Hyperparameter Optimization Using GridSearchCV to Improve Machine Learning Classification Models Performance

Mateen Alam **2019-ag-6050**

Rashid Latif **2019-ag-6010**

Department of Computer Science

University of Agriculture

Faisalabad, Pakistan

2023

Abstract

Hyperparameters are parameter values that control the learning process and have a significant effect on the performance of machine learning models. Most of the machine learning algorithms come with default values of these hyperparameters. But default values do not always perform well on different types of Machine Learning projects. Machine learning models often require careful tuning of hyperparameters to achieve optimal performance. Hyperparameter optimization (HPO) techniques aim to automate this process by searching the hyperparameter space for the best combination of values. This project report explores gridsearchcv optimization method for improving classification models in machine learning. We apply and evaluate popular optimization algorithm on different datasets and classifiers. The results demonstrate the effectiveness of HPO in enhancing classification model performance. In conclusion, hyperparameter optimization techniques play a vital role in improving the performance of machine learning classification models. Our project report provides a comprehensive analysis of different hyperparameters, highlighting their effect on outputs. The findings emphasize the importance of selecting the appropriate hyperparameter values based on the dataset, classification algorithm, and available computational resources. Future work can explore advanced optimization algorithms and hybrid approaches to further enhance the efficiency and effectiveness of hyperparameter optimization in machine learning.

Keywords: Hyperparameters Tuning, Classification Models, GridSearchCV. Hyperparameter Optimization, Optimization Techniques

Table of Contents

List of Figures

List of Tables

1 Introduction

1.1 Machine Learning

Machine Learning is the capability of artificial intelligence systems to learn by extracting patterns from data. It uses data to detect patterns in a datasets and adjust program actions accordingly. It focuses on the development of computer programs that can teach themselves to grow and change when exposed to new data. It enables the computers to find hidden insights using iterative algorithms without being explicitly programmed [1].

1.2 Application of Machine Learning:

Machine learning (ML) algorithms have been widely used in many applications domains, including advertising, recommendation systems, computer vision, natural language processing, and user behavior analytics. This is because they are generic and demonstrate high performance in data analytics problems [2]. Different ML algorithms are suitable for different types of problems or datasets. In general, building an effective machine learning model is a complex and time-consuming process that involves determining the appropriate algorithm and obtaining an optimal model architecture by tuning its hyper-parameters (HPs) [3].

1.3 Hyperparameters

Hyperparameters are parameter values that control the learning process of a Machine Learning model. These values have a direct effect on the learning rate and output of a model. So, it is important to select specific parametric values to get good results. Examples of common hyperparameters include learning rate, regularization strength, number of hidden layers in a neural network, kernel type and parameters in support vector machines, and the number of trees in a random forest [4].

Two types of parameters exist in machine learning models: one that can be initialized and updated through the data learning process (e.g., the weights of neurons in neural networks), named model parameters; while the other, named hyper-parameters, cannot be directly estimated from data learning and must be set before training a ML model because they the architecture of a ML model [5].

1.4 Hyperparameter Optimization

Hyperparameter optimization is the process of finding right combination of hyperparameters values to achieve maximum performance on the data in a reasonable time. Not all model hyperparameters are equally important. Some hyperparameters have an outsized effect on the behavior, and in turn, the performance of a machine learning algorithm [6].

Tuning hyper-parameters is considered a key component of building an effective ML model, especially for tree-based ML models and deep neural networks, which have many hyper parameters.Hyper-parameter tuning process is different among different ML algorithms due to their different types of hyperparameters, including categorical, discrete, and continuous hyper-parameters. Manual testing is a traditional way to tune hyper-parameters and is still prevalent in graduate student research, although it requires a deep understanding of the used ML algorithms and their hyper-parameter value settings.

However, manual tuning is ineffective for many problems due to certain factors, including a

large number of hyper-parameters, complex models, time consuming model evaluations, and non-linear hyper-parameter interactions [7].

1.4.1 Importance of Hyperparameter Optimization

The main aim of HPO is to automate hyper-parameter tuning process and make it possible for users to apply machine learning models to practical problems effectively. The optimal model architecture of a ML model is expected to be obtained after a HPO process.
Some important reasons for applying HPO techniques to ML models are as follows:

1. It reduces the human effort required, since many ML developers spend considerable time tuning the hyper-parameters, especially for large datasets or complex ML algorithms with a large number of hyper-parameters.
2. It improves the performance of ML models. Many ML hyper-parameters have different optimums to achieve best performance in different datasets or problems.
3. It makes the models and research more reproducible. Only when the same level of hyper-parameter tuning process is implemented can different ML algorithms be compared fairly; hence, using a same HPO method on different ML algorithms also helps to determine the most suitable ML model for a specific problem [8].

1.5 Problem Statement and Motivation

The problem addressed in this project is the suboptimal performance of machine learning models in classification tasks due to the improper selection of hyperparameters. Hyperparameters play a crucial role in determining the behavior and performance of these models, and choosing the right values is a challenging and time-consuming task. Suboptimal hyperparameter settings can lead to poor model performance, including under fitting or overfitting, ultimately resulting in inaccurate predictions and reduced generalization ability [9].

The motivation behind this project stems from the critical need to improve the classification performance of machine learning models. Classification tasks are fundamental in various domains, such as healthcare, finance, marketing, and image recognition. However, achieving accurate and reliable classification models requires careful consideration of hyperparameters [10].

Currently, many practitioners resort to manual or arbitrary selection of hyperparameters, which can be inefficient and result in suboptimal model performance. Moreover, the vast number of possible hyperparameter combinations makes manual tuning impractical and time-consuming. Therefore, there is a strong motivation to explore and implement hyperparameter optimization techniques to automate and improve the process of finding optimal hyperparameter configurations [11].

Hyperparameter optimization algorithms offer an automated and systematic approach to explore the hyperparameter space and identify the best settings for a given classification model. By leveraging these techniques, we can significantly enhance the model's performance, achieve better accuracy, and ensure robust generalization to unseen data [12].

The motivation behind this report is to investigate and compare various hyperparameter optimization techniques, evaluate their impact on classification model performance, and provide

practical recommendations for hyperparameter tuning. By addressing the problem of suboptimal hyperparameters and providing insights into effective optimization strategies, this project aims to assist practitioners in building more accurate and reliable classification models for their specific application domains [13].

1.6 Scope & Limitations

The scope of this project encompasses the exploration and implementation of hyperparameter optimization techniques to improve the classification performance of machine learning models. The project will focus on a selected set of classification algorithms, such as logistic regression, support vector machines, random forests, or neural networks, applied to a chosen dataset. Various hyperparameter optimization techniques will be investigated, including grid search, random search, Bayesian optimization, and genetic algorithms [14].

The project will involve preprocessing steps, such as data cleaning, normalization, feature engineering, or encoding categorical variables, to ensure the data is suitable for model training. The models will be trained using both default hyperparameters as a baseline and optimized hyperparameters using the selected optimization techniques. Performance evaluation will be conducted using appropriate metrics, such as accuracy and F1 score [15].

The project will provide an in-depth analysis of the results, comparing the performance of the optimized models against the baseline models. It will also offer recommendations and insights for hyperparameter tuning in classification tasks based on the findings and observations [16].

