***PROJECT REPORT***

**TITLE-** Predicting Blood Donations

**Team –** Krishna Chaitanya Dodda (krd150230), Aastha Dixit (axd155030)

1. **Introduction**

Blood Donations are very crucial in saving life’s and especially most important in arenas like war zones, drought prone regions, rural regions and hinterlands. But to drive people into donating blood, it would help if there was some set criteria that could tell us which people could be targeted. Using resources to target those people who are more likely to donate blood can deliver more optimized collections of blood banks. The crucial question is whether the prowess of machine learning can be used to develop a prediction model to identify people who are more likely to donate blood in the future. It is in this quest that DataDriven has hosted an online competition on a given dataset to predict donations.

This project is based on this online data science competition of predicting whether a person is likely to donate blood. This is a classification problem. This project aims to develop predictive models using latest libraries (Spark Mlib 2.0) and technologies (Spark cluster). The project aims to achieve two key objectives:

* To successfully demonstrate that a predictive model with reasonable accuracy can be developed from the dataset. In doing, so we strived to achieve the highest position possible in the competition leaderboard.
* Our main objective is to do a comparative analysis of existing libraries and packages of R with that of Spark Mlib in developing models for the competition.

1. **Dataset**
   1. **Source**

The dataset is drawn from an online datascience competition- <https://www.drivendata.org/competitions/2/>.The data was obtained from a mobile blood donation vehicle in Taiwan. The Blood Transfusion Service Center vehicle drove to different universities and collects blood as part of a blood drive. The data is available in two comma separated forms i.e – test.csv and train.csv.

* 1. **Description**

Data Set Characteristics: Multivariate

Number of Instances: 748(train and test combined)

Attribute Characteristics: Real

Number of Attributes: 5

Attribute Information:

-R (Recency - months since last donation),

-F (Frequency - total number of donation),

-M (Monetary - total blood donated in c.c.),

-T (Time - months since first donation), and

-a binary variable representing whether he/she donated blood in March 2007 (1 stand for donating blood; 0 stands for not donating blood).

1. **Process Flow**

The data set is a classification problem. Hence we plan to use the following powerful techniques.

* Random Forest Classifier
* Boosting technique – Gradient Boosted Trees
* Multilayer Perceptron

The key challenge is to choose the appropriate parameters that improve the accuracy. To do this, we have divided our pipeline into the following stages. The pipeline is followed both for R packages and for Spark Package.

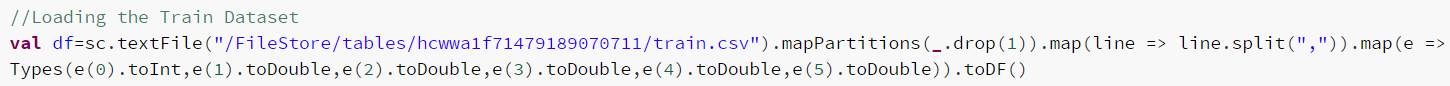
Data preprocessing is the first stage where the data is formatted, tuned and scaled to the appropriate format before being trained. Model Training phase trains the model on the train dataset. We develop three classifier models in this phase. In the final phase of Comparative Model Evaluation, we compare each of these models and try to select the best models based on various parameters.

1. **Data Preprocessing**

This is the phase where raw data is parsed and converted to meaningful representation and format for the models to process.

* 1. **Data Read**

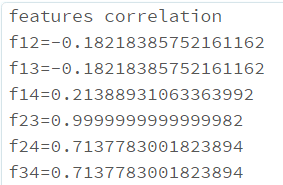
In this phase, we are reading the train.csv file.



This is done in Scala by first reading the csv file and then storing it in the form of an RDD. This RDD is then split linewise and types are assigned to each of the columns. Lastly, it is converted a dataframe since we are using the latest Mlib library (2.0) which operates on dataframes. A similar process is done in our R code where the csv file is read and columns are extracted and labels factored.

