A Hybrid Decision Tree-Neural Network (DT-NN) Model for Large-Scale Classification Problems

Jarrod Carson Kane Hollingsworth Rituparna Datta George Clark Aviv Segev
Department of Computer Science
University of South Alabama
Mobile, AL, USA
{jmc1627@jagmail., kmh1622@jagmail., rdatta@, georgewclark@, segev@}southalabama.edu

***Abstract*—As the Age of Information has evolved over the last several decades, the demand for technology which stores, analyzes, and utilizes data has increased substantially. Countless industries such as the medical, the retail, and the aircraft rely on this technology to guide their decision making. In the present paper, we propose a hybrid machine learning algorithm consisting of Decision Trees and Neural Networks which can effectively and efficiently classify data of varying volume and variety. The structure of the hybrid algorithm consists of a decision tree where each node of the tree is a neural network trained to classify a specific category of the output using binary classification. The data with which we used to train and test the classification ability of our algorithm is the Federal Aviation Administration's (FAA's) Boeing 737 maintenance dataset which consists of 137,236 unique records each composed of 72 variables. We perform this by classifying the discrepancy, or cause, of the incident into whether or not the incident occurred during scheduled maintenance operations and then further classifying specific details relating to the incident. Our results indicate that our hybrid algorithm is able to effectively classify incidents with high accuracy and precision. Additionally the algorithm is able to identify the most significant inputs regarding a classification allowing for higher performance and greater optimization. This demonstrates the algorithm's applicability in real-world scenarios while also showcasing the benefits of combining decision trees and neural networks as opposed to using them individually.**

***Index Terms*—Decision Trees, Machine Learning, Neural Networks, Hybrid Learning, Supervised Learning**

I. Introduction

Machine Learning (ML), a subfield of Artificial Intelligence (AI) and Data Science, provides effective algorithms which accomplish the task of prediction and classification from structured and unstructured data. However, due to the increase in data volume and complexity, in recent years this field faces many challenges when designing and implementing algorithms capable of efficiently and effectively processing the data. As such there has been significant research conducted to create more robust algorithms. ML consists of two main stages: training/learning and testing/predicting. Training/learning involves inserting a dataset with known characteristics into a ML learning model in order to "train" it. Afterwards the model will be able to make predictions based on similar data given to it in the testing/predicting stage. There are a few recent overviews of ML in different domains [1]–[3]. To alleviate the weakness in individual machine learning techniques, hybrid approaches are used.

A hybrid model which is a combination of two models, has started gaining significant attention in machine learning to classify / predict performance as compared to a single learning model [4]–[7]. Neural networks already have been used in numerous fields of science and engineering due to its capability of prediction for new data after the training is performed with existing dataset. On the other hand, decision tree has capability for feature classification. In the present work, a hybrid approach is proposed combining decision tree and neural networks; a neural network is integrated in each node of the decision tree. The method utilizes multiple neural models which are each individually trained and meshed together in a way to where the results actually perform better than if the hybrid method was not implemented. The children of any given node represent subcategories derived from the parent node which may be classified as well. This allows for the algorithm to classify data of varying degrees of granularity. The hybrid approach that we propose utilizes the binary classification category of machine learning and transforms each neural network into an individual node in a binary tree. Each node in the tree is predicting for a specific detail. The further down the tree that the predictions go the more detailed the prediction actually becomes. The height of the binary tree that is structured represents the amount of details that the outcome actually predicts for.

This ensemble learning method that is proposed is tested on a public Boeing 737 dataset. It is essential to train and test the neural model that is created on a public dataset rather than a dataset that was made in a controlled environment. Training the models on a dataset that was obtained from Boeing 737 validates the results that are produced and shows that this ensemble learning approach can provide improved results when used on a dataset that is found in the real world. The dataset that was used included 72 unique variables and 137,236 records from the Boeing 737 that were used as input for the network to train on with each network having a single binary neuron classifying whether the record belongs to the specified category or not after being preprocessed. After being preprocessed, the data was then split into training, testing, and validation. The testing data is used to test the accuracy and F1 of the neural network. The validation data makes sure that there is no overfitting.

II. Related Work

The first concepts of machine learning started out with the study of the brain and how neurons fire off when certain external events occur [8]. Ensemble learning is the concept of using multiple neural models/training models together to create a more accurate algorithm than the original one working alone. Our algorithm involves using binary classifying neural models in ensemble with a binary decision tree in order to provide for a more accurate prediction when being compared to just using a single neural model for the predictions.

