Data Science and Applied Machine Learning

The Australian National University

Submitted By

u6742441(Prateek Arora) and u6651968(Sambit Ghosh)

Regression:

Data containing 100000 users were first used on the rattle. Since Rattle only uses numeric column for exploring correlations, botscore had to be clean partially. This was done by removing the 'deleted','suspended' and 'protected' values with NAs. Through association and correlation in the rattle, the variables that are important for the evaluation of regression and classification of data were known due to which important features can be added for the regressor. Through this analysis, **bias_polarity, statusesCount, utcoffset, listedcount , favouritescount, followerscount , verified, location.objType , mcsize , influence, influence percentile, tweetscount , retweetscount** are the features which would be used for regression algorithm. The important variables according to rattle exploration is shown below (Fig 1). User_id, even though is shown to have a greater correlation than most other values is not included as they are mostly just identifiers assigned by twitter and do not hold and inherent meaning. Training of such data would throw off the predictions in random ways.

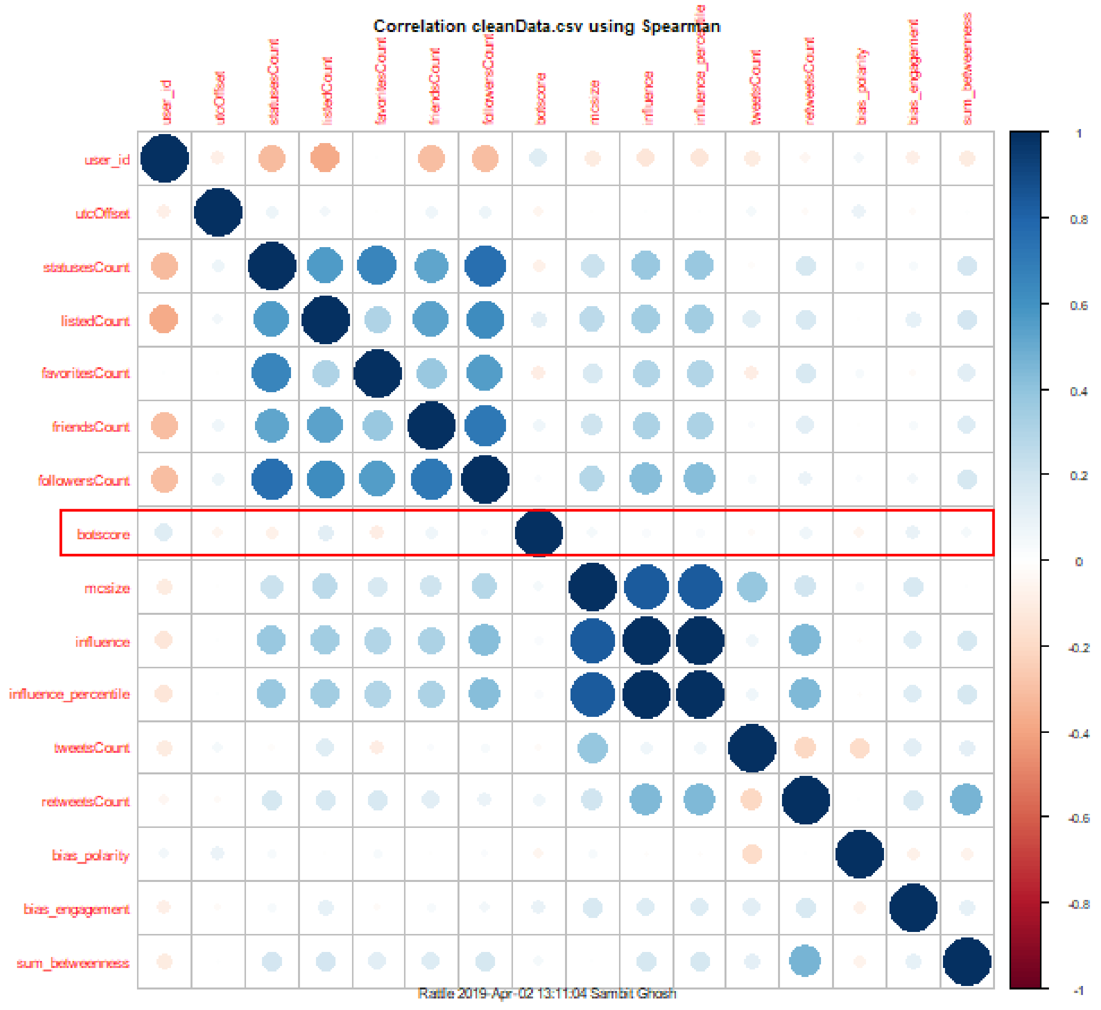


Figure 1

Then the data cleaning was done to ensure that all Na or Nan values were either removed or was replaced by the default values assigned to it. For bias_polarity and statuesCount, all na's were replaced by 0. If the value of botscore was less than 0 or if it was na, then it was deleted so as to keep the data clean. Similarly, **mcsize** was cleaned by replacing na values with 0. **location.objectType** was also cleaned, if it had a place mentioned then it was turned to numeric value 1 otherwise if no value was available they were filled with 0. This indicates that whether the account has a valid place location or not. All other records were dropped if there was no value in any of the selected columns.

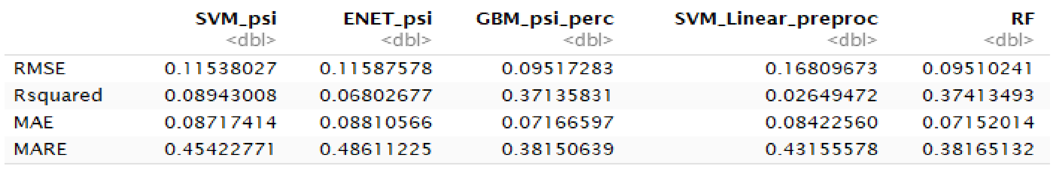
After the cleaning, only 52079 users' values were left. Here a fit control is used for training of data in which number is equal to 5 which means that the training data is being divided into the 5 parts and that 5 parts, values are assigned randomly. Here, as repeats is equal to 10, so each fold will be executed 10 times. This is because the data is less, and the data might not be enough to generate a good prediction model.