* 1. **Feature Selection / Dimensionality Reduction**

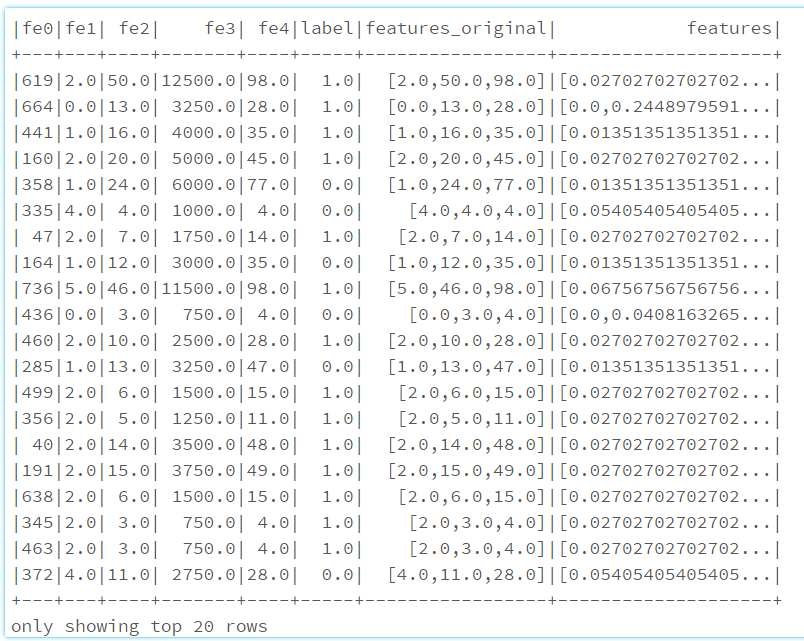
This is a crucial process in developing a prediction model. The key is to identify the correct features to be trained and if possible remove any redundant features. In our case, we did this in Scala by calculating Spearman correlation amongst the feature vectors. The output snipped is as follows.



F12 in the figure represents correlation between features represented by column 1 and 2. We have 4 features and hence 6 such correlations are possible. From the analysis , it is clear that features represented by column 2 and 3 are very highly correlated(0.99) . This essentially means that we can eliminate one of them as they are redundant. This is also logical since column 2 is the frequency of blood donation while column 3 is the actually quantity of blood donated. Naturally these two terms are directly proportional to each other and hence are highly correlated. Thus we can eliminate the column 3. Such a dimensional reduction helps us firstly by improving speed and accuracy of the algorithm and also saves us from the trouble of scaling the column 3 since. Hence, we select column 1, 2 and 4 as our features and drop column 3.

* 1. **Feature assembling and Scaling.**

This is the final phase of Data Preprocessing. In this phase, we ensure that the data is in the right input format as required by the libraries. The spark Mllib Libraries require all the features to be combined in the form of a feature vector called “features” and the class label labelled as “label”. Additionally, we have scaled the dataset to improve the accuracy of our classifiers. The scaling has been done by using an in built library of Spark (MinMaxScaler). However, scaling only increases the accuracy marginally in our case and can be done away with. The Final Data is in the following format.

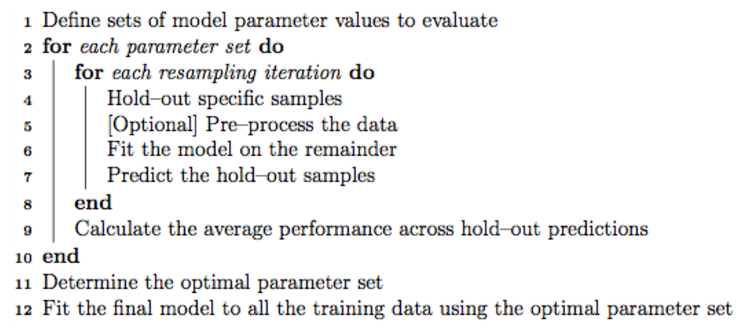


1. **Model training**

We train 3 models in total. Each of these Models we train using two frameworks – R and Spark.

Each of these models is trained using multiple combination of parameters and we strive to choose the best possible parameters for a model. This is done in two ways- cross validation and normal train test validation. Cross validation is not feasible in spark Mlib 2.0 since it has unresolved issues. So we used the train – test validation methodology to choose best set of parameters. In case of R , we performed a 10 fold cross validation using the CARET package available in R.

The following pseudocode explain the algorithm used to build the R model. The algorithm is implemented by the in built ‘train’ function of “CARET” package of R.



