Optimizing Artificial Neural Networks for Classifying Pulsar Neutron Stars

Kunho Kim ak26252@uga.edu

Abbie Thomas agt96353@uga.edu

Abstract

This paper compares four classification algorithms and three training sets for identifying pulsars from a set of pulsar candidates. Pulsars are neutron stars which offer a wealth of information for astronomers. The objective of this experiment is to compare four different pulsar classification algorithms: two ANNs, a Very Fast Decision Tree, and an ensemble algorithm We aim to reach a recall of at least 90% and a false positive rate (FPR) lower than 10%. The data set, HTRU2 contains 17,898 instances of pulsar candidates with eight input attributes and one target attribute for each instance. The data will be preprocessed and divided into three training sets: an unscaled, scaled, and proportionate (50% pulsars) sets. These algorithms will be analyzed for significance separately and compared based on the recall and FPR of the model. The results of this experiment concluded that the Very Fast Decision Tree using the Proportionate Training Set was the best model with a recall of 98.09% and FPR of 0.32% at its most efficient. The other models performed decently with recalls typically between 77-85%. Using the scaled data for the training set proved to be the best for the models overall. Future research on this topic includes optimizing the weight decay parameter, reducing the number of input attributes, and analyzing the effect of the proportionate training set on these models.

1. Introduction

A pulsar is a type of neutron star which rotates quickly, is highly magnetized, and whose radio emissions are detectable on Earth (Lorimer & Kramer, 2012). According to Lorimer and Kramer, pulsars provide astronomers with information about neutron star physics, general relativity, the Galactic gravitational potential and magnetic field, the interstellar medium, celestial mechanics, planetary physics, and cosmology. Zaven Arzoumanian of NASA's Goddard Space Flight Center said, “You can time the pulsations of pulsars distributed in many directions around a spacecraft to figure out where the vehicle is and navigate it anywhere.” Needless to say, these rare stars provide a wealth of knowledge when they are discovered. There are only around 2,000 known pulsars (Hille, 2017) – which means that detecting and identifying them is a difficult and important task.

The objective of this experiment is to create an optimized Artificial Neural Network (ANN) model for pulsar classification based on our research of previous work on the subject. We will judge the performance of the model based on recall and false positive rate (FPR). Our goal is to develop a model with recall higher than 90% and FPR lower than 10%.

The HTRU2 Data Set contains 17,898 pulsar candidates, which were collected during the High Time Resolution Universe Survey (Lyon, 2017). Of this data set, 1,639 are positively identified pulsars – about 9.157% of the data set – and 16,259 are not pulsars – about 90.843% of the data set. Each row represents a pulsar candidate and each candidate has eight attributes: mean of the integrated profile, standard deviation of the integrated profile, excess kurtosis of the integrated profile, skewness of the integrated profile, mean of the DM-SNR curve, standard deviation of the DM-SNR curve, excess kurtosis of the DM-SNR curve, a skewness of the DM-SNR curve. All eight attributes are continuous numerical variables. Each row also contains a target attribute, the class. The class is a nominal variable of either one or zero – one representing pulsars and zero representing non-pulsars.

We will produce four machine learning models: ANN using the logistic sigmoid activation function, ANN using the hyperbolic tangent function, Very Fast Decision Tree (VFDT), and an ensemble algorithm which combines ANN as the sub-learner and VFDT as the meta-learner. These three models will be trained using three sets of training/testing data: unscaled data, scaled data, and data containing 50% pulsar and 50% non-pulsar candidates. These models and testing data set were inspired by our literature review of researchers who also produced pulsar candidate classification algorithms.

The resulting models were compared by their recall and FPR scores. This decision was made in order to optimize the number of pulsars we were correctly identifying as positive and pay less attention to the number of non-pulsars. The reasoning for this is described in detail in Section 2. We found that the VFDT trained with the Proportionate Training Set had the highest recall score of 98.06% and lowest FPR score of 0.32%. Though the Proportionate Training Set performed well for training the VFDT, it performed the worst of all the methods for the remaining three algorithms. The Scaled Training Set was the most consistently productive at training our models while the VFDT was the most consistently productive machine learning algorithm.

