AI TESTING: ENSURING A GOOD DATA SPLIT BETWEEN DATA SETS (TRAINING AND TEST) USING K-MEANS CLUSTERING AND DECISION TREE ANALYSIS

Kishore Sugali, Chris Sprunger and Venkata N Inukollu

Department of Computer Science and, Purdue University, Fort Wayne, USA

ABSTRACT

Artificial Intelligence and Machine Learning have been around for a long time. In recent years, there has been a surge in popularity for applications integrating AI and ML technology. As with traditional development, software testing is a critical component of a successful AI/ML application. The development methodology used in AI/ML contrasts significantly from traditional development. In light of these distinctions, various software testing challenges arise. The emphasis of this paper is on the challenge of effectively splitting the data into training and testing data sets. By applying a k-Means clustering strategy to the data set followed by a decision tree, we can significantly increase the likelihood of the training data set to represent the domain of the full dataset and thus avoid training a model that is likely to fail because it has only learned a subset of the full data domain.

KEYWORDS

Artificial Intelligence (AI), Machine Learning (ML), Software Testing

1. INTRODUCTION

1.1. Overview of Artificial Intelligence and Machine Learning.

Artificial Intelligence (AI), a rapidly emerging branch of Computer Science, emphasizes on the modelling and programming of human intelligence in machines, and, to enable them to think and function like rational intelligent systems. AI can be defined as the capability of a machine to imitate intelligent human behavior. Think about this - a machine than can easily execute simple to complex tasks on a daily basis without much of human intervention. AI has made several breakthroughs in the recent years and is gaining traction for using computers to decipher otherwise complex problems, and, thus surpassing the quality of current computer systems [5]. In [6], Derek Partridge demonstrates various major classes of association that exist between artificial intelligence (AI) and software engineering (SE). These areas of communication are software support environments; AI tools and techniques in standard software; and the use of standard software technology in AI systems. Mark Kandel and Bunke, H, in [7], have also tried to correlate AI and software engineering at certain levels and discussed whether AI can be directly applied to SE problems, and if SE Processes are equipped for taking advantage of AI techniques.

1.2. How AI Impacts Software Testing?

Research illustrates that software testing utilizes enterprise resources and adds no functionality to the application. If regression testing discloses a new error introduced by a revision code, a new

cycle of regression begins. Additionally, most software applications require engineers to write testing scripts, and their skills must be on par with the developers who initially code the app. This extra overhead expense in the quality assurance process is consistent with the growing complexity of software products.[6]

To minimize the costs, the automated testing focuses more on AI capacity, efficiency, and speed. New applications progressively provide AI functionality, sparing the challenge for human testers to comprehensively evaluate the entire product. Either data or market trends, AI will be increasingly needed to certify intelligence -containing systems, partly because the spectrum of input and output possibilities is so wide.
The intent of this paper is to focus on one of the issues and challenges that testers face in effectively testing an AI application. Currently there are various AI methods such as classification and clustering algorithms that rely primarily on monotonous data to train models to predict accurate results [8].

2. ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

2.1. Types of Machine Learning Algorithms

Machine Learning Algorithms have been divided into different categories based on their purpose. The following are the key categories: [3]

- **Supervised Learning** - ML tries to model relationships and dependencies between the target prediction output and the input features. Input data is called training data and has a known label or result. Algorithms include - Nearest Neighbor, Naive Bayes, Decision Trees, Linear Regression, Support Vector Machines (SVM), Neural Networks.

- **Unsupervised Learning** - Input data are not labeled and do not have a known result. Mainly used in pattern detection and descriptive modeling. Algorithms includes - k-means clustering and association rules.

- **Semi-supervised Learning** - In the previous two categories, either there are no labels for all the observations in the dataset or labels are present for all the observations. Semi-supervised learning falls in between the two. Input data is a mix of labeled and unlabeled.