Source - https://topepo.github.io/caret/

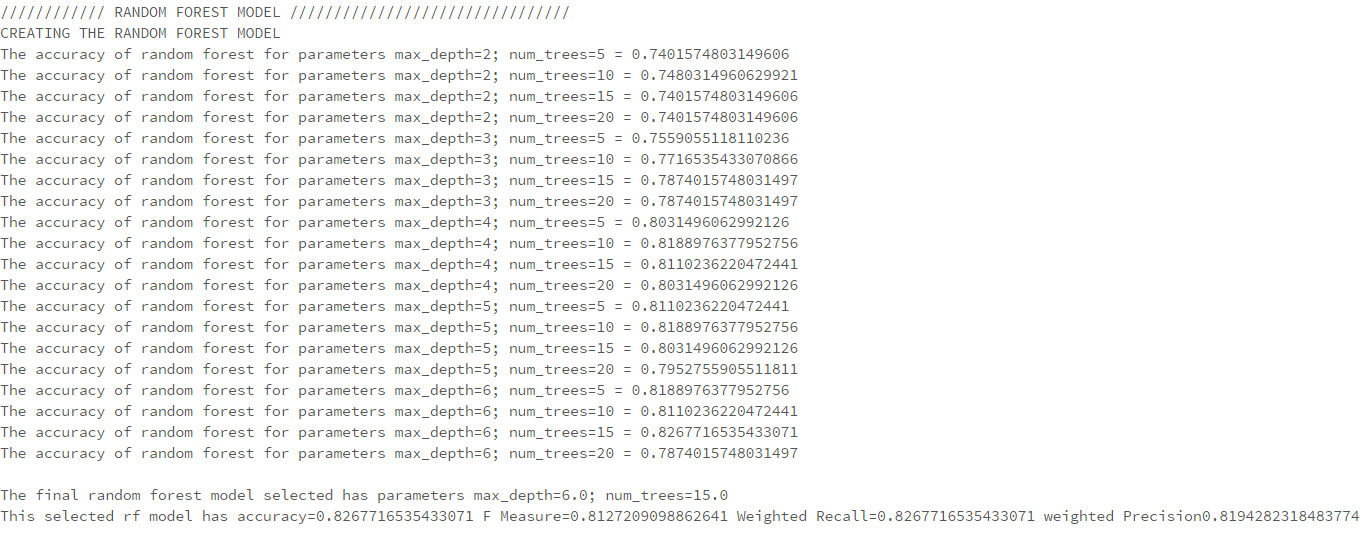
The above image describes the algorithm that is employed. For training and building the model on the dataset. Prior to this is the preprocessing phase where the data is pre processed. Steps 1-2 in the figure are represented by a built in function called – ‘train’. The above algorithm is performed for choosing the best model with the best suited parameters. First, we define the vector of parameters that we want to test the model for. Then, for each parameter, a 10 fold cross validation is performed and average accuracy, average, precision etc are calculated. The parameter is chosen and the model is built which has the highest accuracy. The following code snippet specifies the number of validation to perform to 10.



* 1. **Random Forest Classifier**

This is an Ensemble Classifier that combines parallel weak classifiers to create a resultant optimum classifier. Random Forest is one of the most successful algorithm for classification problems. We chose this classifier as it can eliminate overfitting and also deliver good accuracy by combining multiple classifiers. It employs Bootstrapping to create multiple subsets of the sample and trains classifiers on each of these classifiers before aggregating them.

* + 1. **Spark**

Spark Mllib 2.0 has a Random Forest classifier library. The two main parameters are numTrees(which is the number of trees) and maxDepth (the maximum Depth of each tree). Increasing the tree number might increase accuracy but might not the variance. The depth increase is usually positive but might increase the training time. There are other parameters like subsamplingRate and featuresubset strategy. Changing these parameters from default decreases the accuracy and is also not recommended by the documentation. Overall, the accuracy obtained for a range of parameters is as 

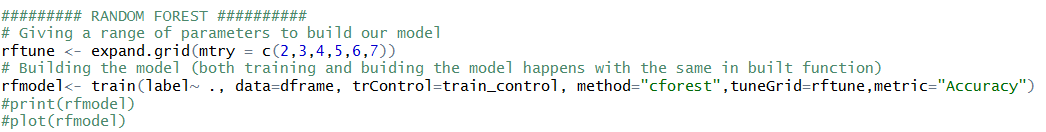
From the output snippet , we can see that the random forest classifier had best accuracy of 0.82 for num of trees=15 and maximum depth= 6 .

