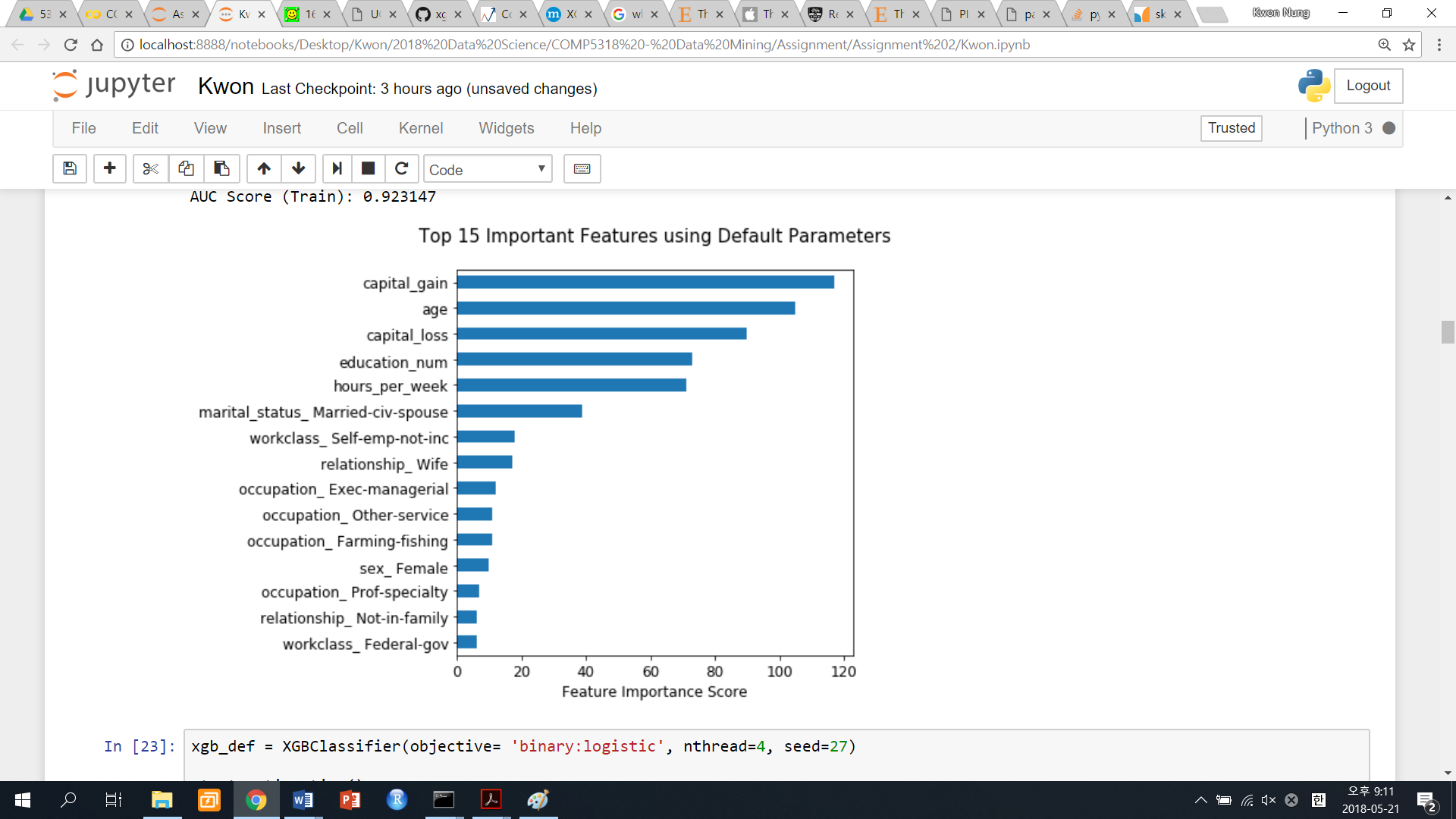
|  |  |  |  |
| --- | --- | --- | --- |
| **Parameters** | **AUC** | **Accuracy** | **Running Time** |
| Default | 0.923 | 0.863 | 49.778 |
| Tuned | 0.944 | 0.885 | 121.419 |