- **Reinforcement Learning** - It allows machines and software agents to automatically determine the ideal behavior within a specific context, in order to maximize its performance. Algorithms includes Q-Learning, Temporal Difference (TD), and Deep Adversarial Networks.

3. CHALLENGE OF SPLITTING DATA INTO TRAINING AND TEST SETS

It can be a major challenge to effectively split a data set into a training data set and a testing data set. A good split leads to a productive training of the model while a poor split is more likely to lead to an inefficient model.

3.1. Splitting data into Training and Testing sets overview

Typically, a data scientist would use a framework for automatically splitting the available data into mutually exclusive datasets for training and testing. According to [1] one popular framework used to do this is SciKit-Learn, which allows developers to split the size of dataset by

random selection. When assessing different models, or retraining models, it is important to override the random seed used to split the data. The outcomes would not be consistent, equivalent, or reproducible if not performed precisely. Typically, 70% - 80% of the data is used for training the model, with the remainder reserved for evaluation. There are numerous advanced methods available to ensure that the division of training and testing has been conducted in a representative way. When considering the coverage of model testing, it should be measured in terms of data, rather than lines of code.

The design of regression testing activities demands attention. In traditional software development the risk of functional regression is typically low unless significant changes are made. In the case of AI almost any change to the algorithm, model parameters, or training data usually needs the model to be rebuilt from scratch, and the risk of regression is very high for previously tested functionality. This is because 100% of the model could potentially change instead of a small percentage of the model based on the necessary modifications. In synopsis:

- The way that the initial data has been gathered is important to understand whether it is representative.
- It is important that the training data is not used to test the model otherwise testing will only appear to pass.
- The data split for training and testing may need to be made reproducible.

Improper splitting of datasets into training, validation, and testing sets will contribute to overfitting and underperformance in production. For instance, if a trained model expects a certain input feature, but at inference time if that feature is not passed to the, the model will fail to render a prediction. Other times however, the model will quietly crash. One of the most critical challenges for a data scientist to consider is data leakage. If one does not how to prevent data leakage, the leakage will come up often, and will destroy the models in the most subtle and dangerous ways. Particularly, leakage causes a model to look accurate and precise until one starts making decisions with the model, and then the model becomes very imprecise.

In this paper, we attempt to understand and propose a means to ensure that the data that is split between the training data set and testing data set robustly represents the domain of the full dataset. In our examples, we are assuming that there is not data leakage where the same data points appear in both the training and the testing data sets. Random splitting of the data into training and test data sets while not leaving out key parts of the total data domain is a common challenge in AI applications. In [2], we see an example in source code summarization. In attempting to generate natural language descriptions in source code, LeClair and McMillan explore the efficacy of splitting the data by method or by project. This is a very specific and narrow use case. We are seeking a methodology that is more generic and has the potential to be of use in a variety of use cases.

3.2. Example of a Poor Split

To help explain the challenge, let us consider a dataset representing flowers. To help visualize the problem, Figures 1 and 2 display a constructed collection of data. In both figures, the letter of the datapoint represents a particular type of flower. The y-axis represents plant height and the x-axis represents petal width. There may be several other attributes included in the dataset, such as leaf shape, flower color, and thorn presence. It is popular to use an 80/20 split when dividing data into training and test data sets, where we use 80 percent of the data to train the model and reserve 20 percent of the data to test the model. Both Figure 1 and Figure 2 show a splitting of

the data by brute force that results in a poor split. In figure 1, very little data for the P flower variety is included in the test data set while in Figure 2, very little data for G variety of flower is seen. This poses a serious problem for training the model. There is one flower variety in either case that is highly under-represented in the test data. In training, we can expect the model to learn effectively on 4 of the 5 flower varieties, but it won't learn much about the fifth variety. Almost all the test results, however, will be for a flower variety that the model did not learn. Thus, we would expect a prominent loss during the testing of the model and ultimately an ineffective model.