Our next steps focus on reducing overfitting of our models. We believe that optimizing the weight decay parameter and reducing the number of input variables have the possibility to improve generalization of our algorithms. We are also interested in investigating the reason for ANNs performing poorly with the proportionate training set while the VFDT performed so well with it.

2. Literature Review

Due to their rarity, finding and positively identifying pulsars is a tedious task. In order to classify a candidate as a pulsar, it must be inspected manually by a researcher. Because there are a large number of pulsar candidates, the process of examining each one by eye is time consuming. It is, therefore, necessary to cut down the number of candidates. This required pruning has inspired many data scientists to develop machine learning algorithms to do so. Upon reviewing the algorithms developed for classifying pulsar candidates, artificial neural networks were quite common.

ANNs are composed of an input layer, hidden layer(s), and an output layer - each of which is made of artificial neurons (Dickson, 2019). The output of a neuron is based on an activation function – in this experiment, we tested both the logistic sigmoid and hyperbolic tangent activation functions. Bates et al. (2012) developed an ANN for classifying pulsar candidates, using the HTRU dataset. Key features of this ANN include the use of the logistic sigmoid activation function, 22 neurons for the input and hidden layers, and 2 for the output layer. The logistic sigmoid function is useful due to its ability to approximate both linear and non-linear behavior; both of which are present in the classification of pulsars. Because this model was developed during the early stages of data collection for the HTRU dataset, a small number of candidates were chosen for the training set, including 70 pulsars and 200 non-pulsar candidates (25.92% of the training set composed of pulsars). The ANN developed by Bates et al. (2012) was able to correctly identify 85% of 510 candidates as pulsars. They acknowledge that these results were not as high as expected and cite some improvements for future experimentation. They discuss first of all, having a more representative sample for the training set – it is possible that the small number of actual pulsars in the training set contributed to a less than satisfactory accuracy rate. Next, the authors point out that there are two types of pulsars: millisecond and normal pulsars. The ANN created was able to correctly identify both types of pulsars, however Bates et al. (2012) discussed the concept of creating two separate algorithms for the two types of pulsars to increase accuracy. This paper contributed to our interest in optimizing an ANN for classifying pulsar candidates. They suggest that any activation function can be used for the pulsar classification problem.

$$Recall = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \quad (1)$$

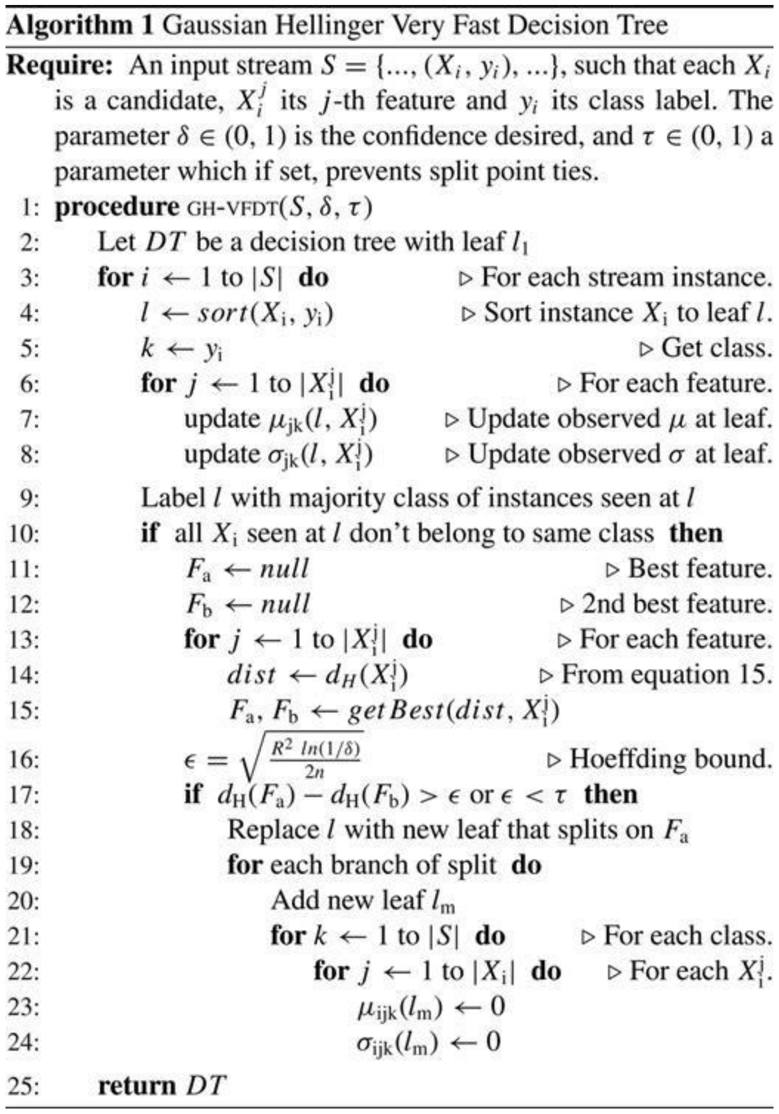
$$False\ Positive\ Rate = \frac{\text{False Positives}}{\text{True Negatives} + \text{False Positives}} \quad (2)$$