Fatath proposed a hybrid model integrating the maximum entropy model, support vector machine, and naive-Bayes for multi document text summarization [9]. To improve the classification accuracy, Polat and Gunes [10] proposed a hybrid machine learning model for multi-class problems. The method consists of the C4.5 decision tree classifier and one-against-all approach. The efficacy of the hybrid method was shown on image segmentation, dermatology, and lymphography open source datasets. A recently developed hybrid method also used decision tree, random forest, and gradient boosting for water quality prediction. The method is known as complete ensemble empirical mode decomposition with adaptive noise (CEEMDAN). The variants of the method are proposed; one is based on gradient boost (CEEMDAN-XGBoost) while the other is based on random forest (CEEMDAN-RF). Interestingly, both the methods proved their superiority to predict different parameters [11]. Arabasadi et al. [12] combined a neural network with a genetic algorithm to increase the performance of the neural networks and the hybrid method was applied on a heart disease dataset. A machine learning ensemble technique is proposed by Pham et al. [13] to assess landslide susceptibility. In the ensemble method, multi layer perceptron (MLP) neural network is integrated with AdaBoost, Bagging, Dagging, MultiBoost, Rotation Forest, and Random SubSpace. Another hybrid prediction model is developed by Chen et al. [14]. In that method, K-means clustering is integrated with the J48 decision tree for the diagnosis of Type 2 diabetes. Lin et al. [15] dealt with the data from manufacturing industries for condition based maintenance using MapReduce and multiple classifier types decision trees with dynamic weight adjustment. A just-in-time defect prediction method was proposed by Yang et al. [16] using ensemble learning. The prediction method is also capable of handling redundancy and data imbalance in addition to ensure robustness. Another study based on just-in-time prediction also used ensemble learning. The authors proposed a two-layer ensemble learning approach (TLEL) based on decision trees [17]. The outer layer uses different Random Forest models for training whereas the inner layer is an integration of decision tree and bagging to build a Random Forest model. Another recent study [18] used ensemble learning for better predictive performance of remaining useful life (RUL) of aircraft engines and used several methods like multiple base learners, including random forests (RFs), classification and regression tree (CART), recurrent neural networks (RNN), autoregressive (AR) model, adaptive network-based fuzzy inference system (ANFIS), relevance vector machine (RVM), and elastic net (EN). Moreover, the method also used particle swarm optimization (PSO) and sequential quadratic optimization (SQP) to achieve the best combination of weights to be used in base learners. An extension of the above study is done by Li et al. [19]. The study used directed acyclic graph (DAG) hybridized with long short term memory (LSTM) and a convolutional neural network (CNN) to predict the RUL. The method is tested with a turbofan engine degradation simulation dataset provided by NASA. We [20] proposed a predictive maintenance strategy for Boeing 737 aircraft using the integrated decision tree-neural network model. Marcello et al. [21] proposed an ensemble learning based big data model for failure rate of equipment subject to different operating conditions. Another recently developed method [22] also used ensemble learning, which is also capable to handle imbalance data. The method uses adaptive boosting (AdaBoost) and random forests (RF). The state of the art ensemble learning can be found in [23]–[25]. Taking the vast literature into consideration, we propose a technique by integrating a neural network in each node of the decision tree.

III. Decision Tree-Neural Network (DT-NN)

A. Integrating Decision Trees and Neural Networks

We hybridize a neural network with a decision tree in the present work. The motivation is the optimization of a single decision in a classification. A neural network is integrated at each node of the decision tree as shown in Fig. 1. This integration of both the decision tree and the neural network approach are superior as compared to both individual methods. The performance of the neural networks are well suited for classification into categories where the boundaries of classification are less distinct. However, the performance of the neural network decreases with the increase in number of categories. The decision tree works with a large number of categories which are distinctly classified. The decision tree is constructed based on a set of binary possible outcomes such as 0, 1. To achieve the same, the best possible result attribute with the highest information gain is selected. To define information gain, we define a measure commonly used in information theory, called entropy, which characterizes the (im)purity of an arbitrary collection of examples.

Entropy $H(S)$ is defined as a measure of the amount of uncertainty in any dataset S

$$H(S) = \sum_{c \in C} -p(c) log_2(p(c))$$

Where

S - The dataset for which entropy is being calculated in the current iteration.

C - The set of the classes in $S, C = 0, 1$.

$p(c)$ - The proportion of the number of elements in class c to the number of elements in set S.

If $H(S) = 0$ then the set S is perfectly classified.

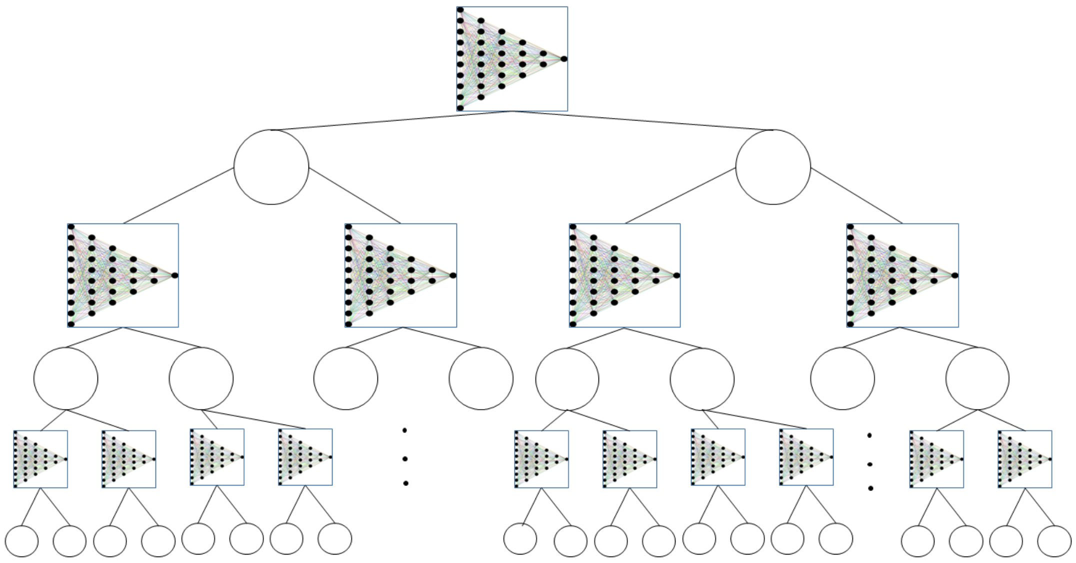


Fig. 1: Hybrid Decision Tree-Neural Network (DT-NN) Model

Information gain $IG(A)$ is defined as the measure of the difference in entropy from before to after the set S is split on a result attribute A. This quantity measures the extent of uncertainty S was reduced after splitting set S on result attribute A.

$$IG(A,S) = H(S) - \sum_{t \in T} p(t)H(t)$$

Where,

$H(S)$ - Entropy of set S.

T - The subsets created from splitting set S by result attribute A such that

$$S = \bigcup_{t \in T} t.$$

$p(t)$ - The proportion of the number of elements in t to the number of elements in S.

$H(t)$ - Entropy of subset t.

The information gain can be estimated for each remaining attribute. The attribute with the largest information gain can be used to split the set S on each iteration. Thereafter, with the largest information gain, a neural network is built. For each binary classification, a neural network which is constructed to classify only if the problem occurs. Each neural network at each node of the decision tree consists of all the result attributes which could lead to either 0 or 1.