The cleaned data was separated for training and testing. For training the first 35000 values were included and the remaining were kept aside for testing. Which is roughly a 70-30 training to testing split ratio. This was reached through a trail and error. 60-40 and 50-50 were worse in performance.

Pre-processors used in the regression were center, scale, YeoJhonson, nzv and pca. Here, center subtracts the mean of predictor's data from predictor values while scale divides by the standard deviation. Scale makes the data condense to between -1 and 1. YeoJhonson is used to transform the predictor variables and nzv identifies numeric predictor columns with a single value and excludes them from further calculations.[1] These pre-processes are largely a trial and error-based inclusion. But scaling and centring in general tends to make the model more generalised as they remove the bias a particular feature might have just due to the fact that its range is wider. Essentially, standardize the data.[2]

LM was used to compute the regression but was discarded as the value for the rmse was quite high, i.e., 0.12. The other reason was that the linear regression is meant for the simplified values so that it can be differentiated easily through a line. But here, it didn't go that well because the values were around 52000 and all the values were scattered.

Other algorithms used for regression were SVM_Linear, SVM_Radial, Elastic Net, gbm (Gradient Boost), ranger (denoted by RF). All these were used to compute the rmse values. Goal for all these regressors were to decrease the value of rmse (root mean square error). The rmse values for the respective regressors are shown below. (Fig 2)



	SVM_psi	ENET_psi	GBM_psi_perc	SVM_Linear_preproc	RF
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
RMSE	0.11538027	0.11587578	0.09517283	0.16809673	0.09510241
Rsquared	0.08943008	0.06802677	0.37135831	0.02649472	0.37413493
MAE	0.08717414	0.08810566	0.07166597	0.08422560	0.07152014
MARE	0.45422771	0.48611225	0.38150639	0.43155578	0.38165132