* + 1. **R Library**

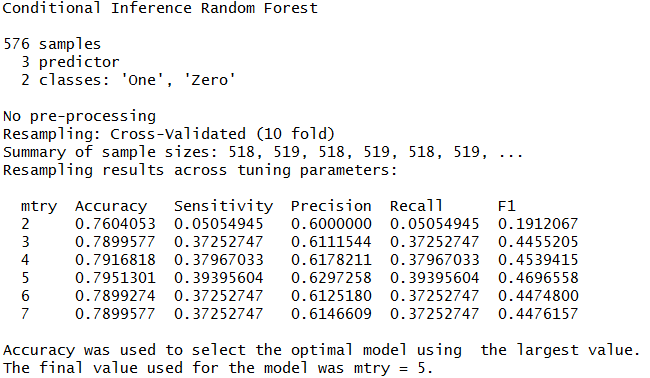
Package –CARET

Function – train

model =”cforest” ; parameter=mtry (#Randomly Selected Predictors)



The Output snippet is as follows –



We can see that its best trained with a accuracy of 0.795 for mtry=5

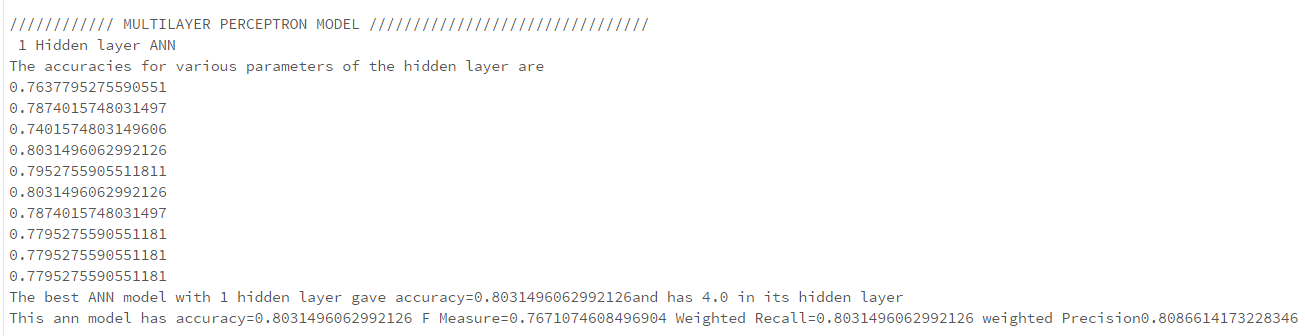
* 1. **Multilayer Perceptron**

Feedforward Artificial Neural Networks or Multilayer Perceptron classifiers are today one of the most popular deep learning classification. They have been very successful in especially image classification and text classification which encouraged us to apply to our model.