B. Outline of the Method

The structure of the hybrid binary tree classifier gives us the flexibility to find more detailed or less detailed classification problems. The structure is formatted in a way so that the height of the tree represents the complexity of the prediction of the problem. The larger the tree, the more detailed an issue that the network can predict on. Each level of the tree is trained on different data that is pulled from the original dataset. If the first node in the tree was predicting for a crack in the airplane, it would assign the original dataset with a crack problem a 1 and all of the other entries without a crack problem 0. If we continued the tree to predict for something with a crack and a fuselage problem, we would separate the original dataset into two datasets, one with crack data and one without crack data, and assign each entry with a fuselage problem a 1 or a 0 and train a new model for each of the two subcategories. Since we split the dataset multiple times, this method would work best in an environment where there is a lot of data. This method of predicting for crack then predicting for fuselage in the binary tree format actually performs better than just predicting for entries that contain crack and fuselage or not from the beginning.

C. Data Preprocessing

Preprocessing the data is an essential part in order for the neural network to accurately predict. In order for the neural network to process the data, it must be converted into a numeric form. To do this, the data is processed in chunks 5000 records at a time. Each record in each chunk is looked at and every unique value is added to a dictionary and kept track of. This process of looking in the dictionary to see if that unique value has been added or not is done for each record. As a new value is added to the dictionary it is given a unique identifier which starts at 100 and increases by 100 for every unique value recorded that is a part of that record. If any entry in the record is found to be null, it is then assigned a -1 to separate it from the data that is not null. If a value is come across that is actually already in a numeric form, is ignored. After the dictionary is created, the dictionary is used with a mapping function alongside with the dataframe to map each of the values found in the dictionary with its corresponding numeric value. This process of remapping all of the corresponding variables is done with each 5000 record chunk in the dataset at a time. Doing this with only 5000 records at a time ensures that there is not a memory error with reading too many values into memory at a time. This technique of assigning a unique numeric value to all of the non-numeric values in the data is an essential step in order for the data to be able to be fed into the neural network for training and producing a working model.

D. *Pseudocode for Preprocessing*

The pseudo code for Neural Network used in hybrid approach is given in the following algorithm:

```
1:  CSV Data: Data read in from a csv file.
2:  trainData: Subset of CSVData used for training neural network.
3:  testData: Subset of CSVData used for testing neural network.
4:  validData: Subset of CSVData used for validating neural network.
5:  nn: H2O DeepLearningEstimator neural network model.
6:  nnMetrics: H2O dataframe containing performance metrics from testing the neural network.
7:  resultsCSV : CSV file for containing neural network performance metrics.
8:  ImportH2O, H2ODeepLearningEstimator
9:  CSV Data = open(”csvfile.csv”,”read”)
10: H2O.init()
11: H2O.read(CSV Data)
12: nn = DeepLearningEstimator(hiddenLayers, activationFunction)
13: nn.train(invars, outvars, trainData, validData)
14: nnMetrics = nn.test(testData).performanceMetrics
15: resultsCSV = open(”results.csv”, ”write”)
16: resultsCSV.write(nnMetrics) =0
```

IV. Experiments

A. *Dataset and Preprocessing*

The dataset that we used for the experiments, Boeing 737 data, which came from the Federal Aviation Administration, contains 137,236 records with each record having 73 variables. These records included data from aircraft that suffered issues dealing with mechanical issues with the aircraft. We chose this data for two main reasons. The first is because it is a real world dataset that was hand documented for what the issue is. Showing that this algorithm can work on a hand recorded dataset shows the robustness of the given algorithm. The second reason is because of the sheer size of the data that we are allowing our algorithm to be trained on. Having a dataset which contains a large number of real world records allows for us to make sure that the algorithm is able to handle complex inputs and perform with high accuracy. The way that we sectioned off this data to use for the neural network was by sectioning off randomly chosen data into 75% training, 15% testing, and 10% validation. Using these percentages for the neural network will allow the network to be trained and have the final output model be the best in terms of accuracy and F1 score.

TABLE I: Accident and Incident Data

Operator Control Number	Difficulty Date
Submission Date	Operator Designator
Submitter Designator	Submitter Type Code
Receiving Region Code	Receiving District Office
SDR Type	JASC Code
Nature Of Condition A	Nature Of Condition B
Nature Of Condition C	Precautionary Procedure A
Precautionary Procedure B	Precautionary Procedure C
Precautionary Procedure D	Stage Of Operation Code
How Discovered Code	Registry N Number
Aircraft Make	Aircraft Model
Aircraft Serial Number	Aircraft Total Time
Aircraft Total Cycles	Engine Make
Engine Model	Engine Serial Number
Engine Total Time	Engine Total Cycles
Propeller Total Time	Propeller Total Cycles
Part Make	Part Name
Part Number	Part Serial Number
Part Condition	Part Location
Part Total Time	Part Total Cycles
Part Time Since	Part Since Code
Component Make	Component Model
Component Name	Component Part Number
Component Serial Number	Component Location
Component Total Time	Component Total Cycles
Component Time Since	Component Since Code
Fuselage Station From	Fuselage Station To
Stringer From	Stringer From Side
Stringer To	Stringer To Side
Wing Station From	Wing Station From Side
Wing Station To	Wing Station To Side
Butt Line From	Butt Line From Side
Butt Line To	Butt Line To Side
Water Line From	Water Line To
Crack Length	Number Of Cracks
Corrosion Level	Structural Other
Discrepancy	

The first 72 variables are given as input into the neural network with the last variable, discrepancy, being the output for the network. To feed the data that we are working with, we must first format it using a preprocessing algorithm. The goal that our preprocessing algorithm accomplishes is that it turns string data into corresponding numerical keys which are all recorded in a dictionary. Performing this algorithm starts off with reading the data from a CSV file in chunks of 5000 records at a time using the Pandas Dataframe located in the Pandas library. Each of these records that are read in will then have corresponding values in a dictionary. Each unique entry per column will be entered into a dictionary with a corresponding numerical value. This unique identifying value for that column will then have a corresponding value of the string that was originally there. After another unique value is found in that column, this value increases by 100 and is assigned that. If a value in the Pandas dataframe is a NULL value the number -1 is assigned to it. If a numerical value is encountered, that value is ignored due to the fact that it could be an important value in determining the result. After all of this data is translated to its unique numerical identifier, it is then used to generate a CSV file with all of the numerical values. The final step in the preprocessing stage of the data is the final creation of the text file which contains the key to what numerical value maps to what for each column. If for example, the data that was encountered was 115.31, and the type for that cell was a real number, then that value will be ignored. If a piece of data that was a string, the value would be entered into the text file just as it was documented like in the CSV file.