While this project aims to explore and implement hyperparameter optimization techniques for classification models, there are certain limitations that should be acknowledged:

Dataset Selection:
The project's findings and recommendations will be specific to the selected dataset used for experimentation. The dataset's characteristics, size, and complexity may influence the effectiveness of hyperparameter optimization techniques, and the conclusions drawn may not be universally applicable to all datasets.

Model Selection:
The project will focus on a set of chosen classification models, but there are numerous other models available in the field of machine learning. The findings may not be generalizable to all types of classification models.

Hyperparameter Space:
The exploration of hyperparameters is subject to the constraints of computational resources and time. Due to the vast number of possible hyperparameter combinations, exhaustive search may not be feasible, and certain regions of the hyperparameter space may remain unexplored.

Evaluation Metrics:
The performance evaluation of classification models relies on the selection of appropriate metrics. While commonly used metrics will be employed in this project, there might be other d omain-specific metrics that are relevant to certain applications but not explicitly covered.

Optimization Techniques:
A specific hyperparameter optimization technique will be explored, the project may not cover all existing methods.

2 Background

Hyperparameter optimization is very important approach to improve the learning rate of a model. This process is time consuming if it is performed manually. So, it is performed through different techniques.

2.1 Hyperparameter Optimization Techniques

This section provides an overview of existing hyperparameter optimization methods. It discusses traditional techniques, such as manual tuning and grid search, as well as more advanced methods that leverage optimization algorithms [17].

2.1.1 Grid Search

It involves exhaustively searching the hyperparameter space by evaluating the model's performance for all possible combinations of hyperparameter values. While grid search ensures a thorough exploration, it can be computationally expensive and inefficient [18].

2.1.2 Random Search

Random search is another popular technique that randomly samples hyperparameter configurations from the search space. This technique randomly samples hyperparameter values within predefined ranges, allowing for a more diverse exploration. Unlike grid search, it does not cover the entire space but focuses on a subset of randomly selected configurations. Random search has been shown to be more efficient than grid search, especially when the search space is large [19].

2.1.3 Bayesian optimization

It builds a probability model of the objective function and use it to select the most promising hyperparameters to evaluate in the true objective function. I t keeps a track of past evaluation results which they use to form a probabilistic model mapping hyperparameters to a probability of a score on the objective function:

P(score/hyperparameter) [20].

2.1.4 Evolutionary algorithms

Evolutionary algorithms allow to find set of optimal hyperparameters in an autonomous manner. These algorithms are of two types:

- **Particle Swarm Optimization:** It represents a computational method for optimizing continuous non-linear functions. The method is effective for optimizing a wide range of functions.
- **Genetic Algorithms:** It is motivated by the concept of natural selection. The GA maintains a population of possible solutions to the optimizing problems, which evolve through multiple generations in order to produce best solutions [21].

2.2 Evaluation Metrics:

To assess the performance of classification models, various evaluation metrics are commonly used:

- **Accuracy:** The proportion of correctly classified instances.
- **Precision:** The ability of the model to correctly identify positive instances out of the predicted positive instances.
- **Recall:** The ability of the model to correctly identify positive instances out of all actual positive instances.
- **F1 score:** The harmonic mean of precision and recall, providing a balanced measure of the model's performance.
- **Area Under the ROC Curve (AUC-ROC):** A metric that measures the model's ability to discriminate between positive and negative instances across different probability thresholds [22].

2.3 Model-Specific Hyperparameters:

Different machine learning models have their own set of hyperparameters. For example:

- Neural Networks: Hyperparameters include the number of hidden layers, the number of neurons per layer, the learning rate, activation functions, and regularization techniques such as dropout or L2 regularization.
- Support Vector Machines: Hyperparameters include the choice of the kernel function, the kernel parameters, the regularization parameter (C), and the tolerance for convergence.
- Random Forests: Hyperparameters include the number of decision trees, the depth of each tree, the number of features to consider at each split, and the maximum number of leaf nodes [23].

2.4 Cross Validation:

Cross-validation is a widely used technique to estimate the performance of models and select optimal hyperparameters. It involves partitioning the dataset into multiple subsets (folds), training the model on a subset, and evaluating it on the remaining fold. This process is repeated several times, and the average performance across folds is computed. Common cross-validation techniques include k-fold cross-validation and stratified cross-validation [24].

3 Materials and Methods

This section describes experiment setup and procedures employed to achieve research objectives. It provides a detailed description of datasets, classification algorithms, hyperparameters to be tuned and an overview of hyperparameter optimization techniques used in this projects.

3.1 Datasets Desccription:.

In this project, we used three different datasets and applied classification algorithms to optimize models hyperparametrs. Results of an experiment are affected by nature of datasets.

3.1.1 Breast_cancer dataset

Breast cancer is the most common cancer amongst women in the world. It accounts for 25% of all cancer cases, and affected over 2.1 Million people in 2015 alone. It starts when cells in the breast begin to grow out of control. These cells usually form tumors that can be seen via X-ray or felt as lumps in the breast area.
Breast_Cancer dataset describes cancer cells properties like mean radius, mean texture, mean area etc. This dataset consists of 569 rows and 32 columns. Target column consist of 0 and 1 value, as 0 describe malignant tumor and 1 describe about benign tumor. The complete dataset is available at Breast Cancer Dataset | Kaggle.

3.1.2 Heart_attack dataset:

This dataset defines different features of person and level of different materials in their body. These materials are related with heart conditions. This dataset consists of 303 rows and 14 columns. Target column contains two values 0 and 1. Number 0, for less chances of heart attack and 1 for more chances of heart attack. The full dataset is available at Heart Attack Analysis & Prediction Dataset | Kaggle.

3.1.3 Seed dataset:

Seed dataset contains data about three wheat seeds and finally describe about the type of seed. This dataset consists of 210 rows and 8 columns. Target column contains three values 0,1 and 2. The value 0, for Type 1, 1 for Type 2 and 2 for Type 3. Different classification models could be used to classify specific seed from three different types of seed on the basis of properties. This dataset is available at seeds dataset | Kaggle.

3.2 Classification Algorithms

Classification algorithms predicts categorical class labels or classifies information into discrete values. In classification, the goal is to build models that can assign input data instances to predefined categories or classes based on their features. These models learn patterns and relationships from labeled training data and use them to make predictions on unseen data. However, the performance of classification models heavily relies on their configuration, specifically the choice of hyperparameters [25].

3.2.1 Support Vector Machines(SVM)

Support vector machine is on the basis of statistical learning theory. The support vector machine is a novel small-sample learning method, because it is based on the principle of structural risk minimization. It is superior to existing methods on many performances. Support vector machine is a two dimensional description of the optimal surface evolved from the linearly separable case [26].

SVM Tuned Hyperparapmetes:

Table 1: List of SVM Hyperparameters values.