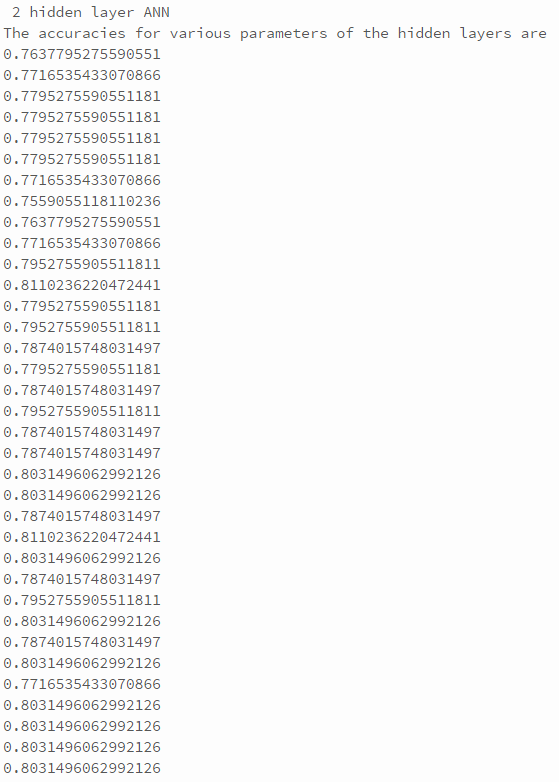
* + 1. **Spark**

We have trained our ANN in spark using 1 hidden layer and 2 hidden layers. The number of hidden layers and the number of nodes in each of the hidden layers is the parameter of selection for us in case of Spark MP library. So we have developed models for various number of nodes and various number of levels. The parameters are sent in the form of an array. The number of nodes in the input layer is equal to the number of features (3) and number of nodes in output layer is equal to the number of classes in the label (2). Nodes in intermediate layers use sigmoid (logistic) function and nodes in the output layer use softmax function. The Multilayer Perceptron employs backpropagation for learning the model. It uses the logistic loss function for optimization and L-BFGS as an optimization routine.

The training snippet for 1 layer ANN is as follows.



The training snippet for 2 layer ANN is as follows



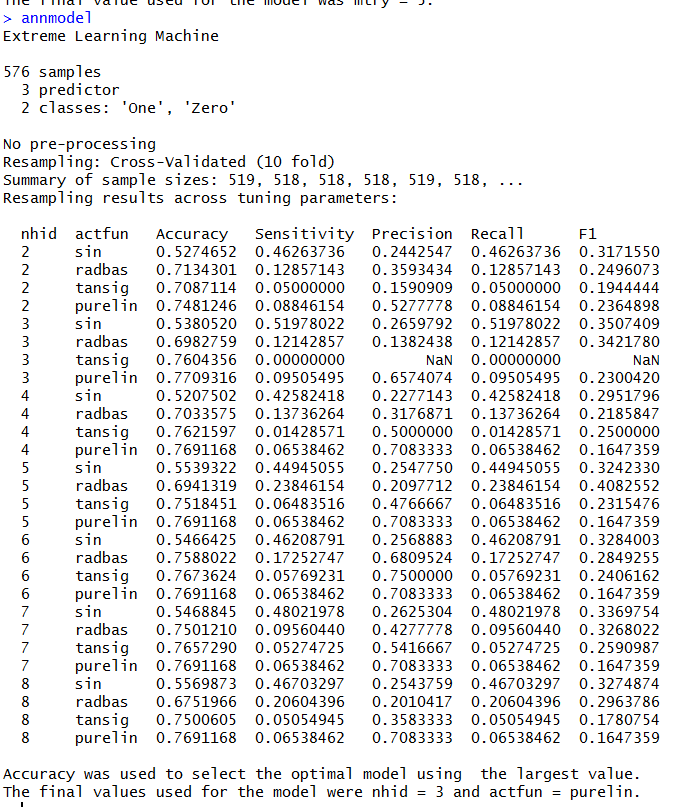
Thus the best model was a 2 layer ANN with number of hidden nodes = 2 in each of its layers.

* + 1. **R**

Package –CARET

Function – train

model =”elm” ; parameter=nhid (#no of hidden layers) ; actfun (activation function)



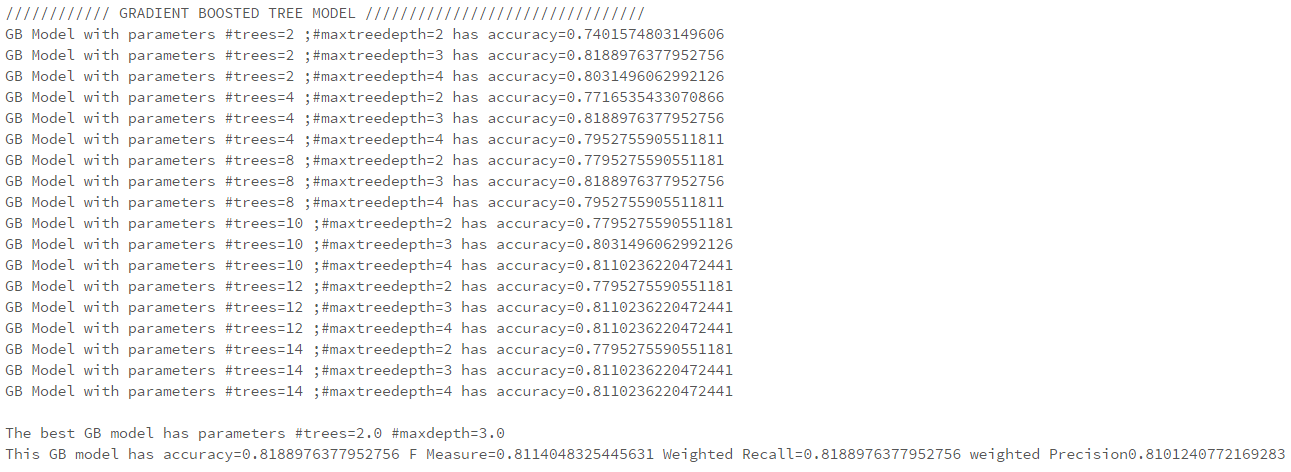
The best model had nhid=3 and a linear activation function with an accuracy of 0.77.