V. Correlations

The Boeing 737 incident data contains 73 variables relating to mechanical issues. These variables are listed in Table I. These variables were categorized into 72 input variables and the remaining variable discrepancy, which represents the actual incident in the reports, was categorized as the lone output variable. The structure of our complete Neural Network (NN) consists of 72 inputs in an input layer that is connected to layers of hidden neurons producing one output for classification. During the training process, the NN performs two tasks, it determines the weight associated with each input and optimizes the classification decision. These tasks were examined with the goal of identifying and testing three correlations; 1) number of significant inputs to classification accuracy, 2) total number of inputs to classification accuracy, and 3) variables to all other variables.

A. Number of Significant Inputs Correlation

In order to determine the correlation of significant inputs to classification accuracy, the NN's determination of input weights was analyzed. During the NN training phase, input variables that do not contribute to the optimization of the classification are increasingly ignored and therefore the weights between the input layer and the first hidden layer of the NN are reduced. It is expected that in the final NN that input variables or features with low mean weight values have less affect on the optimized classification. For our analysis, we organize the input variables in descending weight order and test the NN as input variables are added and removed one at a time. The values for Area Under the Curve (AUC) and Accuracy for each of the six types of activation functions were then compared.

B. Number of Total Inputs Correlation

To determine the correlation of the total number of the input variables to classification accuracy, we analyzed the NN's accuracy as the amount of inputs are increased. The input variables were again organized in descending order of the mean value of the weight connecting the input layer and the first hidden layer. Again, input variables were added and removed one at a time. A comparison was made for the values of Area Under the Curve (AUC), Accuracy, Precision, Recall, and F1 versus the total number of inputs.

C. Correlation of Variables to all Other Variables

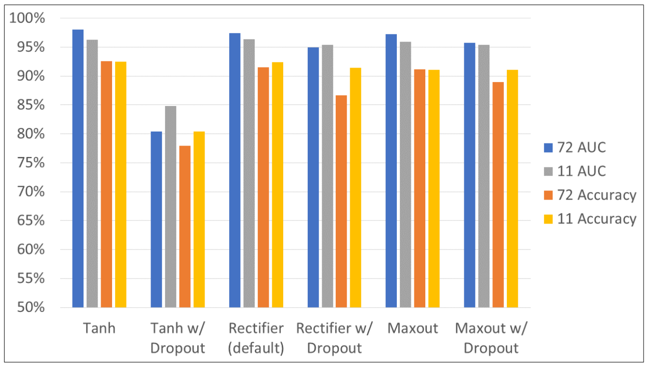
The correlation of each input variable with every other input variable of the NN was determined by using the Pearson and Spearman statistical correlation algorithms. The Pearson correlation was used to identify any linear change relationships between the input variables. The Spearman correlation was employed to identify monotonic relationships between the input variables. For both correlation algorithms, heatmaps were produced showing the level of correlation of the variables. Additionally the variables with the highest correlation with the Discrepancy output were identified using both correlation algorithms.

TABLE II: Pearson Correlation

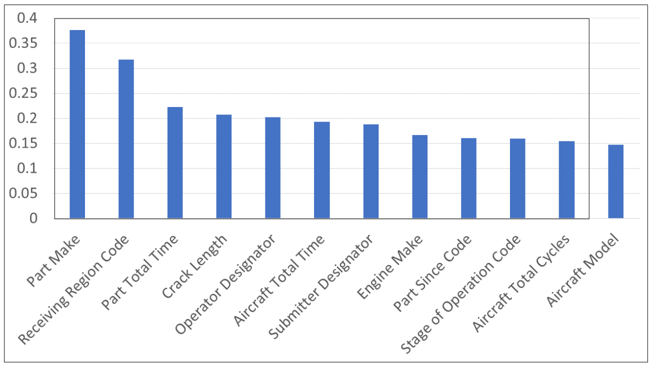
Variable Names	Discrepancy
OperatorControlNumber	-0.162920734
OperatorDesignator	-0.191397581
SubmitterDesignator	-0.139558448
SubmitterTypeCode	-0.114425491
ReceivingRegionCode	-0.030500605
JASCCode	0.098274881
StageOfOperationCode	-0.238238549
HowDiscoveredCode	0.032754564
RegistryNNumber	-0.351326718
AircraftModel	-0.248164469
AircraftSerialNumber	-0.061595756
AircraftTotalTime	0.225559051
AircraftTotalCycles	0.425399651
EngineMake	-0.217505143
PropellerTotalCycles	-0.035912637
PartSerialNumber	-0.015229973
PartTotalTime	-0.084394422
PartTimeSince	-0.062439797
PartSinceCode	0.28765802
ComponentSinceCode	-0.017284174
FuselageStationFrom	0.029334189
FuselageStationTo	-0.028950023
StringerFrom	0.009530162
StringerFromSide	0.19310386
CorrosionLevel	-0.148433297