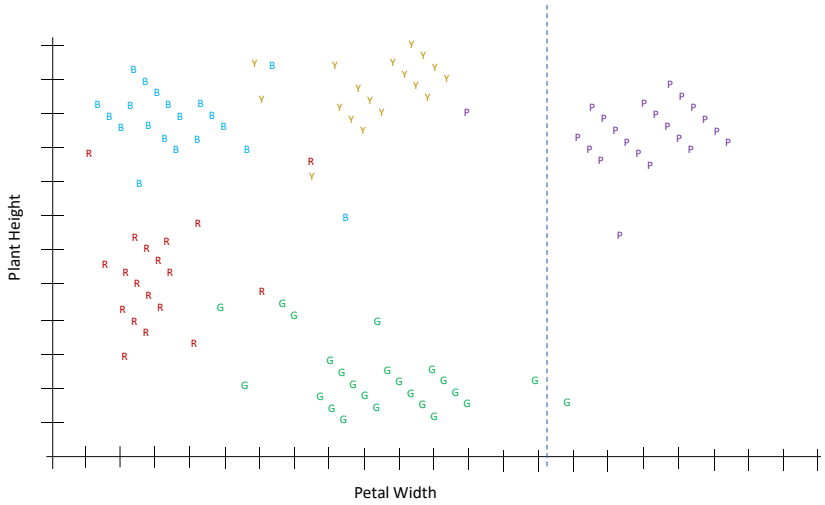


Figure 1 Data split example

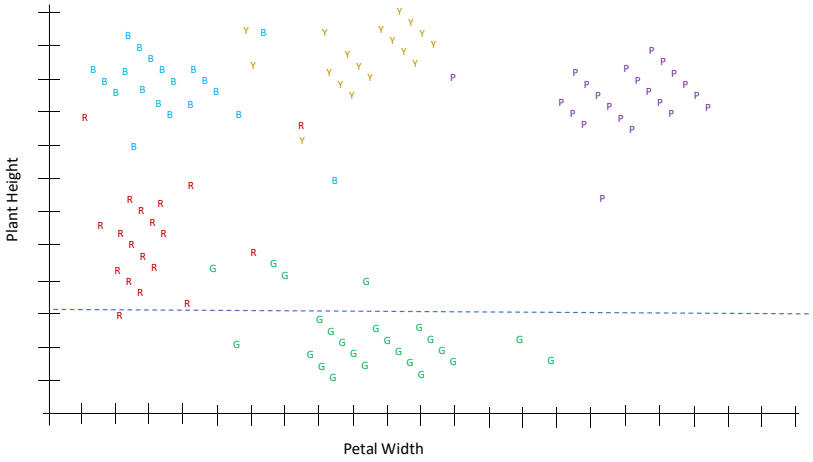


Figure 2 Data split example

Of course, in the real world, none would split data in the above presented brute force manner. We would really like to pick data points for the training and for the test data sets at random. A straight random data split could have a complete data domain coverage such that there are no holes that will fail to learn about an important subset of the data from the model. However, it is not difficult to imagine that there is a chance of under representing 1 or 2 of the flower varieties by a strictly random split of the data. It would be ideal if we could apply a bit of intelligence to the splitting of the data while retaining an approach that still arbitrarily selects individual data points randomly.

3.3. k-Means Clustering

Clustering is a technique of Machine Learning technique that involves data point grouping. Theoretically, data points that are in the same group should have similar properties or characteristics, while data points from different clusters should have dissimilar properties or characteristics. In clustering, we do not have a target to predict; rather the model will understand the data and try to club similar observations and form different clusters. Hence, it is an unsupervised learning model. There are different clustering algorithms in AI that are available, and each algorithm has its own purpose.