Figure 1: Formulas for calculating the Recall (1) and FPR (2) statistics which will be used to assess the performance of each of our models. The variables on the right side of the equation can be generated using a confusion matrix.

SPINN (Straightforward Pulsar Identification using Neural Networks) is a machine learning model developed by Morello et al. (2014) to reduce the number of pulsar candidates that must be visually inspected. This algorithm produced a 100% recall for pulsars with a 64% false positive rate using the HTRU dataset. This ANN used the hyperbolic tangent activation function and a two-layered network with eight neurons in the hidden layer and one neuron in the output layer. Though the hyperbolic tangent function is capable of handling non-linear functions, it outputs a number between [-1, 1] while logistic sigmoid outputs a number between [0, 1]. Both Morello et al. (2014) and Bates et al. (2012) do not go into great detail explaining why the selected activation function was chosen. Because of this discrepancy, we decided to examine the difference between the two functions in our ANN model. This model oversampled pulsars for the training set in a 4:1 ratio of non-pulsars to pulsars (Morello et al., 2014). This ratio is similar to the training set used by Bates et al. (2012), around 20-25%. Because neither ANN model used a higher ratio of non-pulsar to pulsar candidates, we decided to train our algorithm using both the initial 10% representation as well as 50% representation. SPINN was optimized using 5-fold cross validation to find the best performing algorithm. Finally, Morello et al. (2014) discusses the importance of regularization – in their case using L2 weight decay. Since the model weights are being determined by the training set, it is possible that they are being overfitted to the training data. If this were the case, the ANN would not generalize well to other pulsar candidates, since they are different from the pulsar candidates used for training. Regularization is a way to mediate this risk by making sure that one variable weight does not get unreasonably higher than the others. Finding the optimal value for the weight decay parameter, is therefore important.

Morello et al. (2014) made a convincing argument for the relevant metrics used for evaluating the performance of the model. When testing machine learning models, it is common to rely heavily on the accuracy of the model. Pulsar candidate identification, however, is not a typical problem. The machine learning tools we are creating are used to narrow down a large number of pulsar candidates into a smaller pool of candidates. This must be done because each candidate must be visually inspected in order to positively identify it as a pulsar. Therefore, it is more important that the machine learning algorithms have a high recall and low false positive rate (FPR). When the score of an algorithm has higher recall, it means that more actual pulsars are being identified. When the FPR is lower, it means that there are less non-pulsars being identified as pulsars – we want this rate to be low so that the researchers manually inspecting pulsar candidates waste less time. The formulas for recall and FPR can be found in Figure 1.

Similar to Bates et al. (2012), Eatough et al. (2010) also used ANN models in order to classify pulsar candidates. They mainly focused on “8:8:2” ANN models. This means that there are 8 features in the input layer, 8 nodes in the hidden layer, and 2 in the output layer (pulsar or non-pulsar classification). Since this seemed to be a very simple yet effective ANN, we decided