Param_C	Kernel	degree
1	Rbf	3
3	Linear	5
5	Poly	7

3.2.2 Random Forest Classifier

Random forests are a combination of tree predictors such that each tree depends on the values of a random vector sampled independently and with same distribution for all trees in the forest. The generalization error for forests converges to a limit as the number of trees in the forest becomes large. The generalization error of a forest of tree classifiers depends on the strength of the individual trees in the forest and the correlation between them. Using a random selection of features to split each node yields error rates that compare favorably to Adaboost, but are more robust with respect to noise. Internal estimates monitor error, strength, and correlation and these are used to show the response to increasing the number of features used in the splitting. Internal estimates are also used to measure variable importance [27].

RandomForest Tuned Hyperparameters:

Random Forest Classifier have some effective hyperparameters that control the learning process and manage output. There is a list of tuned hyperparameters:

Table 2: List of RFC Hyperparameters values

n_estimators	criterion	max_depth
130	gini	3
160	entropy	6
190	log_loss	9

3.2.3 K Nearest Neighbors(KNN)

KNN is a machine learning technique for classification and regression. It is based on feature similarity and finds the k the closest training examples in the dataset using the distance function. In KNN, for K number of the nearest neighbors, the distance between the query examples and all the training cases is computed using Euclidean distance for calculating the distance of unknown data from all the data points [28].

3.2.4 KNN Tuned Hyperparameters:

Table 3: List of KNN Hyperparameters values

leaf_size	n_neighbors	param_p
30	7	3
40	15	5
50	25	7

3.3 Optimization Technique (Grid Search CV)

It is a technique or method for finding optimal hyperprameter values from a given set of hyperparameteres in a grid. It is essentially a cross validation technique. We passed predefined values for hyperparameters to Grid Search CV function. This is done by using a dictionary in which we mentioned effective hyperparameters along with their values.

Grid Search CV tries all the combination of values passed in the dictionary and evaluates the model for each combination by using Cross-Validation method. Hence after using this function it gives accuracy/loss for every combination of hyperparameters [29].

3.3.1 Arguments in Grid Search CV:

- **estimator**: Passed the model instance that we desired to check hyperparameters
- **param-grid:** The dictionary object that holds the hyperameters
- **scoring:** Evaluation metrics
- **cv:** Value of cross-validation
- **verbose**: To get detailed result

3.4 Tableau

Tableau is a powerful and fastest growing data visualization tool used in the Business Intelligence Industry. It helps in simplifying raw data in a very easily understandable format. Tableau helps create the data that can be understood by professionals at any level in an organization. It also allows non-technical users to create customized dashboards.
Data analysis is very fast with Tableau tool and the visualizations created are in the form of dashboards and worksheets [30].

The best features of Tableau software are

- Data Blending
- Real time analysis
- Collaboration of data

3.5 Weka

Weka is a collection of machine learning algorithms for data mining tasks. It contains tools for data preparation, classification, regression, clustering, association rules mining, and visualization.

- Found only on the islands of New Zealand, the Weka is a flightless bird with an inquisitive nature. The name is pronounced like **this**, and the bird sounds like **this**.
- Weka is open source software issued under the **GNU General Public License [**31**]**.

3.5.1 J48 Classifier

It is an algorithm to generate a decision tree that is generated by C4.5 (an extension of ID3). It is also known as a statistical classifier. For decision tree classification, we need a database [32].

Steps include:

1. Open WEKA explorer.
2. Select weather csv file from the “choose file” under the preprocess tab option.
3. Go to the “Classify” tab for classifying the unclassified data. Click on the “Choose” button. From this, select “trees -> J48”

4. Click on Start Button. The classifier output will be seen on the Right-hand panel.

3.5.2 Random Trees:

The Random Trees node can be used with data in a distributed environment. In this node, you build an ensemble model that consists of multiple decision trees.

The Random Trees node is a tree-based classification and prediction method that is built on Classification and Regression Tree methodology. As with C&R Tree, this prediction method uses recursive partitioning to split the training records into segments with similar output field values. The node starts by examining the input fields available to it to find the best split, which is measured by the reduction in an impurity index that results from the split. The split defines two subgroups, each of which is then split into two more subgroups, and so on, until one of the stopping criteria is triggered. All splits are binary (only two subgroups) [33].

The Random Trees node uses bootstrap sampling with replacement to generate sample data. The sample data is used to grow a tree model. During tree growth, Random Trees will not sample the data again. Instead, it randomly selects part of the predictors and uses the best one to split a tree node. This process is repeated when splitting each tree node. This is the basic idea of growing a tree in random forest.

Random Trees uses C&R Tree-like trees. Since such trees are binary, each field for splitting results in two branches. For a categorical field with multiple categories, the categories are grouped into two groups based on the inner splitting criterion. Each tree grows to the largest extent possible (there is no pruning). In scoring, Random Trees combines individual tree scores by majority voting (for classification) or average (for regression) [34].

Random Trees differ from C&R Trees as follows:

- Random Trees nodes randomly select a specified number of predictors and uses the best one from the selection to split a node. In contrast, C&R Tree finds the best one from all predictors.

- Each tree in Random Trees grows fully until each leaf node typically contains a single record. So the tree depth could be very large. But standard C&R Tree uses different stopping rules for tree growth, which usually leads to a much shallower tree.

Random Trees adds two features compared to C&R Tree:

- The first feature is *bagging*, where replicas of the training dataset are created by sampling with replacement from the original dataset. This action creates bootstrap samples that are of equal size to the original dataset, after which a *component model* is built on each replica. Together these component models form an ensemble model.
- The second feature is that, at each split of the tree, only a sampling of the input fields is considered for the impurity measure [35].

4 Results and Discussions

Results are most important part of an experiment. This section will explain findings of our project in the form of tables and figures.

4.1 Support Vector Machin Results

Support vector machine classifier shows different result for different datasets. As we can see in result table, it is showing maximum accuracy for breast_cancer dataset 0.96%. SVM classifying seed dataset with 0.94% accuracy less as compared to breast_cancer dataset. The value of cross validation also has an effect on final result. Heart_attack dataset is classified with less accuracy due to its different nature.

Table 4: SVM parameters best combination with their values showing different output on different datasets.

Dataset	Cross_validation	param_C	degree	kernel	Accuracy
Breast_cancer	3	3	7	Linear	0.964789
Breast_cancer	5	3	7	Linear	0.969466
Breast_cancer	7	1	3	Linear	0.967135
Seed	3	5	7	Poly	0.942912
Seed	5	3	7	Poly	0.968347
Seed	7	5	7	Poly	0.948899
Heart_attack	3	1	3	Linear	0.766725
Heart_attack	5	5	7	Linear	0.801643
Heart_attack	7	1	3	Linear	0.823593