We will be concentrating on K-Means clustering in our paper. K-means is one of the simplest algorithms for unsupervised learning that follows a simple and uncomplicated way of classifying a data set through a variety of clusters. The main objective is to define k centers, one for each cluster. As per [14], the algorithm is comprised of the following steps:

1) Let X = { x1, x2, x3,…,xn} represent the set of data points, and ,V={v1,v2,v3…,vn} the set of centers.
2) Randomly select ‘c’ cluster centers.
3) Calculate the distance between each data point and cluster centers.
4) Assign the data point to the cluster center whose distance from the cluster center is minimum of all the cluster centers.
5) Recalculate the new cluster center using:

$$v_i = (1/c_i) \sum_{j=1}^{c_i} x_i$$

Where, ‘ci’ represents the number of data points in ith cluster.

6) Recalculate the distance between each data point and newly obtained cluster centers.
7) If no data point was reassigned then stop, else, repeat from step 3.

The algorithm is significantly sensitive to the initially selected random cluster centers. The k-means algorithm can be run multiple times to reduce this effect. The results depend on the value of k and there is no optimal way to describe a best “k”.

3.4. Decision Tree

Decision tree is a machine learning prediction technique. Decision tree builds by repeatedly splitting data into smaller and smaller samples, Decision trees are trained by passing data down from a root node to leaves. The data is then repeatedly split according to predictor variables so that the child nodes are more “pure” or identical in terms of the outcome variables. One of the predictor variables will be chosen to make the root split. This creates a leaf node, which will further split into child nodes. All the leaves either contain only one class of outcome or they are too small to split further. At every node, a set of possible split points is identified for every predictor variable. The algorithm calculates the improvement in purity of the data that would be created by each split point of each variable. The split with the greatest improvement is chosen to partition the data and create child nodes. Calculating the improvement for a split [15], when the outcome is numeric, the relevant improvement is the difference in the sum of squared errors between the node and its child nodes after the split. For any node, the squared error is:

$$\sum_{i=1}^{n} (y_i - c)^2$$

Where n is the number of cases at that node, c is the average outcome of all cases at that node, and y_i is the outcome value of the i^{th} case. If all the y_i are close to c, then the error is low. A good clean split will create two nodes, both which have all case outcomes close to the average outcome of all cases at that node. When the outcome is categorical, the split may be based on either the improvement of Gini impurity or cross-entropy:

$$Gini\ impurity = \sum_{i=1}^{k} p_i(1 - p_i) cross - entropy = -\sum_{i=1}^{k} p_i log(p_i)$$

where k is the number of classes and pi is the proportion of cases belonging to class i. These two measures give similar results and are minimal when the probability of class membership is close to zero or one. Let us consider the class's red and blue with sample data points from the example above and calculate the Gini impurity as shown below:

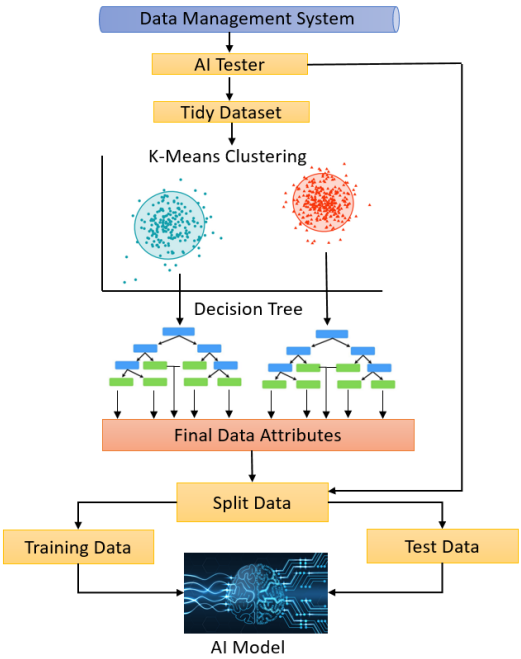
Node	Class: red	Class: blue	Gini
count	10	5	
proportion	67%	33%	
p (1 - p)	0.222	0.222	0.444