Algorithm 1 Gaussian Hellinger Very Fast Decision Tree

```
Require: An input stream $S = \{..., (X_i, y_i), ...\}$, such that each $X_i$
    is a candidate, $X_i^j$ its $j$-th feature and $y_i$ its class label. The
    parameter $\delta \in (0, 1)$ is the confidence desired, and $\tau \in (0, 1)$ a
    parameter which if set, prevents split point ties.
 1: procedure GH-VFDT($S, \delta, \tau$)
 2:     Let $DT$ be a decision tree with leaf $l_1$
 3:     for $i \leftarrow 1$ to $|S|$ do                    ▷ For each stream instance.
 4:         $l \leftarrow sort(X_i, y_i)$                   ▷ Sort instance $X_i$ to leaf $l$.
 5:         $k \leftarrow y_i$                              ▷ Get class.
 6:         for $j \leftarrow 1$ to $|X_i^j|$ do            ▷ For each feature.
 7:             update $\mu_{jk}(l, X_i^j)$                 ▷ Update observed $\mu$ at leaf.
 8:             update $\sigma_{jk}(l, X_i^j)$              ▷ Update observed $\sigma$ at leaf.
 9:         Label $l$ with majority class of instances seen at $l$
10:         if all $X_i$ seen at $l$ don't belong to same class then
11:             $F_a \leftarrow null$                       ▷ Best feature.
12:             $F_b \leftarrow null$                       ▷ 2nd best feature.
13:             for $j \leftarrow 1$ to $|X_i^j|$ do        ▷ For each feature.
14:                 $dist \leftarrow d_H(X_i^j)$            ▷ From equation 15.
15:                 $F_a, F_b \leftarrow getBest(dist, X_i^j)$
16:             $\epsilon = \sqrt{\frac{R^2 \, ln(1/\delta)}{2n}}$   ▷ Hoeffding bound.
17:             if $d_H(F_a) - d_H(F_b) > \epsilon$ or $\epsilon < \tau$ then
18:                 Replace $l$ with new leaf that splits on $F_a$
19:                 for each branch of split do
20:                     Add new leaf $l_m$
21:                     for $k \leftarrow 1$ to $|S|$ do    ▷ For each class.
22:                         for $j \leftarrow 1$ to $|X_i|$ do   ▷ For each $X_i^j$.
23:                             $\mu_{ijk}(l_m) \leftarrow 0$
24:                             $\sigma_{ijk}(l_m) \leftarrow 0$
25:     return $DT$
```

Figure 2: Gaussian Hellinger Very Fast Decision Tree algorithm used by Lyon et al. (2016) which the version of our VFDT was based on.

to model our ANN after this. Additionally, it is similar to the network structure used by Morello et al (2014).

Very fast decision trees (VFDT) are decision trees that are designed to maximize performance in classification problems in which the dataset is heavily imbalanced towards one class. In our case, the pulsar dataset has many more 0's (non-pulsars) than 1's (pulsars). Additionally, very fast decision trees allow for the existing tree to be updated using new data without having to go back to previous instances. This is extremely helpful for situations in which the dataset is still being completed or if the dataset is too large (Lyon et al., 2016). A standard supervised model isn't enough for this classification problem. Since new data is being captured every day regarding pulsar classification, we need a more dynamic model such as the VFDT. Lyon et al. (2016) used a variant of the VFDT called the Gaussian Hellinger Very Fast Decision Tree. Although we could not find a python version of this model, we did find a generic VFDT model that the Gaussian Hellinger VFDT was based on. This generic VFDT was developed by Hulten, Spencer, and Domingos (2001). The algorithm for the Gaussian Hellinger Very Fast Decision Tree is shown in Figure 2. More specifically, according to Lyon et al. (2016), usage of VFDT's has shown to improve recall rates for pulsar data, which we will further discuss in our results section.

3. Data – Preprocessing

As discussed in the introduction, we chose to train each algorithm on three different training sets: unscaled data (Unscaled Training Set), scaled data (Scaled Training Set), and scaled data containing 50% pulsar and 50% non-pulsar candidates (Proportionate Training Set). This data set is quite unbalanced, with less than 10% of the data being the positive target in class one and more than 90% of the data being the negative target in class zero. The Unscaled and Scaled Training Sets reflects this ratio. Additionally, the original data set doesn't include column labels, which caused problems in converting the data to a Pandas data frame. As a solution for this, we added the correct column labels as the header row. This step made it easier to access the columns of the data frames in the code in later stages of our experiment.