4.1.1 Accuracy results in Tableau

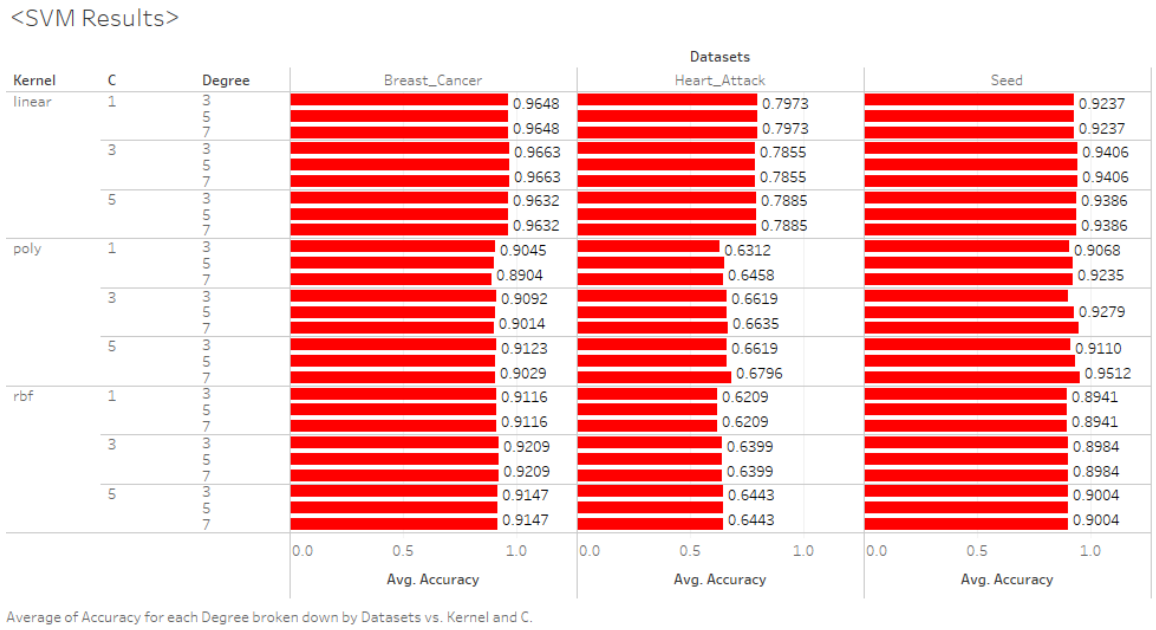


Figure 1: SVM accuracy results.

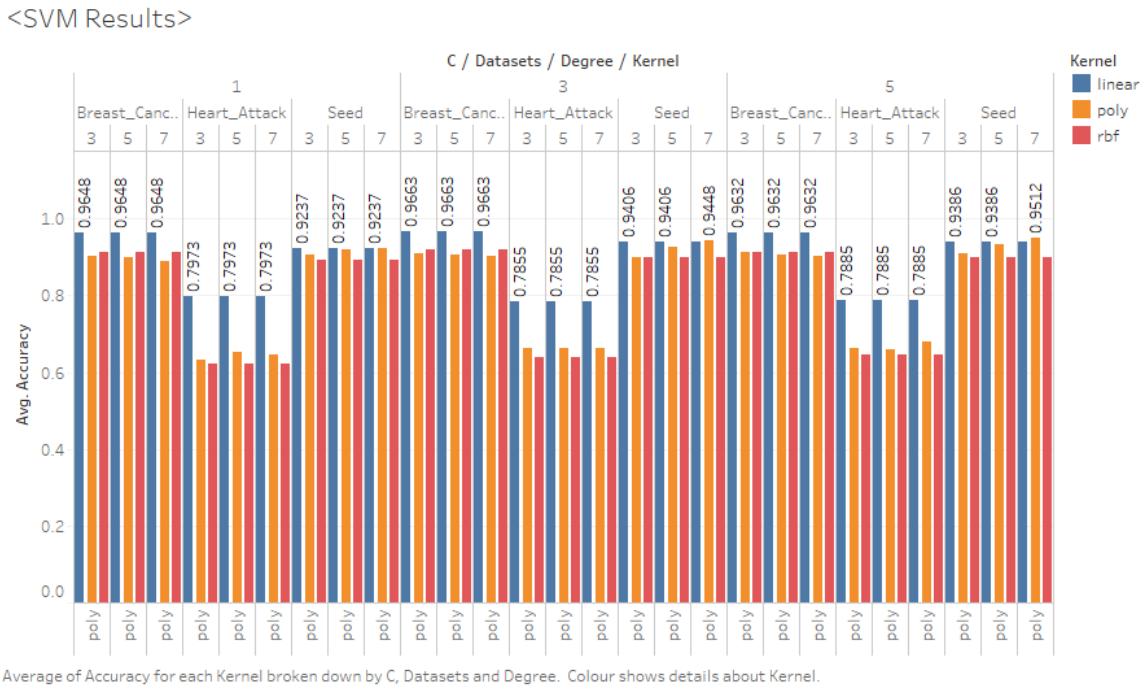


Figure 2: SVM results Bars.

4.1.2 SVM J48 Tree

As j48 is a statistical tree, it is showing three different levels of accuracy defined by finding three different quartiles for each accuracy results. Quartile 1 is used to specify low limit, Q2 for medium and Q3 for high accuracy.