Child 1	Class: red	Class: blue	Gini
count	7	1	
proportion	88%	13%	
p (1 - p)	0.109	0.109	0.219

Child 2	Class: red	Class: blue	Gini
count	3	4	
proportion	43%	57%	
p (1 - p)	0.245	0.245	0.490

The initial node contains 10 red and 5 blue cases and has a Gini impurity of 0.444. The child nodes have Gini impurities of 0.219 and 0.490. Their weighted sum is (0.219 * 8 + 0.490 * 7) / 15 = 0.345. Because this is lower than 0.444, the split is an improvement. Similarly, at every node, the purity will be determined and the model will decide whether further splits are needed.

3.5. Application of k-Means Clustering and Decision Tree for Accurate Data Split



The above architecture demonstrates the schematic overview of the main phases of our data split process. The architecture consists of five phases: Tidying phase, Clustering (k-Means clustering) phase, Decision tree phase, Data split phase and Training phase.

Data to be split for training and testing phases are stored in a data management system, which is confined to an unsupervised specific type of data. In unsupervised data, there are no output variables to predict. Input data are not labeled and do not have a known result.

The first phase of our architecture is tidying the dataset, which is a crucial part of our proposal. Tidy data is a standard way of mapping the meaning of a dataset to its structure. A dataset is messy or tidy depending on how rows, columns, and tables line up with observations or data points, variables, and types. Around 80% of their time is spent by Data Scientists in cleaning, structuring and organizing the data. Tidy data is a way of structuring datasets to simplify analysis. In tidy data: Each variable must have its own column, each observation must have its own row, and each type of observational unit forms a table. [16] Messy data is any other arrangement of the data and it can be of these types:

- Column headers are values, not variable names.
- Multiple variables are stored in one column.
- Variables are stored in both rows and columns.
- Multiple types of observational units are stored in the same table.
- A single observational unit is stored in multiple tables.

There are more types of messy data not mentioned here, but they can be tidied in a similar way.

In our approach, the AI tester is responsible for tidying the dataset, to perform this activity. The AI tester must have a sound knowledge of Artificial Intelligence and Data Mining. It is also important to train the tester on how to understand the data, how to clean the data, and how to restructure messy data into a comprehensible format. There are several dataset tidying tools

available, XLMiner being the robust data mining add-in for Excel that could be used to tidy the dataset. Once the dataset is tidy in nature, the AI tester can pass the dataset to the Clustering algorithm, which is the next phase of our architecture.

The K-Means clustering phase involves the grouping of observations that have similar properties or features, while data points from different clusters should have different properties or features. In clustering, we do not have a target variable to predict, rather the algorithm will understand the data and will group similar observations or characteristics to form different clusters. K-means is one of the simplest unsupervised learning algorithms that follows simple and uncomplicated methods to classify a data set into a variety of clusters. The main objective is to define k centers, one for each cluster. K value can be passed as an input parameter to the clustering algorithm to determine how many clusters we need, but if we do not pass k value, after going through many iterations, the algorithm itself will attempt to group the observations into different clusters. In our architecture, we have considered two clusters as an example of grouping all the similar observations in to two distinct clusters. After the clusters are created, the output data from each cluster will be passed as an input to the affiliated decision tree for next phase.

During the decision tree phase, each decision tree builds by repeatedly splitting the input data passed from the respective cluster into smaller and smaller samples. Decision trees are typically trained by passing data down to leaf nodes from a root node. The data splits repeatedly according to predictor variables so that child nodes are more "pure" or identical in terms of the outcome variables. All the leaves either contain only one class of outcome or are too small to split further. As mentioned in the previous section regarding the Gini impurity, the decision tree will split the nodes based on the Gini value and determine whether or not the split is necessary. Eventually, the decision tree produces different output attributes or variables which contain only one class of values per attribute with good purity.