We decided to include all of the eight features in the data set. There were several motivations for this, but the main reasons were that including all eight features gave us the best results without sacrificing a large amount of time. Using scikit-learn’s recursive feature elimination (RFE), we determined that each input attribute was performing equally, with each having an RFE ranking of one. More about this will be discussed in the future work and experiments section, but we currently believe that all eight of these features hold significance in predicting pulsars.

We used scikit-learn’s StandardScaler to scale the data frames containing the desired input data. The StandardScaler was used to scale our data so that it had a variance of one and mean of zero. This step was taken because Morello et al. (2014) describe feature scaling as an important preprocessing step in their model. Since SPINN had such a high success rate, we decided to compare the results of scaled versus non-scaled input variables in our models. The scaled data was used for the Scaled Training Set and Proportionate Training Set.

In order to create the Proportionate Training Set, we split the current data set into two separate data sets: the rows of class zero and the rows of class one (method two). Since there are significantly more instances of class zero than class one, the class zero data set was further reduced by choosing a random sample. This random sample's size is the same size as the class one data set so that the two can be combined without error. These two data sets were then each split using the 80/20 method mentioned earlier. The 80% splits were appended together to form the modified testing set and the 20% splits were appended to form the modified training set. The input data was then scaled using the StandardScaler, as previously explained. The result of this preprocessing step was a training and testing set that had an even distribution of class one and class zero (50% pulsars, 50% non-pulsars) – the Proportionate Training Set. We speculated that partitioning the data set with equal instances of both classes would mitigate the impact of having an unbalanced data set based on our literature review in Section 2. Bates et al. (2012) listed an underrepresentation of positively identified pulsars as a possible cause for lower performance of their model. The results of these proportionately partitioned data sets will be discussed further in the analysis section

To train the models, we did a standard 80% training, 20% testing split of the Training Sets for each model. More on this will be explained in the experiments section. After all preprocessing steps were done, we were left with three sets of training data to run: Unscaled, Scaled, and Proportionate Training Sets.

4. Experiments

As discussed in Section 1: Introduction, we decided to develop four different machine learning algorithms and train each model using three different training sets. These models included ANN using the logistic sigmoid activation function, ANN using the hyperbolic tangent function, Very Fast Decision Tree (VFDT), and an ensemble algorithm which combines ANN as the sub-learner and VFDT as the meta-learner. For each algorithm, we trained it using the three data sets: Unscaled, Scaled, and Proportionate Training Sets.

Using Jupyter Notebook, the data was read in as a Pandas data frame from a csv file. Because the original data set did not include column labels in the csv file, the first instance in the data set was used as each column label. We created practical column names and added the first instance to the data frame. Next, we saved the first eight columns – the input attributes – into an array and the last column – the target attributes – into an array. We created the three Training Sets, as described in Section 3: Preprocessing. The data was split so that 20% of the data was used for training and 80% was used for testing using scikit-learn's train_test_split from model_selection.

This resulted in four NumPy ndarrays: input training, input testing, target training, and target testing for each Training Set.

We built the ANN model using Keras by following Sachit Tanwar's guide on "Building our first neural network in Keras." In order to avoid overfitting, we decided to use stratified k-fold cross-validation for all of our ANN experiments. In our case k=10. We were able to accomplish this by using scikit-learn's StratifiedKFold. Furthermore, since we were building an ANN we chose to use a sequential model from Keras. In this model, we created an input layer with 8 nodes, a hidden layer with 8 nodes, and an output layer with 2 nodes. This was strictly based on Eatough et al. (2010) and their work on pulsar classification using "8:8:2" ANN's. In Keras, we were able to select our activation function for each layer. Since our problem is a binary classification problem, the activation function on the output layer had to be the sigmoid function. Since we were comparing ANN models with the logistic sigmoid activation function and ANN models with the hyperbolic tanh function, for each model we changed the activation function of the input and hidden layers. This was relatively simple to do in Keras – when adding a layer we specified the activation function by either writing "activation='sigmoid'" or "activation='tanh'." This would automatically apply that specific activation function to the layer and we could easily compare the performance.

For the VFDT, we used a python implementation that we were able to find on a public-use GitHub repository. We had to do some extra preprocessing on our data to be able to use the VFDT code correctly. This included splitting our dataset manually instead of using scikit-learn. By doing this, we were able to have our tree learn continuously over the split datasets instead of all at once. The link to access the code will be located in our references section.