TABLE III: Spearman Correlation

Variable Names	Discrepancy
OperatorControlNumber	-0.162920813
OperatorDesignator	0.106873514
SubmitterDesignator	0.158101481
SubmitterTypeCode	-0.129345355
ReceivingRegionCode	0.048436281
JASCCode	0.118254811
StageOfOperationCode	-0.271904763
HowDiscoveredCode	0.048619706
RegistryNNumber	-0.340778447
AircraftModel	-0.199877064
AircraftSerialNumber	0.010545024
AircraftTotalTime	0.34909541
AircraftTotalCycles	0.43531384
EngineMake	-0.236875842
PropellerTotalCycles	-0.035912637
PartSerialNumber	-0.071002955
PartTotalTime	-0.239128945
PartTimeSince	-0.117941956
PartSinceCode	0.341907897
ComponentSinceCode	-0.021837315
FuselageStationFrom	0.111152493
FuselageStationTo	-0.0815809
StringerFrom	0.188622223
StringerFromSide	0.206177495
CorrosionLevel	-0.151251426



(a) AUC/Accuracy of All Inputs vs. Significant Inputs



(b) Average Mean for Leading First Layer Input Weights

Fig. 2: Significant Inputs

VI. Results

A. *Number of Significant Inputs Correlation*

Figure 2a shows the comparison of classification results into Maintenance and Non-Maintenance categories by the number of inputs. In particular, the figure shows the Accuracy and AUC results when using all 72 input variables as opposed to using the top eleven mean weight input variables. The results are further broken down by the six activation functions tested in the NN. The figure shows that the accuracy is nearly identical for the activation functions regardless of whether all 72 input variables are used or only the top eleven mean input variables are used. The accuracy difference ranged from 0.12% less accurate with all 72 input variables with the tanh activation function to 4.77% more accurate when only the top eleven mean weight input variables were used with the rectifier with dropout activation function. Figure 2a also shows that the AUC comparisons were more varied. In this case, the range in AUC was -1.76% using the tanh activation function with all variables to 4.44% using the tanh with dropout for the top eleven mean weight input variables.

In Figure 2b we show the average mean for the leading inputs weight value between the input layer and the first layer of the NN. The top 11 variables in the circumference box have a mean weight of 0.15 or greater and are the main variables relevant for high accuracy results from the NN. The results show that issues such as Part Make, Receiving Region Code, and Part Total Time can clearly be identified as the most relevant classifiers. The Aircraft Model is already identified as a less unique classifier for the type of issue involved.

B. *Number of Total Inputs Correlation*

Figure 3 details our findings for the correlation of the number of inputs to AUC, Accuracy, Recall, and Precision. Figures 3a and 3b show that both the NN's AUC and accuracy increase initially as the number of inputs into the NN increase. However, both AUC and accuracy are less affected as the number of inputs continue to increase. AUC increases only until 16 total inputs, are used and accuracy of the NN does not improve after the 11 inputs which were identified as the important variables.

The AUC difference can be viewed as a less accurate value for measuring performance. In this case, it can be attributed to the low number of values measured to create the curve. This could explain the difference when measuring the area with AUC versus comparing a single Accuracy result.

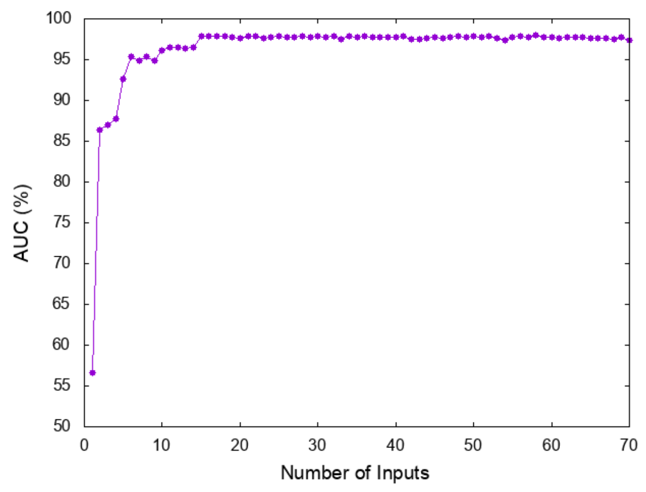
Figure 3c shows that there is a small drop in recall from 100% to 85% as more values are added. Figure 3d shows the Precision as the number of variables increases with the mean weight decreasing. The precision value increases and then stabilizes once the top 11 weighted variables are used. The results show that precision is determined by the previously identified significant variables while Recall is only slightly affected by an increase in input variables.

When viewing the F1 value appearing in Figure 4a, these findings are more evident. F1 increases as strongly weighted variables are introduced and peaks at 11 variables. The F1 becomes stable at around 90% using just 11 variables. Additionally, Figure 4b shows that most outcomes are clustered in the top right except for the high-recall initial values.

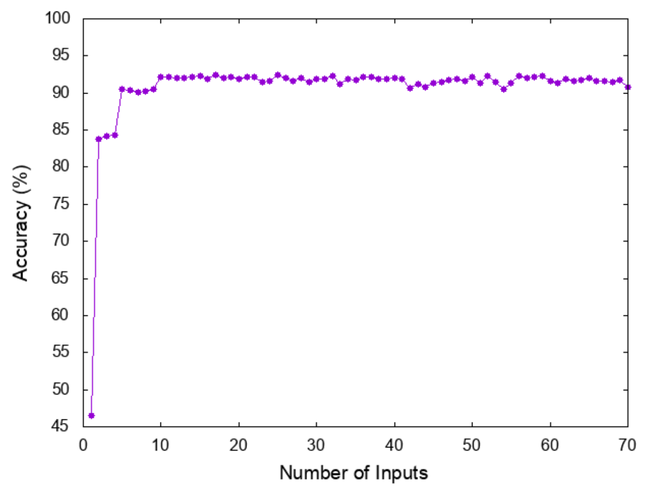
The results show the correct identification of the significant inputs by the method of classifying mean weights in descending order. The additional input variables, which do not seem to improve the results, can be attributed to constant values, variables which are dependent on other inputs, or values which are inconsistent with the expected results.