**Table 6 -** 10-fold cross validation AUC, accuracy and running time for XGBoost using default and tuned parameters on training set



**Figure 8 -** ROC curve of XGBoost using default and tuned parameters for test set

Furthermore, the top 15 important features selected by XGBoost was also different between default and tuned parameters. As illustrated in Figure 9, capital gain has the highest important feature score for default parameters in predicting whether a person’s income is above 50k. This was followed by age and capital loss. Conversely, age was the most important feature for tuned parameters, followed by capital gain and hours per week. Not only were the top features changed, but the feature importance score is increased overall. Under the default parameters, age had the feature importance score of approximately 100. This increased to over 600 under the tuned parameters. Similarly, the feature importance score of approximately 120 for capital gain under default parameters was increased to roughly 500 under the tuned parameters. This suggests that there are more number of occurrence of features in the ensemble for the tuned parameters.



**Figure 9 -** Comparison of top 15 important features from default and tuned hyperparameters

Final result summary– including ensemble modelling

The final performance comparison based on test set shows that the stacked model performs better than random forest, though underperformed than XGB. Also, the running time for stacked model is relatively high as it runs on multiple models. The top performance goes to XGB tuned and followed by XGB default. For random forest and logistic regression, they performed worse though with relative short running time. Its concluded that XGB tuned/default is ideal model for this data set with moderate running time.

|  | **model** | **accuracy** | **precision** | **recall** | **F\_score** | **running\_time** |
| --- | --- | --- | --- | --- | --- | --- |
|  | **XGB tuned** | **0.8693** | 0.8337 | 0.8003 | 0.8147 | 9.3151 |
|  | XGB default | 0.8643 | 0.8372 | 0.7766 | 0.7999 | 4.6824 |
|  | ensembled model | 0.8612 | 0.8284 | 0.7776 | 0.7978 | 19.3026 |
|  | Random Forest | 0.8585 | 0.8203 | 0.7803 | 0.7969 | 2.4989 |
|  | logistic\_regression | 0.8450 | 0.8058 | 0.7503 | 0.7713 | 0.2492 |

**Table 7 –** final result summary

Conclusions and future work

From the result, it is shown that XGBoost performs good in accuracy with decent running time. Random forest and logistic regression has relatively shorter running time but not as good accuracy.

The highest test accuracy achieved by XGB tuned model is 86.93%, which is higher than most algorithms done on the adult data set. This suggests robust prediction power of XGBoost.

For future works, the algorithm proposed could be potentially extended to other classification problems and could test on areas e.g. in income check for loan assessment, insurance underwriting, social benefits, pensions assessment, etc. Also, for XGBoost, better ways of tuning parameters speeding up could be researched for better implementation.