	Predicted: 1	Predicted: 0
Actual: 1	TP = 310	FP = 4
Actual: 0	FN = 1	TN = 341

Very Fast Decision Tree using Proportionate Training Set

	Predicted: 1	Predicted: 0
Actual: 1	TP = 136	FP = 10
Actual: 0	FN = 28	TN = 1616

ANN using hyperbolic tangent activation function and Scaled Training Data

	Predicted: 1	Predicted: 0
Actual: 1	TP = 141	FP = 14
Actual: 0	FN = 23	TN = 1612

ANN using logistic sigmoid activation function and Scaled Training Data

Figure 3: The three confusion matrices for the top performing algorithms: VFDT using Proportionate Training Set, ANN using hyperbolic tangent activation function and Scaled Training Data, and ANN using logistic sigmoid activation function and Scaled Training Data. The false positive rates for all these models are very low – this was the desired goal.

We decided to combine these two promising models that had already been used in previous works into one ensemble model (aka stacked model). To our knowledge, this is a novel approach. Our development of the ANN-VFDT ensemble model was significantly influenced by Jason Brownlee's guide on "Stacking Ensemble for Deep Learning Neural Networks in Python." In order to create our ensemble model, we need to have a sub-learner, which is the "first layer" of the model. In this layer, the learner takes in training input and learns to make predictions from this data. We decided to choose the ANN to be our sub-learner. In addition to a sub-learner, the “second layer” of the model is the meta-learner. In our case, the VFDT was the meta-learner. In future works it would be interesting to see if there is any significant difference in performance if the VFDT was the sub-learner and the ANN was the meta-learner.

As we mentioned earlier, all of these different models were tested with several different versions of the dataset: Unscaled, Scaled, and Proportionate Training Sets. The Proportionate Training Set was used in our previous attempt from Mini Project One to combat the unbalanced nature of the pulsar dataset. To briefly explain, we randomly chose the same number of non-pulsar candidates as pulsar candidates and put them in a separate Pandas data frame. This gave us a smaller, yet balanced dataset. This was then also scaled using the StandardScaler.

For all of the models, we calculated Recall, Precision, False Positive Rate, True Positive Rate, and Accuracy. This was done using scikit-learn’s various functions regarding metrics and scores. Confusion matrices were also created using the performance data of each model. However, we weren’t able to use scikit-learn’s confusion_matrix to create confusion matrices for the ANN models. Thus, we used the true positive, true negative, false positive, and false negative values in order to create the confusion matrix. The performance results are discussed in the section below.

5. Analysis

As discussed in Section 2: Literature Review, we decided to compare the recall and false positive rate (FPR) of four different machine learning algorithms trained on three different training sets. Though both recall and FPR will be considered when assessing the performance of the algorithms, recall will be considered more important due to the fact that a lower recall means

	ANN using logistic sigmoid activation function	ANN using hyperbolic tangent activation function	Very Fast Decision Tree (VFDT)	VFDT as meta-learner and ANN as sub-learner
Unscaled Training Set	**Recall: 79.86%** Precision: 94.68% **FPR: 0.6%** TPR: 81.15% Accuracy: 97.73%	**Recall: 78.31%** Precision: 94.51% **FPR: 0.62%** TPR: 79.56% Accuracy: 97.56%	**Recall: 81.08%** Precision: 92.90% **FPR: 7.10%** TPR: 98.08% Accuracy: 97.66%	**Recall: 77.56%** Precision: 93.08% **FPR: 6.92%** TPR: 97.89% Accuracy: 97.54%
Scaled Training Set	**Recall: 85.09%** Precision: 93.06% **FPR: 0.74%** TPR: 87.80% Accuracy: 98.21%	**Recall: 83.47%** Precision: 95.08% **FPR: 0.92%** TPR: 85.89% Accuracy: 97.88%	**Recall: 82.40%** Precision: 90.30% **FPR: 9.70%** TPR: 98.28% Accuracy: 97.62%	**Recall: 80.18%** Precision: 94.60% **FPR: 5.40%** TPR: 98.03% Accuracy: 97.77%
Proportionate Training Set	**Recall: 64.33%** Precision: 64.55% **FPR: 0.00%** TPR: 99.70% Accuracy: 99.85%	**Recall: 64.47%** Precision: 64.55% **FPR: 0.00%** TPR: 99.88% Accuracy: 99.94%	**Recall: 98.09%** Precision: 99.68% **FPR: 0.32%** TPR: 98.27% Accuracy: 99.09%	**Recall: 41.07%** Precision: 65.82% **FPR: 34.18%** TPR: 60.87% Accuracy: 62.30%