C. *Correlation of Variables to All Other Variables*

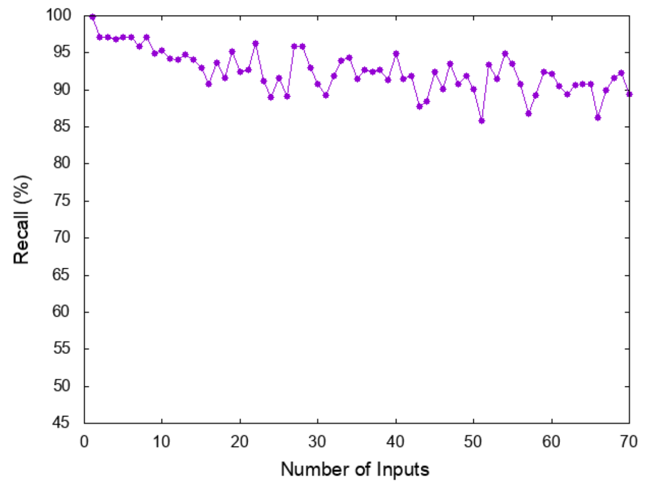
The heatmaps that are represented in Tables II and III demonstrate the correlation of each variable to every other variable using the Pearson/Spearman statistical correlation algorithm to find out how closely associated each variable is with another. In these tables, a 25 input sampling of the 72 input variables is shown. The closer the number is to +1, the higher the positive linear correlation is with the variable being compared and the greener the area is in the heatmap. The closer the number is to -1, the higher the the negative linear correlation is with the variable being compared and the redder the area is on the heatmap. When the number is zero,