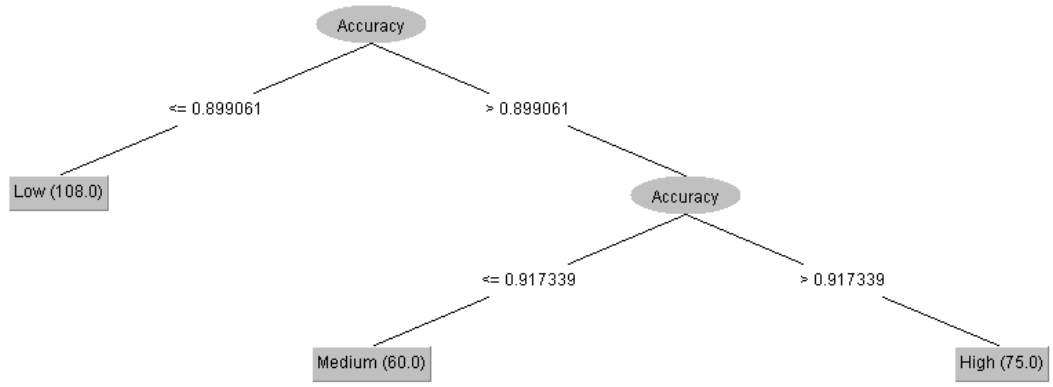


Figure 3: SVM J48 Tree

4.1.3 SVM Random Tree:

Random tree consists of ensemble distribution

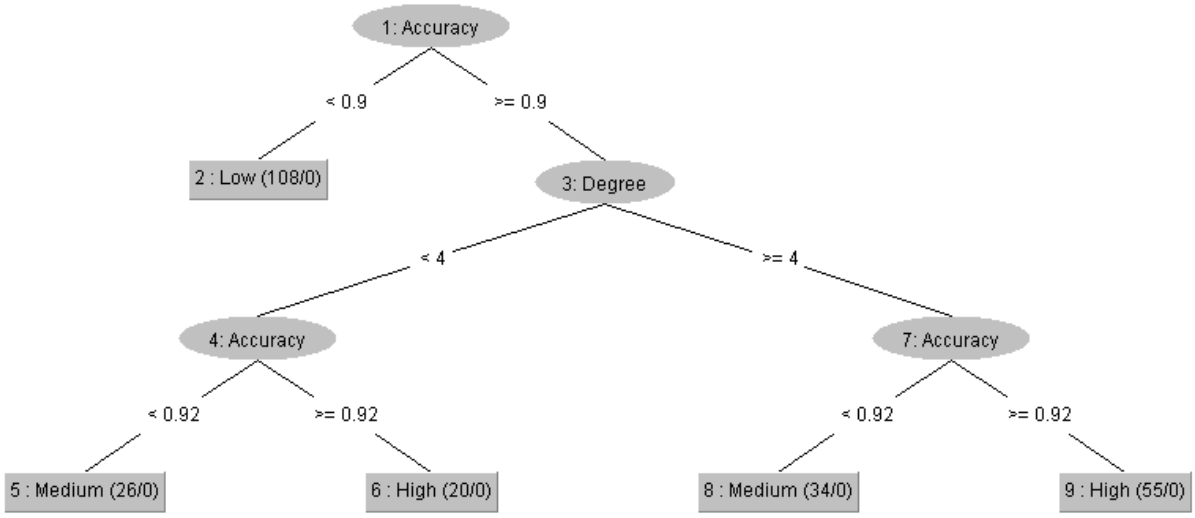


Figure 4: SVM Random Tree

4.2 Random Forest Results:

Random forest classifier produce maximum accuracy results as compared to other classification models. It shows maximum accuracy on breast cancer dataset 0.96. Accuracy in seed dataset is less as compared to breast cancer dataset. In heart attack dataset, it shows 0.84% accuracy.

Table 5: RFC parameters best combination with their values showing different output on different datasets.

Dataset	Cross_validation	criterion	max_depth	n_estimators	Accuracy
Breast_cancer	3	log_loss	6	130	0.966592
Breast_cancer	5	entropy	9	130	0.969439
Breast_cancer	7	entropy	9	160	0.96492
Seed	3	gini	6	130	0.87619
Seed	5	gini	9	190	0.923992
Seed	7	log_loss	9	190	0.9
Heart_attack	3	entropy	3	130	0.848185
Heart_attack	5	entropy	3	190	0.832367
Heart_attack	7	gini	9	190	0.848233

4.2.1 RFC results in Tableau:

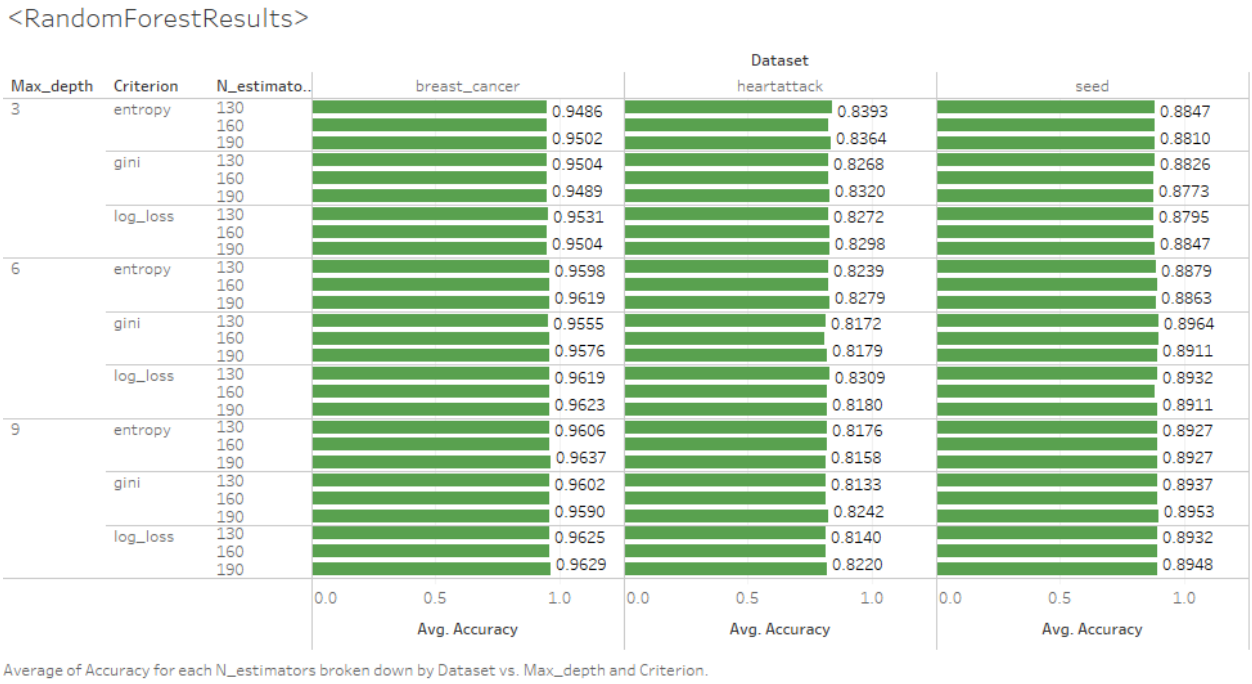


Figure 5: RFC results in Tableau.

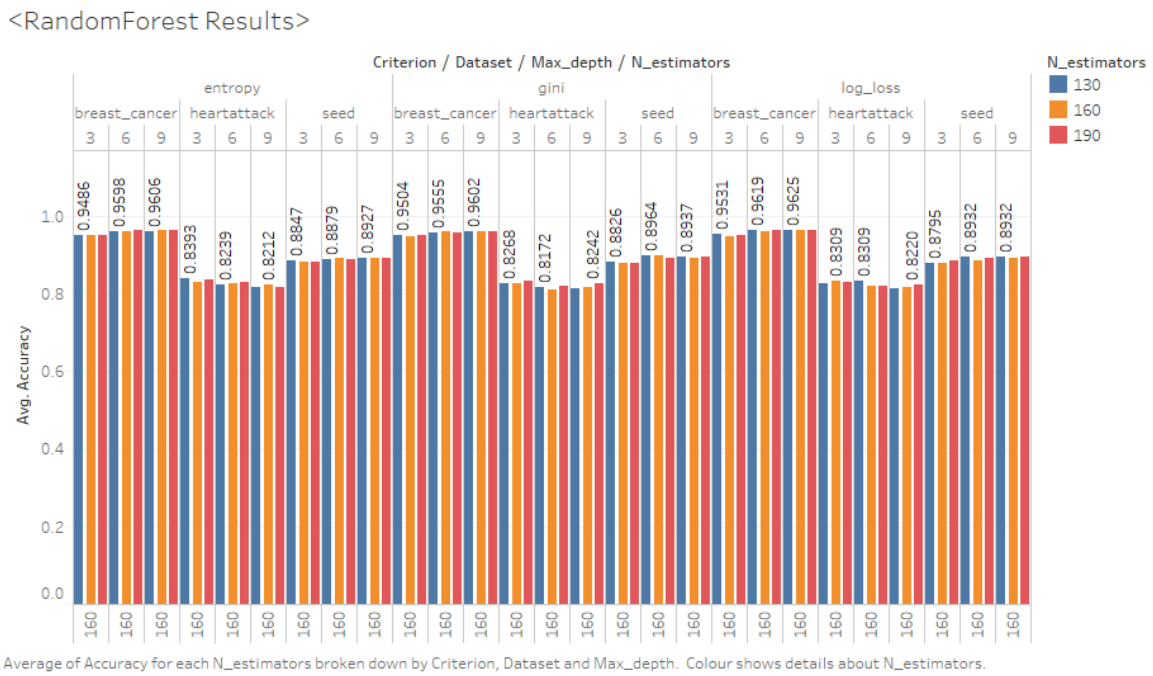


Figure 6: RFC Tableau Bars result.