* 1. **Boosting(GBT)**

GBT and other Boosting techniques have recently found to be very good for problems on classification. This technique builds multiple trees and creates a final model by aggregating them and minimizing over a loss function. Both Gradient-Boosted Trees (GBTs) and Random Forests are algorithms for learning ensembles of trees, but the training processes are different. GBT are slower to train as they train one tree at a time and don’t work in parallel. Also, GBT are more prone to overfitting and hence can reduce accuracy on new test data.

* + 1. **Spark- GBT**

The parameters are loss function, number of trees, learning rate. The loss function is set as log loss which is to be minimized. The learning rate is set to default as changing this is not recommended in the documentation and won’t improve accuracy. The number of trees is varied and the best choice is chosen based on accuracy metrics.



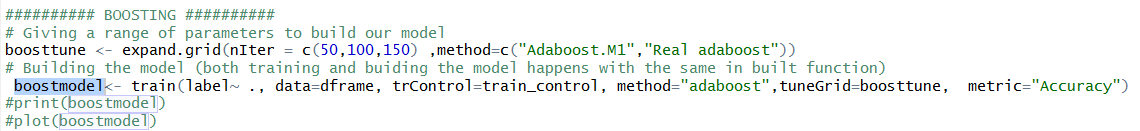
The best boosting model had 2 trees and a maximum tree depth of 3 with accuracy of 0.81

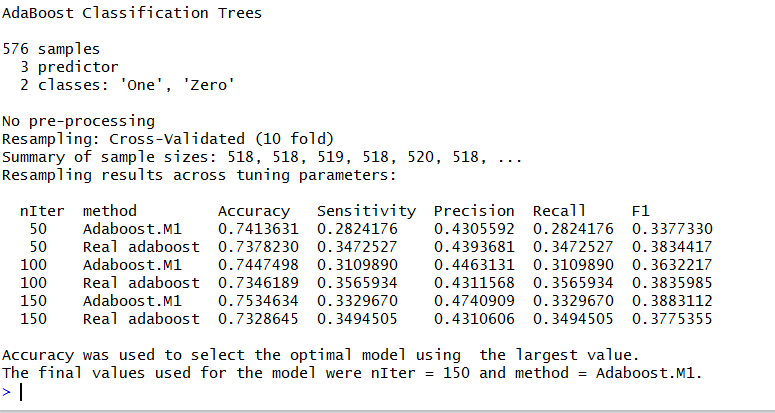
* + 1. **R - AdaBoost**

Package –CARET

Function – train

model =”adaboost” ; parameter=nIter (#trees) ; method





The best model had an accuracy of 0.75 and n.trees=150 and method =Adaboost.M1 as its parameters

1. **Comparative Model Evaluation**

In this stage each of the models that are developed are evaluated based on various metrics. We additionally do a comparative analysis on Spark Mllib vs the classifiers built using R packages.

Initially while using spark we calculated these 4 metrics: Precision, Recall, Accuracy and FMeasure metrics.

Accuracy is the proportion of the total number of correct predictions.