References

1. Zadrozny B. Learning and evaluating classifiers under sample selection bias. InProceedings of the twenty-first international conference on Machine learning 2004 Jul 4 (p. 114). ACM. [↑](#endnote-ref-1)
2. Keerthi SS, Lin CJ. Asymptotic behaviors of support vector machines with Gaussian kernel. Neural computation. 2003 Jul 1;15(7):1667-89. [↑](#endnote-ref-2)
3. Cerquides J, de Màntaras RL. Maximum a posteriori tree augmented naive bayes classifiers. InInternational Conference on Discovery Science 2004 Oct 2 (pp. 73-88). Springer, Berlin, Heidelberg. [↑](#endnote-ref-3)
4. Kurt I, Ture M, Kurum AT. Comparing performances of logistic regression, classification and regression tree, and neural networks for predicting coronary artery disease. Expert systems with applications. 2008 Jan 1;34(1):366-74. [↑](#endnote-ref-4)
5. Pal M, Mather PM. An assessment of the effectiveness of decision tree methods for land cover classification. Remote sensing of environment. 2003 Aug 30;86(4):554-65. [↑](#endnote-ref-5)
6. Mingers J. An empirical comparison of pruning methods for decision tree induction. Machine learning. 1989 Nov 1;4(2):227-43. [↑](#endnote-ref-6)
7. Brieman, L. and Cutler, A. (2007). Random forests - classification description. [online] Stat.berkeley.edu. Available at: https://www.stat.berkeley.edu/~breiman/RandomForests/cc\_home.htm [Accessed 20 May 2018]. [↑](#endnote-ref-7)
8. Hastie, Trevor; Tibshirani, Robert; Friedman, Jerome (2008). The Elements of Statistical Learning (2nd ed.). Springer. ISBN 0-387-95284-5. [↑](#endnote-ref-8)
9. Fette I, Sadeh N, Tomasic A. Learning to detect phishing emails. InProceedings of the 16th international conference on World Wide Web 2007 May 8 (pp. 649-656). ACM. [↑](#endnote-ref-9)
10. Kesler SR, Rao A, Blayney DW, Oakley-Girvan IA, Karuturi M, Palesh O. Predicting Long-Term Cognitive Outcome Following Breast Cancer with Pre-Treatment Resting State fMRI and Random Forest Machine Learning. Frontiers in human neuroscience. 2017 Nov 15;11:555. [↑](#endnote-ref-10)
11. van der Zande JJ, Dauwan M, Van Dellen E, Scheltens P, Lemstra AW, Stam CJ. Applying random forest machine learning to diagnose Alzheimer’s disease and dementia with Lewy bodies: A combination of electroencephalography (EEG), clinical parameters and biomarkers. Alzheimer's & Dementia: The Journal of the Alzheimer's Association. 2016 Jul 1;12(7):P661-2. [↑](#endnote-ref-11)
12. Chen T, Guestrin C. Xgboost: A scalable tree boosting system. In Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining 2016 Aug 13 (pp. 785-794). ACM. [↑](#endnote-ref-12)
13. Dong H, Xu X, Wang L, Pu F. Gaofen-3 PolSAR image classification via XGBoost and polarimetric spatial information. Sensors. 2018 Feb 17;18(2):611. [↑](#endnote-ref-13)
14. Torlay L, Perrone-Bertolotti M, Thomas E, Baciu M. Machine learning–XGBoost analysis of language networks to classify patients with epilepsy. Brain informatics. 2017 Sep 1;4(3):159-69. [↑](#endnote-ref-14)
15. Gaikwad DP, Thool RC. Intrusion detection system using bagging ensemble method of machine learning. InComputing Communication Control and Automation (ICCUBEA), 2015 International Conference on 2015 Feb 26 (pp. 291-295). IEEE. [↑](#endnote-ref-15)
16. Baumgartner D, Serpen G. Performance of global–local hybrid ensemble versus boosting and bagging ensembles. International Journal of Machine Learning and Cybernetics. 2013 Aug 1;4(4):301-17. [↑](#endnote-ref-16)
17. Georganos S, Grippa T, Vanhuysse S, Lennert M, Shimoni M, Wolff E. Very High Resolution Object-Based Land Use-Land Cover Urban Classification Using Extreme Gradient Boosting. IEEE Geoscience and Remote Sensing Letters. 2018 Feb 28. [↑](#endnote-ref-17)
18. Zhang D, Qian L, Mao B, Huang C, Huang B, Si Y. A Data-Driven Design for Fault Detection of Wind Turbines Using Random Forests and XGboost. IEEE Access. 2018;6:21020-31. [↑](#endnote-ref-18)
19. Random forest explanation, https://medium.com/@williamkoehrsen/random-forest-simple-explanation-377895a60d2d, 2017 [↑](#endnote-ref-19)
20. Raileanu LE, Stoffel K. Theoretical comparison between the gini index and information gain criteria. Annals of Mathematics and Artificial Intelligence. 2004 May 1;41(1):77-93. [↑](#endnote-ref-20)
21. Breiman L. Random forests. Machine learning. 2001 Oct 1;45(1):5-32. [↑](#endnote-ref-21)
22. Bradley AP. The use of the area under the ROC curve in the evaluation of machine learning algorithms. Pattern recognition. 1997 Jul 1;30(7):1145-59. [↑](#endnote-ref-22)
23. Fang, Z. (2018). Parallel Gradient Boosting Decision Trees. [online] Zhanpengfang.github.io. Available at: http://zhanpengfang.github.io/418home.html [Accessed 25 May 2018]. [↑](#endnote-ref-23)