In the next data split and training phase, the AI tester collects the data attributes from each leaf of the decision tree and splits the data into training and testing sets. There is no standard percentage to decide how much to select for training and testing sets. It all depends on how much data the AI tester has available. If the AI tester has a large data set, the 75:25 training and testing ratio is okay to consider. Even if you get a very good precision after training the model with the training dataset, there is no guarantee that your trained model is a generalized one. One explanation for a non-generalized model is having a small training set, and the models appear to over-fit for small training sets. Testing your model with distinct input combinations will therefore allow you to conclude whether your model is generalized. If you have a tiny data set, however, it is easier to go with 90:10. Another point to consider is Cross-validation, sometimes called rotation estimation, or out-of-sample testing. [17] It helps to assess how the findings of a statistical analysis will generalize to an independent data set. In the next section we will go through an example of exactly how our proposed architecture generates the accurate output.

3.6. Example Results

Let us now return to the example of flower data and apply the architecture to use k-means clustering and then the decision tree. To begin with as shown in Figure 4 we want the dataset to be tidy. Each column is a variable whereas each row is an observation of one flower. The AI Tester must determine how much data to use for training and how much to reserve for testing. A common selection is 80% for training and 20% for testing, but this is really at the discretion of the tester. The AI tester must also have data domain knowledge and be sufficiently familiar enough with the k-means clustering principle to select a reasonable value for k. In this example, we have chosen k = 5. Figure 3 then shows a likely result of clustering the flower data into 5 clusters.

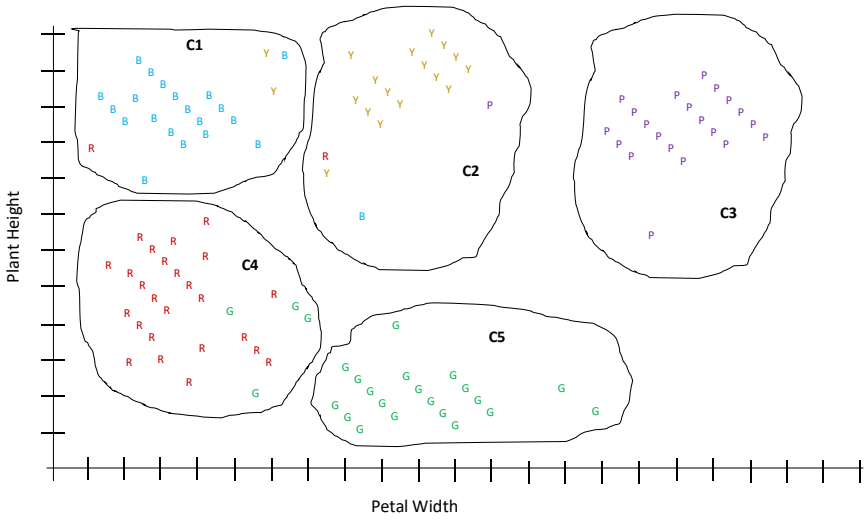


Figure 3 Data Clustered

Next, we move on to the decision tree. Here we begin to consider the flower data's additional key attributes. For this example, we will look at the leaf shape, flower color, and the presence of thorns as three additional key attributes to use in the decision tree. We must apply a decision tree to each of the five clusters. For the sake of brevity, let us concentrate on only one of the clusters. The focus will be Cluster 4.