4.2.2 RFC J48 Tree

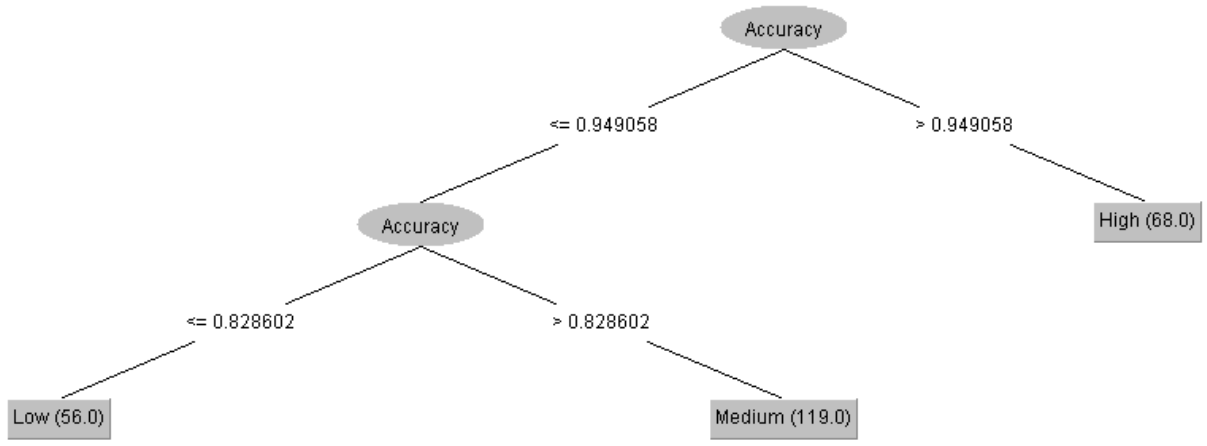


Figure 7: RFC j48 Tree

4.2.3 RFC Random Tree

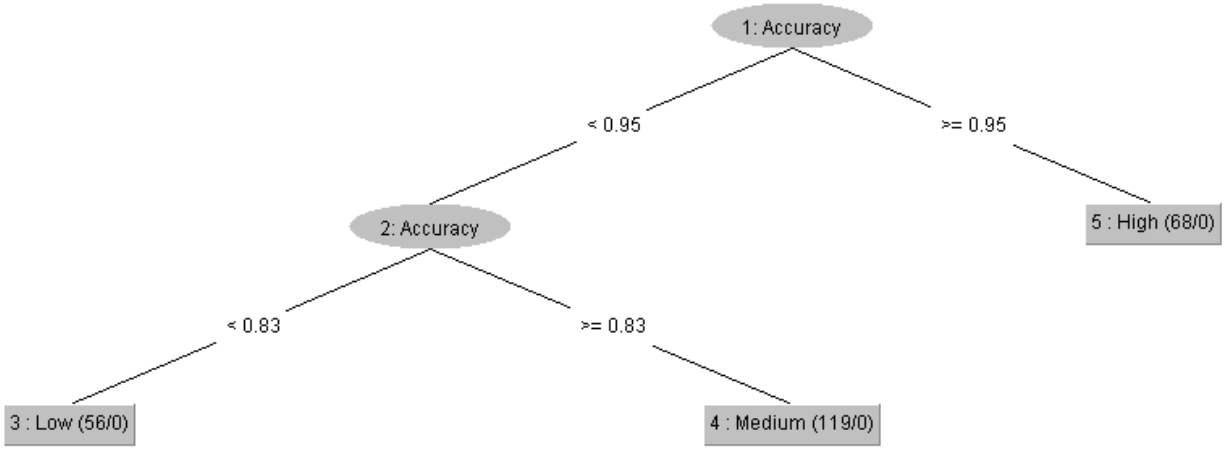


Figure 8: RFC random Tree

4.3 KNN Results:

K Nearest Neighbors classified data with 0.91% accuracy, which is less as compared to other classification models used in this project. It is showing lowest accuracy for heart_attack dataset. Overall, KNN is less accurate classifier then other ML claissification modles. KNN also shows variations in results with changing datasets.

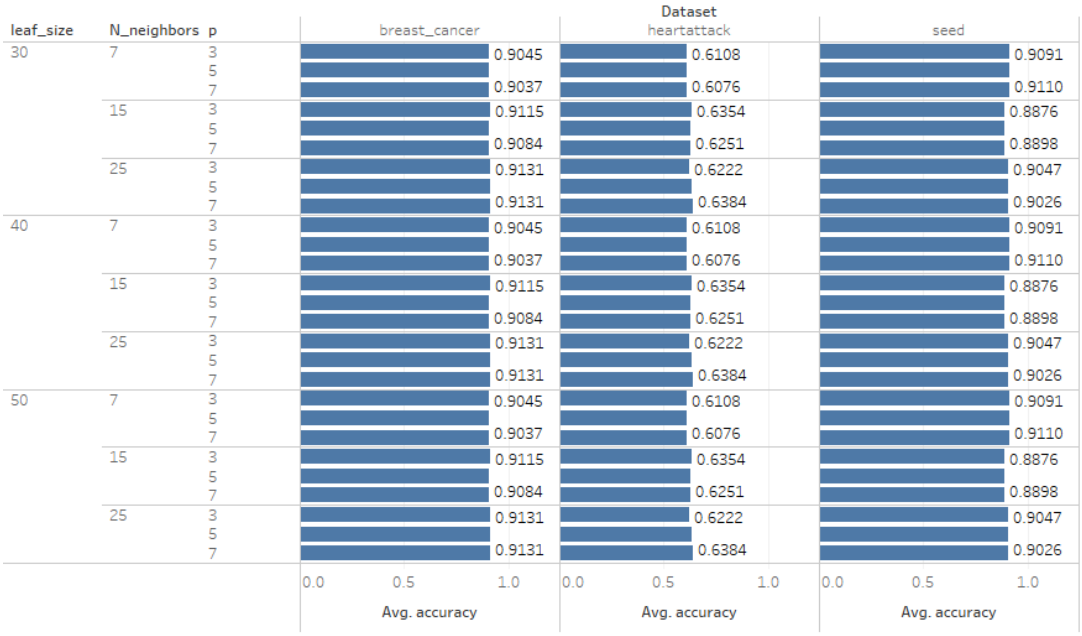
4.3.1 Results in GridSearchCV:

Table 6: KNN parameters best combination with their values showing different output on different datasets

Dataset	Cross_validation	leaf_size	n_neighbors	param_p	Accuracy
Breast_cancer	3	30	15	3	0.913146
Breast_cancer	5	50	25	3	0.915404
Breast_cancer	7	50	25	3	0.915496
Seed	3	30	7	3	0.904572
Seed	5	30	7	5	0.91754
Seed	7	30	7	3	0.91135
Heart_attack	3	50	25	7	0.616842
Heart_attack	5	40	25	5	0.651498
Heart_attack	7	50	15	3	0.655574