Table 1: The collection of results from each experiment including four different machine learning models and three different training sets. Each experiment had five results: recall, precision, FPR, TPR, and accuracy. The important values have been bolded. Additionally, the top three performing algorithms have thicker borders while the bottom three have gray shading.

losing possible rare pulsar candidates. The results of our experiments, as presented in Table 1, show the top three performing algorithms (thick border) and the bottom three performing algorithms (light grey shading). The confusion matrices for the top three performing algorithms can be found in Figure 3.

First, the top performing model was the VFDT trained on the Proportionate Training Set with 98% recall and 0.32% FPR. This model had the highest recall value by far – the second highest being 85.09%. Though the results are excellent, we must consider the possibility that this algorithm overfitted the training data. Second was the ANN using logistic sigmoid activation function trained on the Scaled Training Set and third was the ANN using hyperbolic tangent activation function trained on the Scaled Training Set (See Table 1 for results). These two models performed very similarly with a difference in recall value of only 1.62% and a difference in FPR of only 0.18%. The difference between the logistic sigmoid activation function and the hyperbolic tangent function produce extremely similar results for the Unscaled and Proportionate Training Sets as well. This is an interesting observation for our experiments. In Section 2: Literature Review, we discussed that Morello et al. (2014) and Bates et al. (2012) did not explain the reasoning for which activation function they chose. Based on these results, we can hypothesize that the two activation functions work so similarly that which function was used was an unimportant factor to the overall performance of the ANN model.

Overall, we concluded that the VFDT model performed the best. If we were to assume that the model trained on the Proportionate Training Set were overfitted, the VFDT trained on the Scaled Training Set performed almost as well as the ANN models. The ensemble algorithm had the lowest recall and highest FPR scores of all the models.

We can also conclude that training using the Scaled Training Set was the most effective in producing models. Unfortunately, the Proportionate Training Set did not perform well on any model aside from the VFDT. The bottom three performing algorithms have one thing in common: The Proportionate Training Set. This set contained 50% pulsar candidates – considering that the HTRU2 dataset only has a proportion of 10% pulsars, this oversampling might have damaged the ANN by confusing the frequency of pulsars. The dramatic difference in performance between the ANN models and the VFDT model when trained on the Proportionate Training Set poses interesting questions regarding how the model is trained. This will be discussed further in Section 6: Future Work.

6. Future Work

If we decide to continue with this project in the future, there are a few things that may help us optimize our models further. In order to generalize our models, we could find an optimal value for the weight decay parameter. In our experiments, we decided to use the standard alpha value of 0.0001. Using this weight decay parameter, we were able to get good results, however, it would be interesting to try to find the best alpha value for pulsar classification. Regularization using L2 weight decay was mentioned by Morello et al. (2014) as an important step for reducing overfitting. Though we included the parameter, further research is required to find the optimal value of alpha.

We are also interested in further investigating the parameters of our ANN to simplify it. It is common that a less complex model will generalize better to other testing sets. One way that Batch et al. (2012) suggests doing this is by reducing the number of input attributes to the most essential and useful attributes. By doing this, we could possibly reduce the impact that outliers in the training data set could have and therefore reduce overfitting.

Additionally, although the results weren't better than the solo models, the ANN-VFDT ensemble model still showed promising results. It's possible that with a simpler ANN model and better configured VFDT that this ensemble model could outperform the other models. However, more work, especially in truly understanding how each of the model works (ANN, VFDT, ensemble models) is needed to fully unlock the capabilities of each.

Lastly, we are interested in figuring out why the ANN models trained using the Proportionate Training Set performed much worse than the VFDT trained on the same data set. This difference in performance could have something to do with how the model is trained. We also want to look at how the ANN models would perform if the training set with oversampled pulsars had a lower proportion of them. For example, how would the ANNs perform when pulsars represented 30%, 25%, or 15% of the data set as opposed to 50%?