Leaf Shape	Flower Color	Has Thorns
Round	Red	Thorns
Round	Red	Thorns
Round	Red	Thorns
Oval	Yellow	Thorns
Round	Red	Thorns
Oval	Purple	Thorns
Round	Red	Thorns
Oval	Yellow	Thorns
Oval	Purple	No Thorns
Round	Red	Thorns
Oval	Purple	Thorns
Round	Red	Thorns
Oval	Blue	Thorns
Round	Red	Thorns
Round	Red	Thorns
Round	Red	Thorns
Oval	Purple	No Thorns
Oval	Yellow	Thorns
Oval	White	No Thorns
Round	Red	Thorns
Round	Red	Thorns
Round	Red	Thorns
Oval	Yellow	Thorns
Oval	Purple	Thorns
Round	Red	Thorns
Oval	White	No Thorns
Round	Red	Thorns
Oval	Blue	Thorns
Round	Red	Thorns

Figure 4 Cluster 4 Attributes

In cluster 4, there are two varieties of flower represented. The R variety is the bulk of the data points, but there are also some data points of the G variety. Application of a decision tree to this data helps to identify key subgroups within the cluster. The end result of implementing a decision tree is shown in Figure 5.

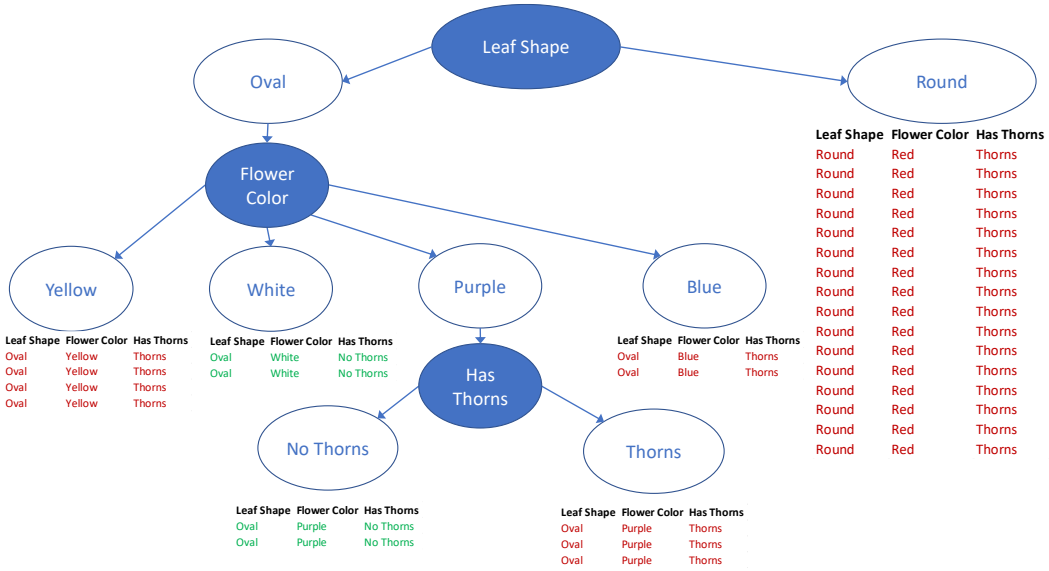


Figure 5 Decision Tree Applied to Cluster 4

Six key data subgroups within cluster 4 were revealed by the decision tree. At this point, the tester applies the chosen percentage split to each key data subgroup from each cluster. The effect is a test data set that covers the entire data set domain robustly and, essentially, an efficient model where the loss seen when testing the model is very similar to the loss seen when training the model.

3.7. Related Work

Moderate research exists in terms of software testing in AI. In [14] the key challenges of validating the quality of AI software have been summarized, which includes how to define quality assurance standard systems and develop adequate quality test coverage [9], how to connect quality assurance requirements and testing coverage criteria for AI systems based on big data, and how to use systematic methods to develop quality test models. [11] presents several new challenges that AI systems are facing in predicting system behavior such as determining the exact pre-conditions and inputs to obtain an expected result, defining expected results and verifying the accuracy of test outputs, and measuring the test coverage. Also [11] has interpreted the numerous challenges in test generation on the code, unit, module, or component levels. Generating tests from code makes it very challenging for AI to understand the state of the software and its data and necessary dependencies. Parameters can be complex and goals and output optimizations may be unclear. The challenges that the model has in adapting itself to function more accurately in identification of gender images [12] have been clearly described. [1] Focused on the challenges faced in terms of testing facial recognition in AI systems. Data privacy is another big challenge in AI methodologies because of predictive analysis. Organizations are concerned with the transparency of their personal data [13]. Due to the disparity in training data and test data collection, overfitting is another problem facing AI models today[4]. [11] defines issues with the integration testing of the AI system in terms of transformation, cleaning, extraction, and normalization of data.