4.3.2 Tableau Results

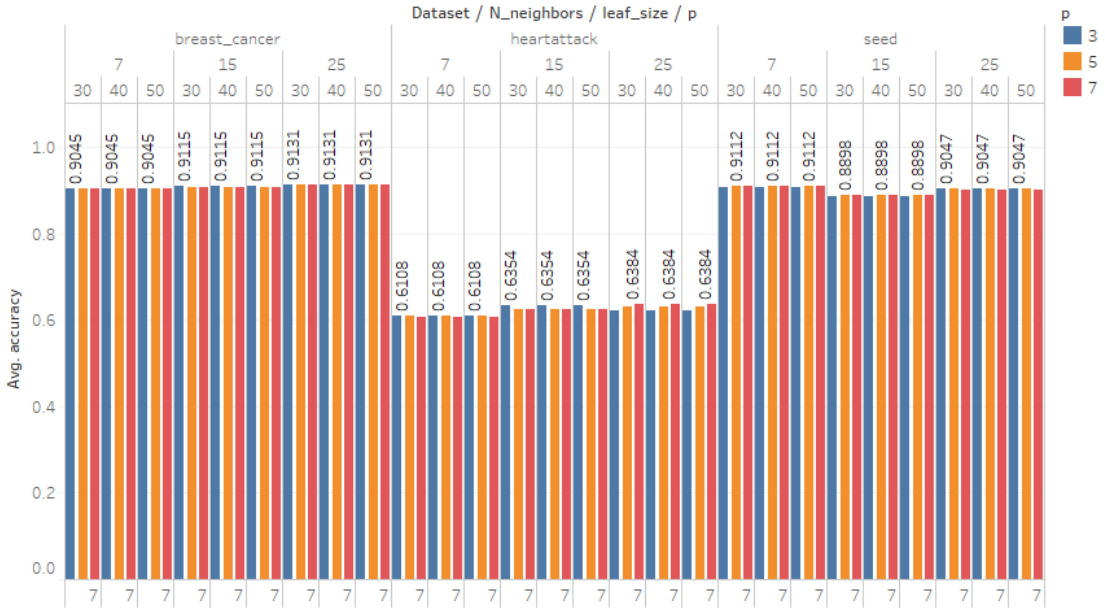
<KNearestNeighborsResults>



Average of accuracy for each p broken down by Dataset vs. leaf_size and N_neighbors.

Figure 9 KNN Results in Tableau

<KNNResults>



Average of accuracy for each p broken down by Dataset, N_neighbors and leaf_size. Colour shows details about p.

Figure 10. KNN Tableau Results bars

4.3.3 KNN J 48 Tree

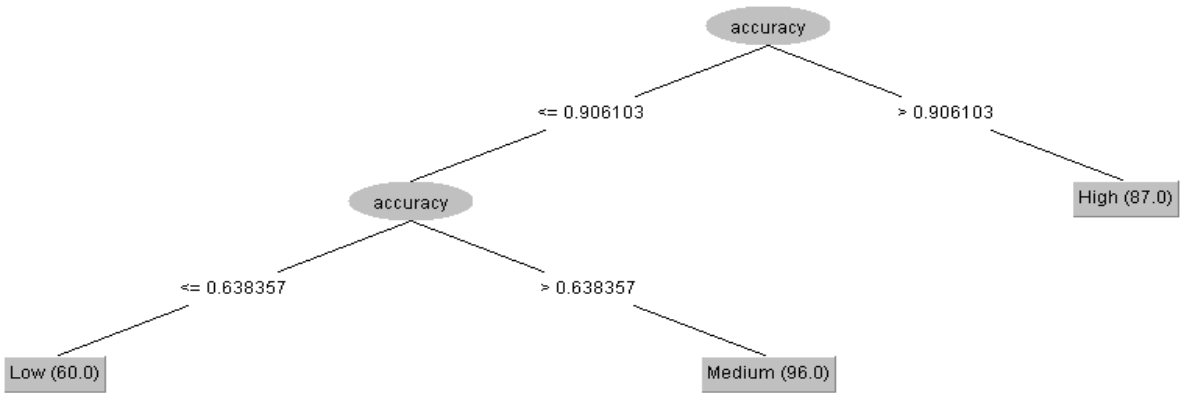


Figure 11 KNN J48 Tree

4.3.4 KNN Random Tree

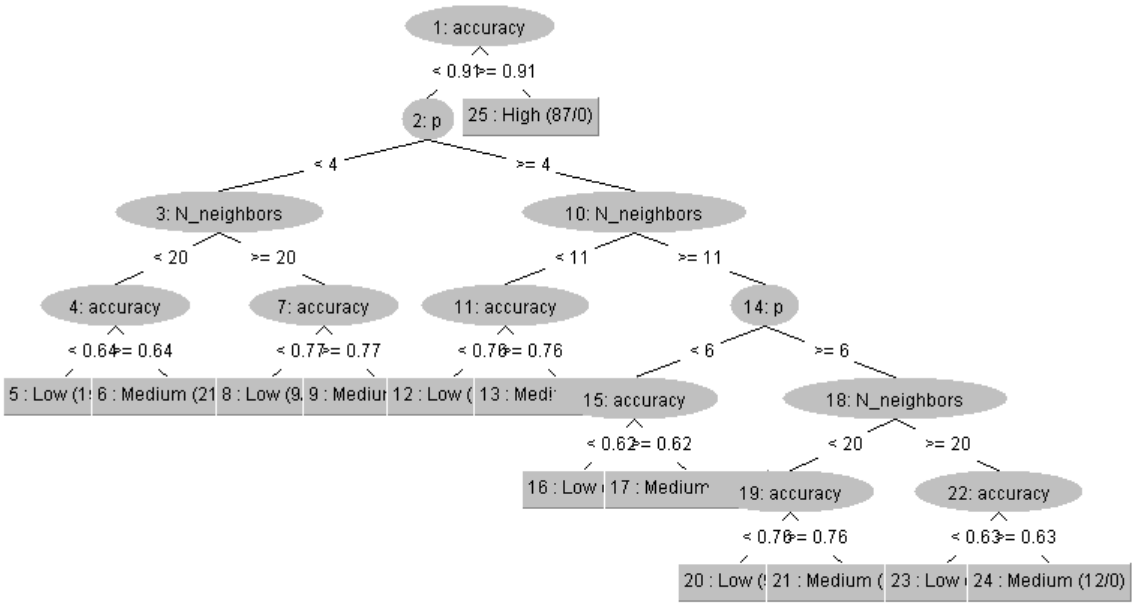


Figure 12 KNN Random Tree.

5 Conclusions

Hyperparameter optimization enhances the performance of Machine Learning classification models. Through optimization we can find best parameter values combination to perform classification with good speed and accuracy. GridSearchCV proved very effective technique for the process of optimization. Machine learning models performance greatly depends on the type and nature of dataset. Cross validation values are also effective to enhance performance of ML models. RandomForest Classifier shows highest accuracy for most of the datasets as compared to ML classification models. Many other Machine Learning models could be optimized by using different optimization technique.