7. Conclusion

In the end, the Very Fast Decision Tree (VFDT) model, trained using the Proportionate Training Set performed the best 98.09% recall and 0.32% false positive rate. Many other models performed similarly well at around 80 – 85% recall, which was promising to see. The best overall performing model was the VFDT and the best overall performing training set was Scaled Training Set. With our current work and potential future endeavors in this project, we hope that our findings can help astronomers in using data and machine learning to identify and classify galactic objects, such as pulsars.

8. Extension of Previous Work (Mini Project One)

This project was an extension of our first project – Mini Project One. Both projects dealt with the classification of pulsar candidates from the HTRU2 data set. The difference in our project lay in the algorithms used for classification and the criteria for evaluating models. Based on our literature review, we found that many data scientists were using ANNs for this particular classification problem. Along with this, the VFDT used by Lyon et al. (2016) seemed like it had good potential. We decided to build two ANNs, a VFDT, and an ensemble algorithm combining the two. Additionally, we evaluated the performance of the models based on recall and false positive rate in this project – as opposed to accuracy, which we used in Mini Project One. The reason for this was described in detail in Sections 1 and 2. In the end, though we used the same dataset, the algorithms and analysis were entirely different based on the literature review.

References

Bates, S. D., Bailes, M., Barsdell, B. R., Bhat, N. D., Burgay, M., Burke-Spolaor, S., … van Straten, W. (2012). The High Time Resolution Universe Pulsar Survey — VI. An artificial neural network and timing of 75 pulsars. Monthly Notices of the Royal Astronomical Society, 427(2), 1052–1065. https://doi.org/10.1111/j.1365-2966.2012.22042.x

Brownlee, J. (2018, December 21). Stacking Ensemble for Deep Learning Neural Networks in Python. Machine Learning Mastery. https://machinelearningmastery.com/stacking-ensemble-for-deep-learning-neural-networks/.

Dickson, B. (2019, August 5). What are artificial neural networks (ANN)? TechTalks. https://bdtechtalks.com/2019/08/05/what-is-artificial-neural-network-ann/.

Dixit, S. (2018, April 17). Building your own Artificial Neural Network from scratch on Churn Modeling Dataset using Keras in Python. Towards Data Science. https://towardsdatascience.com/building-your-own-artificial-neural-network-from-scratch-on-churn-modeling-dataset-using-keras-in-690782f7d051.

Eatough, R. P., Molkenthin, N., Kramer, M., Noutsos, A., Keith, M. J., Stappers, B. W., & Lyne, A. G. (2010). Selection of radio pulsar candidates using artificial neural networks. Monthly Notices of the Royal Astronomical Society, 407(4), 2443–2450. https://doi.org/10.1111/j.1365-2966.2010.17082.x

Lorimer, D. R., & Kramer, M. (2012). Handbook of pulsar astronomy. Cambridge: University Press.

Lyon, R. J., Cooper S., Brooke J. M., Knowles J. D., & Stappers B. W. (2016, March 16). Fifty years of pulsar candidate selection: from simple filters to a new principled real-time classification approach, Monthly Notices of the Royal Astronomical Society, Volume 459, Issue 1, 11 June 2016, Pages 1104–1123, https://doi.org/10.1093/mnras/stw656

Lyon, R. J. (2017) HTRU2, DOI: 10.6084/m9.figshare.3080389.v1.

Hille, K. (2017, August 1). NASA Continues to Study Pulsars, 50 Years After Their Chance Discovery. NASA. https://www.nasa.gov/feature/goddard/2017/nasa-continues-to-study-pulsars-50-years-after-their-chance-discovery.

Hulten, G. Spencer, L. Domingos, P. (2001). Proc. of 7th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining

Morello, V., Barr, E. D., Bailes, M., Flynn, C. M., Keane, E. F., & van Straten, W. (2014). SPINN: a straightforward machine learning solution to the pulsar candidate selection problem. Monthly Notices of the Royal Astronomical Society, 443(2), 1651–1662. https://doi.org/10.1093/mnras/stu1188