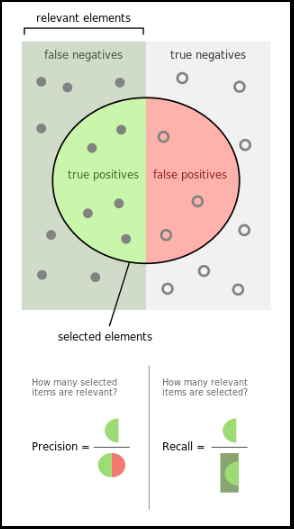
Precision is the positive predictive value i.e. the proportion of positive cases that were correctly identified. This is how it takes in consideration the entire selected component that includes true positives and false positives.

Recall or sensitivity is the proportion of negative cases that are correctly identified. Recall takes into consideration true positives and the corresponding false negatives (which were ignored by precision) and tells us about the how many positives were classified correctly amongst all the positive scenarios.

f-measure however considers both precision and recall and tells us about the true accuracy of our model.

It is well known that “*A measurement system is considered valid if it is both accurate and precise*.” This is why we chose accuracy for modelling our data, although precision is also a good option, but as per our dataset the elements with the high values of accuracy also have high precision.

The comparison between the various values of metrics have been given the previous discussions above.

  
*<<Ref: https://en.wikipedia.org/wiki/Precision\_and\_recall>>*

From the screenshots attached so far in the report we conclude that gradient boosting performs well but it is unreliable and as seen by testing the model several times we clearly saw that **RANDOM FOREST has the best performance** keeping in mind the accuracy of the models. Random Forest is followed by **ANN which is the 2nd best model**.

These results are in line with what we expected since Random Forest is several other models in terms of the following:

* GBT need way more tuning than random forests. It’s quite time consuming to tune an algorithm to the max for each of the many datasets.
* GBT are more susceptible to jiggling data making it more prone to overfit, so if the test data is more verbose than train data this algorithm is not better than RF in our case as it cuts down on overfitting.
* Features such as averaging and randomization (while growing of tree) help Random Forest to estimate several functions and maintain a lower rate of error by reducing variance at the same time.
* ANN is a good choice since it can be made deep and has the capability to work with huge amounts of features.
* Although ANN is also a good choice but RF is simpler to tune and works well on categorical inputs making it a model of our choice as per our dataset.

The table below shows the difference between the models:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **Accuracy** | **F measure** | **Weighted Recall** | **Weighted precision** |
| Random Forest | 0.826771654 | 0.81272091 | 0.826771654 | 0.819428232 |
| ANN 1-hidden layer | 0.803149606 | 0.76710746 | 0.803149606 | 0.808661417 |
| ANN 2-hidden layer | 0.811023622 | 0.79569554 | 0.811023622 | 0.800072161 |
| Gradient Boosting | 0.818897638 | 0.81140483 | 0.818897638 | 0.810124077 |

As far as Spark vs R packages are concerned, below are the major motivations of choosing spark:

* We have found spark libraries to deliver higher level of accuracies, which is one of the main reasons for its popularity in recent times.
* Apart from this spark enables a framework of parallel processing and hence fast data mining applications can be run.
* Due to the above reasons and also because it captures several R functionalities, its good for scaling and gives high performance.
* For large amounts of data, the computation speed has huge difference, where spark has much higher computation power and is much faster than R.
* It also gave us a chance to work on something new along with our work in R.

1. **Conclusion**

We understood how blood donations work and what impact it would have to understand the data to be able to predict donations. We used some complex models to predict the categorical output of whether the blood donor will donate blood within a specific time window, which depends hugely on number of donations made in the last few months, how much volume of blood has been donated and when did the person first donate blood. This information can be broadly classified into two major factors, the “intervals of donations” w.r.t a donor and the “amount of blood” he/she donates. We found that it can very well predicted with an accuracy of **approximately 82%** whether a person will donate blood during a specified interval which is March 2007 in our case.

We succeeded in this “DrivenData” competition by finishing at “20th Rank” in the online competition (<https://www.drivendata.org/competitions/2/leaderboard/> - submission is in the name of KrishnaDodda) and learnt how well each model works of this dataset concluding Random Forest to work the best amongst Random Forest, ANN and Gradient Descent Tree. The reasons for RF performing well with are data have been specified in section 6.

1. **References**
2. <https://www.analyticsvidhya.com/blog/2016/02/7-important-model-evaluation-error-metrics/>
3. R CARET Package
4. Spark Mllib 2.0
5. Course Lecture Slides
6. https://www.drivendata.org/competitions/2/