6 References

[1] Jordan, M. I., Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. Science, 349(6245), 255-260. doi:10.1126/science.aaa8415

[2] Zöller, M. A., Huber, M. F. (2019). Benchmark and Survey of Automated Machine Learning Frameworks. arXiv preprint arXiv:1904.12054. Retrieved from https://arxiv.org/abs/1904.12054

[3] Shawi, R. E., Maher, M., Sakr, S. (2019). Automated machine learning: State-of-the-art and open challenges. arXiv preprint arXiv:1906.02287. Retrieved from http://arxiv.org/abs/1906.02287

[4] Kuhn, M., Johnson, K. (2013). Applied Predictive Modeling. Springer. ISBN: 9781461468493

[5] Diaz, G. I., Fokoue-Nkoutche, A., Nannicini, G., Samulowitz, H. (2017). An effective algorithm for hyperparameter optimization of neural networks. IBM Journal of Research and Development, 61(4), 1-20. doi:10.1147/JRD.2017.2709578

[6] Hutter, F., Kotthoff, L., Vanschoren, J. (Eds.). (2019). Automatic Machine Learning: Methods, Systems, Challenges. Springer. ISBN: 9783030053185

[7] Decastro-García, N., Muñoz Castañeda, A. L., Escudero García, D., Carriegos, M. V. (2019). Effect of the Sampling of a Dataset in the Hyperparameter Optimization Phase over the Efficiency of a Machine Learning Algorithm. Complexity, 2019. doi:10.1155/2019/6278908

[8] Abreu, S. (2019). Automated Architecture Design for Deep Neural Networks. arXiv preprint arXiv:1908.10714. Retrieved from http://arxiv.org/abs/1908.10714

[9] Steinholtz, O. S. (2018). A Comparative Study of Black-box Optimization Algorithms for Tuning of Hyper-parameters in Deep Neural Networks. M.S. thesis, Department of Electrical Engineering, Luleå University of Technology.

[10] Luo, G. (2016). A review of automatic selection methods for machine learning algorithms and hyper-parameter values. Network Modeling Analysis in Health Informatics and Bioinformatics, 5(1), 1-16. doi:10.1007/s13721-016-0125-6

[11] D. Maclaurin, D. Duvenaud, R.P. Adams, Gradient-based Hyper-parameter Optimization through Reversible Learning, arXiv preprint arXiv:1502.03492, (2015). http://arxiv.org/abs/1502.03492.

[12] J. Bergstra, R. Bardenet, Y. Bengio, and B. K´egl, Algorithms for hyper-parameter optimization, Proc. Adv. Neural Inf. Process. Syst., (2011) 2546–2554.

[13] B. James and B. Yoshua, Random Search for Hyper-Parameter Optimization, J. Mach. Learn. Res. 13 (1) (2012) 281–305.

[14] K. Eggensperger, M. Feurer, F. Hutter, J. Bergstra, J. Snoek, H. Hoos, K. Leyton-Brown, Towards an Empirical Foundation for Assessing Bayesian Optimization of Hyperparameters, BayesOpt Work. (2013) 1–5.

[15] K. Eggensperger, F. Hutter, H.H. Hoos, K. Leyton-Brown, Efficient benchmarking of hyperparameter optimizers via surrogates, Proc. Natl. Conf. Artif. Intell. 2 (2015) 1114–1120.

[16] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, Hyperband: A novel bandit-based approach to hyperparameter optimization, J. Mach. Learn. Res. 18 (2012) 1–52.

[17] Q. Yao et al., Taking Human out of Learning Applications: A Survey on Automated Machine Learning, arXiv preprint arXiv:1810.13306, (2018). http://arxiv.org/abs/1810.13306.

[18] S. Lessmann, R. Stahlbock, S.F. Crone, Optimizing hyperparameters of support vector machines by genetic algorithms, Proc. 2005 Int. Conf. Artif. Intell. ICAI'05. 1 (2005) 74–80.

[19] P. R. Lorenzo, J. Nalepa, M. Kawulok, L. S. Ramos, and J. R. Paster, Particle swarm optimization for hyper-parameter selection in deep neural networks, Proc. ACM Int. Conf. Genet. Evol. Comput., (2017) 481–488.

[20] S. Sun, Z. Cao, H. Zhu, J. Zhao, A Survey of Optimization Methods from a Machine Learning Perspective, arXiv preprint arXiv:1906.06821, (2019). https://arxiv.org/abs/1906.06821.

[21] T.M. S. Bradley, A. Hax, Applied Mathematical Programming, Addison-Wesley, Reading, Massachusetts. (1977).

[22] S. Bubeck, Convex optimization: Algorithms and complexity, Found. Trends Mach. Learn. 8 (2015) 231–357. https://doi.org/10.1561/2200000050.

[23] B. Shahriari, A. Bouchard-Côté, and N. de Freitas, "Unbounded Bayesian optimization via regularization," Proc. Artif. Intell. Statist., (2016) 1168–1176.

[24] G.I. Diaz, A. Fokoue-Nkoutche, G. Nannicini, H. Samulowitz, An effective algorithm for hyperparameter optimization of neural networks, IBM J. Res. Dev. 61 (2017) 1–20. https://doi.org/10.1147/JRD.2017.2709578.

[25] C. Gambella, B. Ghaddar, and J. Naoum-Sawaya, Optimization Models for Machine Learning: A Survey, arXiv preprint arXiv:1901.05331, (2019). http://arxiv.org/abs/1901.05331.

[26] E. R. Sparks, A. Talwalkar, D. Haas, M. J. Franklin, M. I. Jordan, and T. Kraska, Automating model search for large scale machine learning, Proc. 6th ACM Symp. Cloud Comput., (2015) 368–380.

[27] J. Nocedal and S. Wright, Numerical Optimization, (2006) Springer-Verlag, ISBN: 978-0-387-40065-5.

[28] A. Moubayed, M. Injadat, A. Shami, H. Lutfiyya, DNS Typo-Squatting Domain Detection: A Data Analytics & Machine Learning Based Approach, 2018 IEEE Glob. Commun. Conf. GLOBECOM 2018 - Proc. (2018). https://doi.org/10.1109/GLOCOM.2018.8647679.

[29] R. Caruana, A. Niculescu-Mizil, An empirical comparison of supervised learning algorithms, ACM Int. Conf. Proceeding Ser. 148 (2006) 161–168. https://doi.org/10.1145/1143844.1143865.

[30] O. Kramer, Scikit-Learn, in Machine Learning for Evolution Strategies. Cham, Switzerland: Springer International Publishing, (2016) 45–53.

[31] F. Pedregosa et al., Scikit-learn: Machine learning in Python, J. Mach. Learn. Res., 12 (2011) 2825–2830.

[32] T.Chen, C.Guestrin, XGBoost: a scalable tree boosting system, arXiv preprint arXiv:1603.02754, (2016). http://arxiv.org/abs/1603.02754.

[33] F. Chollet, Keras, 2015. https://github.com/fchollet/keras.

[34] C. Gambella, B. Ghaddar, J. Naoum-Sawaya, Optimization Models for Machine Learning: A Survey, (2019) 1–40. http://arxiv.org/abs/1901.05331.

[35] C.M. Bishop, Pattern Recognition and Machine Learning. (2006) Springer, ISBN: 978-0-387-31073-2.