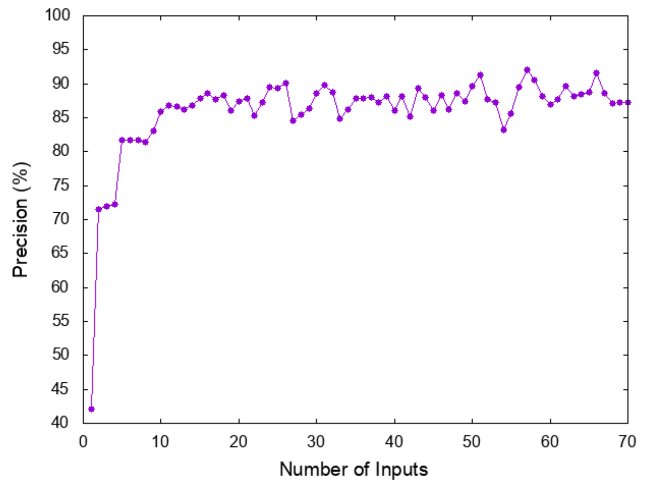
(a) AUC vs. Inputs



(b) Accuracy vs. Inputs

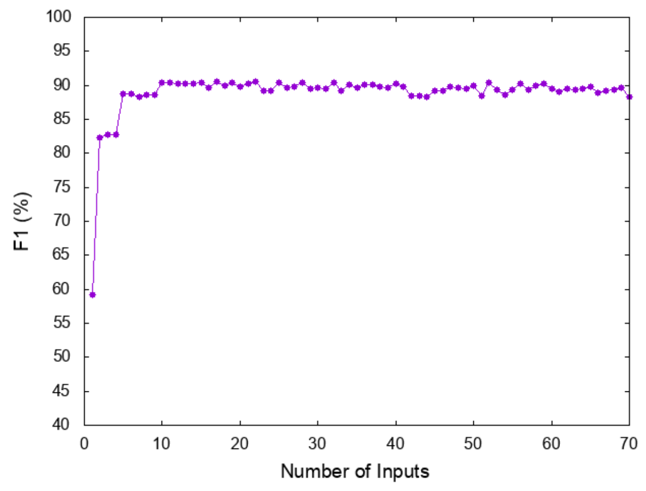


(c) Recall vs. Inputs

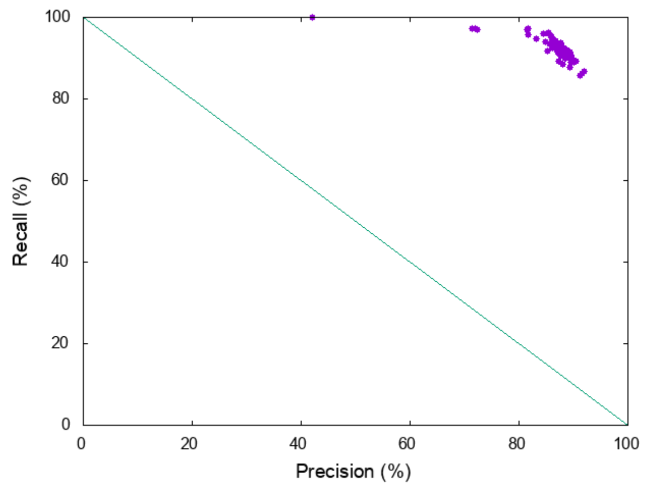


(d) Precision vs. Inputs

Fig. 3: AUC, Accuracy, Recall, and Precision vs. Number of Inputs



(a) F1 vs. Inputs



(b) Precision vs. Recall

Fig. 4: F1, Precision vs. Recall vs. Number of Inputs

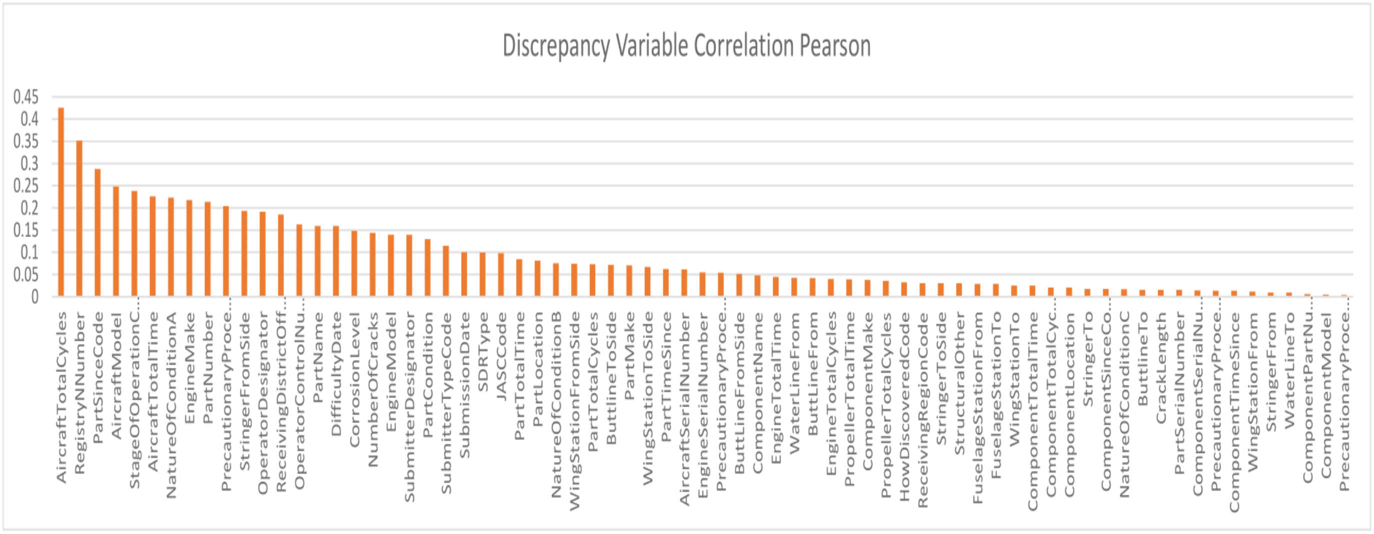


Fig. 5: Pearson correlation data bar chart

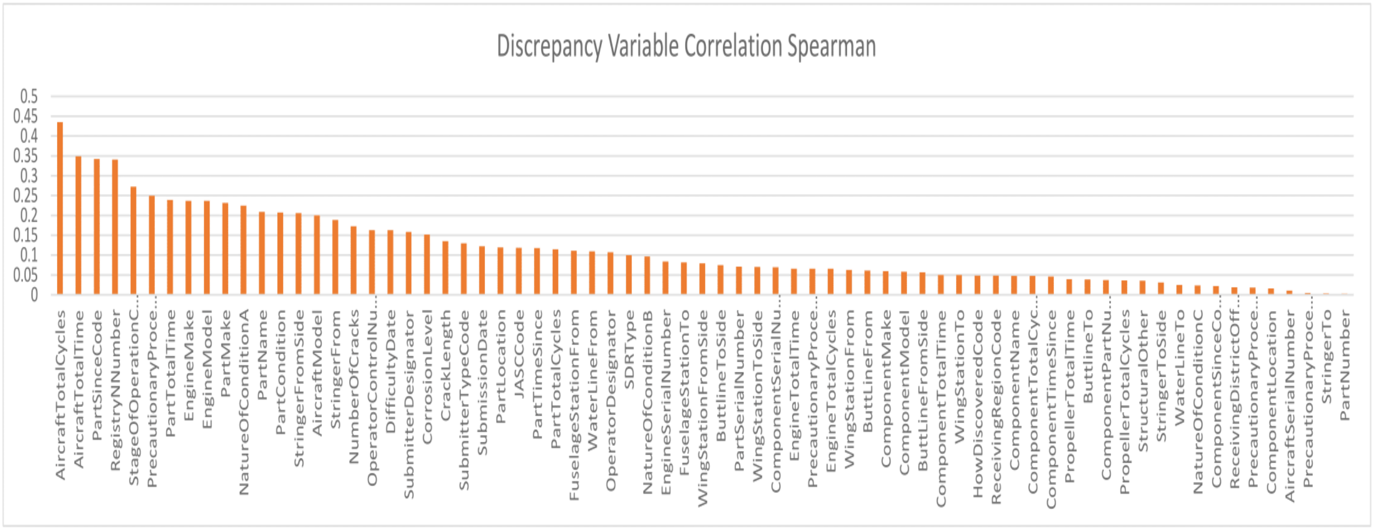


Fig. 6: Spearman correlation data bar chart

it means that there is no linear correlation between the two variables and the area is yellow. So, closer the number is to 1, the more association the variables have with each other.

Using both the Spearman and Pearson correlation tests, the variables with the highest correlation to Discrepancy were also determined. The Pearson test results shown in Figure 5 show a linear correlation coefficient of 0.15 or greater between the input variables AircraftTotalCycles, PartSinceCode, AircraftTotalTime, StringerFromSide. The Spearman test shown in Figure 6 shows a correlation coefficient of 0.15 or greater for the input variables AircraftTotalCycles, AircraftTotalTime, PartSinceCode, StringerFromSide, StringerFrom, and SubmitterDesignator. Analyzing the results of the combined tests reveals that AircraftTotalCycles, PartSinceCode, and AircraftTotalTime are the three variables with the highest degree of correlation. AircraftTotalCycles was shown to be the most significant variable with both Pearson and the Spearman test. These results could be implemented/applied in the future to determine which significant inputs could be used for training the neural networks.

VII. Conclusion

In this paper we propose a hybrid learning strategy strategy by integrating a neural network with decision tree. The hybrid algorithm is tested with the Federal Aviation Administration (FAA) data for Boeing 737. Several simulated experiments have been performed to test the efficacy of the proposed hybrid method. The method is tested with various network architectures, activation functions, and different hidden layers. The hybrid method is also verified by selecting the contributing input features, and the similar prediction results confirm that it successfully identified the redundant features.

To optimize our NN, three correlations were examined with regard to classifying Discrepancy; 1) the number of significant inputs to classification accuracy, 2) the total number of inputs to classification accuracy, and 3) the correlation of variables to all other variables. The first correlation showed that the number of inputs could be reduced from 72 to eleven significant inputs without a reduction in accuracy of the NN. The second correlation further validated this by showing that AUC, Accuracy, Recall and Precision stabilize with the eleven significant inputs and gradually deteriorate with the addition of new inputs. The third correlation further showed via heatmaps that only a small number of inputs have a high correlation with Discrepancy. Using these three correlations, we show that the significant input features can be identified and that the total number of features can be reduced without affecting the accuracy of the NN. The hybrid learning method can be tested in more case studies in the future. Moreover, the method can also be tested for transfer learning in different domains.