4. CONCLUSION

Software testing is just as critical in AI/ML applications as it is in any other type of software development. Due to the nature of how AI systems work and are developed, there are many difficulties that the software testers face with AI applications This paper attempted to lay out one potential solution to the problem of splitting data into training and testing data sets to ensure that the training data set is selected in such a way that it effectively covers the entire dataset domain. The aim is to train a successful model. The architecture in this paper sets out a methodology that increases the odds of a quality data split. It begins with a tidy data set that is the input to the k-means clustering phase to identify natural groupings of data points with similar characteristics. Next, each cluster of data becomes an input to the decision tree phase to further break each cluster into smaller samples of related data points. Finally, on each leaf node of each decision tree, we perform the actual splitting of the data. We have provided an example of how this could work. We look forward to applying the architecture to a number of applications and working to perfect it for further study.

REFERENCES

[1] H.Zhuy et al., "Datamorphic Testing: A Methodology for Testing AI Applications", Technical Report OBU-ECM-AFM-2018-02.

[2] Chen, TsongYueh, et al. "Metamorphic testing: A review of challenges and opportunities." ACM Computing Surveys (CSUR) 51.1 (2018): 1-27.

[3] David Fumo, https://towardsdatascience.com/types-of-machine-learning-algorithms-you-should-know-953a08248861.

[4] Salman, Shaeke, and Xiuwen Liu. "Overfitting mechanism and avoidance in deep neural networks." arXiv preprint arXiv:1901.06566 (2019).

[5] Partridge D. "To add AI, or not to add AI? In Proceedings of Expert Systems, the 8th Annual BCS SGES Technical conference", pages 3-13, 1988.

[6] Partridge, D." The relationships of AI to software engineering", Software Engineering and AI (Artificial Intelligence), IEE Colloquium on (Digest No.087), Apr 1992.

[7] Last, M., Kandel, A., and Bunke, H. 2004 Artificial Intelligence Methods in Software Testing (Series in Machine Perception &Artifical Intelligence "Vol. 56). World Scientific Publishing Co., Inc.

[8] ManjunathaKukkuru, "Testing Imperatives for AI Systems", https://www.infosys.com/insights/ai-automation/testing-imperative-for-ai-systems.html.

[9] Du Bousquet, Lydie, and Masahide Nakamura. "Improving Testability of Software Systems that Include a Learning Feature." Learning 12 (2018): 13.

[10] J. Gao, et. al., "What is AI testing? and why", 2019 IEEE International Conference on Service-Oriented System Engineering (SOSE), DOI: 10.1109/SOSE.2019.00015.

[11] Alliance For Qualification, "AI and Software Testing Foundation Syllabus", 17 September 2019,

[12] S. Wojcik, E. Remy, "The challenges of using machine learning to identify gender in images", 5 September 2019,

[13] Tom Simonit, "Microsoft and Google Want to Let AI Loose on Our Most Private Data", "MIT Technology Review", April 19, 2016,

[14] Sanatan Mishra, Technical Article "Unsupervised Learning and Data Clustering" 2019

[15] Jake Hoare, Data scientist https://www.displayr.com/how-is-splitting-decided-for-decision-trees/

[16] Rodrigo Mariono, https://towardsdatascience.com/whats-tidy-data-how-to-organize-messy-datasets-in-python-with-melt-and-pivotable-functions-5d52daa996c9

[17] Research Scholar Analysis review " AI Data Split" Research Gate, 2019