REFERENCES

[1] A. H. Vo, T. R. Van Vleet, R. R. Gupta, M. J. Liguori, and M. S. Rao, "An overview of machine learning and big data for drug toxicity evaluation," *Chemical Research in Toxicology*, vol. 33, no. 1, pp. 20–37, 2019.

[2] F. Samie, L. Bauer, and J. Henkel, "From cloud down to things: An overview of machine learning in internet of things," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4921–4934, 2019.

[3] I. H. Sarker, A. Kayes, S. Badsha, H. Alqahtani, P. Watters, and A. Ng, "Cybersecurity data science: an overview from machine learning perspective," *Journal of Big Data*, vol. 7, no. 1, pp. 1–29, 2020.

[4] T. Shon and J. Moon, "A hybrid machine learning approach to network anomaly detection," *Information Sciences*, vol. 177, no. 18, pp. 3799–3821, 2007.

[5] R. R. Bies, M. F. Muldoon, B. G. Pollock, S. Manuck, G. Smith, and M. E. Sale, "A genetic algorithm-based, hybrid machine learning approach to model selection," *Journal of pharmacokinetics and pharmacodynamics*, vol. 33, no. 2, pp. 195–221, 2006.

[6] S. Mohan, C. Thirumalai, and G. Srivastava, "Effective heart disease prediction using hybrid machine learning techniques," *IEEE Access*, vol. 7, pp. 81542–81554, 2019.

[7] C. Qi, H.-B. Ly, Q. Chen, T.-T. Le, V. M. Le, and B. T. Pham, "Flocculation-dewatering prediction of fine mineral tailings using a hybrid machine learning approach," *Chemosphere*, vol. 244, p. 125450, 2020.

[8] J. Qiu, Q. Wu, G. Ding, Y. Xu, and S. Feng, "A survey of machine learning for big data processing," *EURASIP Journal on Advances in Signal Processing*, vol. 2016, no. 1, p. 67, 2016.

[9] M. A. Fattah, "A hybrid machine learning model for multi-document summarization," *Applied intelligence*, vol. 40, no. 4, pp. 592–600, 2014.

[10] K. Polat and S. Güneş, "A novel hybrid intelligent method based on c4. 5 decision tree classifier and one-against-all approach for multi-class classification problems," *Expert Systems with Applications*, vol. 36, no. 2, pp. 1587–1592, 2009.

[11] H. Lu and X. Ma, "Hybrid decision tree-based machine learning models for short-term water quality prediction," *Chemosphere*, vol. 249, p. 126169, 2020.

[12] Z. Arabasadi, R. Alizadehsani, M. Roshanzamir, H. Moosaei, and A. A. Yarifard, "Computer aided decision making for heart disease detection using hybrid neural network-genetic algorithm," *Computer methods and programs in biomedicine*, vol. 141, pp. 19–26, 2017.

[13] B. T. Pham, D. T. Bui, I. Prakash, and M. Dholakia, "Hybrid integration of multilayer perceptron neural networks and machine learning ensembles for landslide susceptibility assessment at himalayan area (india) using gis," *Catena*, vol. 149, pp. 52–63, 2017.

[14] W. Chen, S. Chen, H. Zhang, and T. Wu, "A hybrid prediction model for type 2 diabetes using k-means and decision tree," in *2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, pp. 386–390, IEEE, 2017.

[15] C.-C. Lin, L. Shu, D.-J. Deng, T.-L. Yeh, Y.-H. Chen, and H.-L. Hsieh, "A mapreduce-based ensemble learning method with multiple classifier types and diversity for condition-based maintenance with concept drifts," *IEEE Cloud Computing*, vol. 4, no. 6, pp. 38–48, 2017.

[16] I. H. Laradji, M. Alshayeb, and L. Ghouti, "Software defect prediction using ensemble learning on selected features," *Information and Software Technology*, vol. 58, pp. 388–402, 2015.

[17] X. Yang, D. Lo, X. Xia, and J. Sun, "Tlel: A two-layer ensemble learning approach for just-in-time defect prediction," *Information and Software Technology*, vol. 87, pp. 206–220, 2017.

[18] Z. Li, K. Goebel, and D. Wu, "Degradation modeling and remaining useful life prediction of aircraft engines using ensemble learning," *Journal of Engineering for Gas Turbines and Power*, vol. 141, no. 4, 2019.

[19] J. Li, X. Li, and D. He, "A directed acyclic graph network combined with cnn and lstm for remaining useful life prediction," *IEEE Access*, vol. 7, pp. 75464–75475, 2019.

[20] J. Carson, K. Hollingsworth, R. Datta, and A. Segev, "Failing &! falling (f&! f): Learning to classify accidents and incidents in aircraft data," in *2019 IEEE International Conference on Big Data (Big Data)*, pp. 4357–4365, IEEE, 2019.

[21] B. Marcello, C. Davide, F. Marco, G. Roberto, M. Leonardo, and P. Luca, "An ensemble-learning model for failure rate prediction," *Procedia Manufacturing*, vol. 42, pp. 41–48, 2020.

[22] P. Zuvela, M. Lovric, A. Yousefian-Jazi, and J. J. Liu, "Ensemble learning approaches to data imbalance and competing objectives in design of an industrial machine vision system," *Industrial & Engineering Chemistry Research*, vol. 59, no. 10, pp. 4636–4645, 2020.

[23] R. Polikar, "Ensemble learning," in *Ensemble machine learning*, pp. 1–34, Springer, 2012.

[24] O. Sagi and L. Rokach, "Ensemble learning: A survey," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 8, no. 4, p. e1249, 2018.

[25] X. Dong, Z. Yu, W. Cao, Y. Shi, and Q. Ma, "A survey on ensemble learning," *Frontiers of Computer Science*, pp. 1–18, 2020.