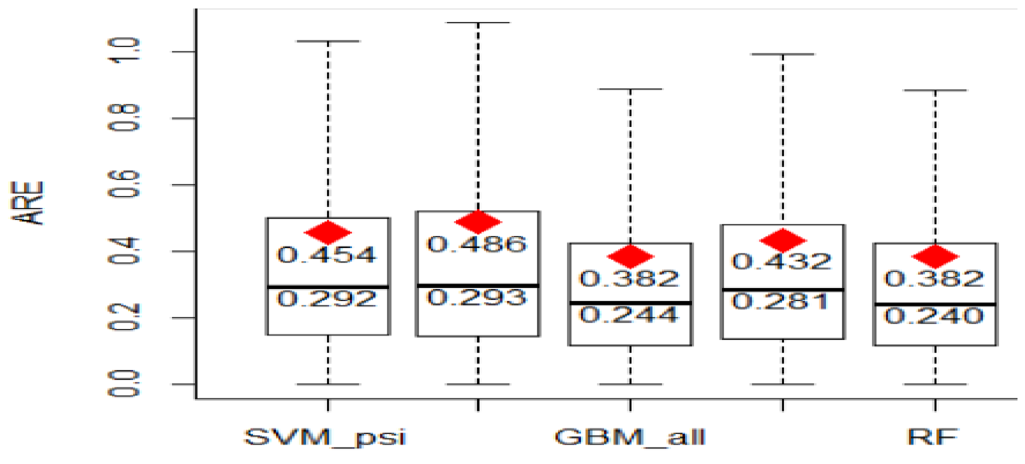


Figure 2

As it can be deduced from the above data, gbm and ranger worked almost identically well but, ranger was a fraction better and therefore it was kept to produce the predictions for the given test set. Ranger is a version of random forest for high dimensional data. Random forest is designed to work well on non-linear data.[3] Since the data available for training is non-linear it is seeming to work well on this data.

Classifier:

First, data is checked so as to gauge what needs to be done. A box plot for the distribution of the values of botscore is plotted, as shown below in Figure 3.

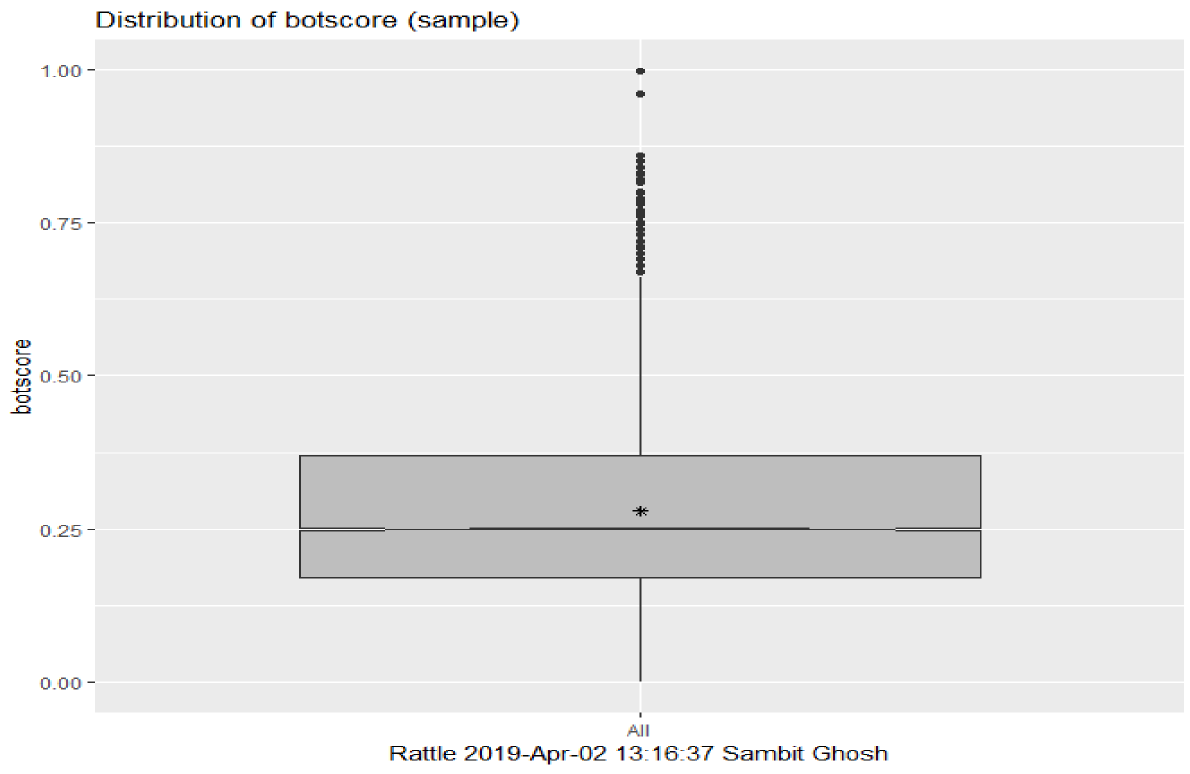


Figure 3

From the above plot, it is clearly seen that most values are below 0.50. This means that the training data has much more not bots than bots. This means out classifier would likely get biased towards predicting not bots.

The same cleaning methods are applied as before. We use the same 52079 values we got while cleaning for regression and use the same features. But this time the given botscore column is removed because that is what the training labels are based on.

This time the split is done 50-50 at random. Differently from the regression, random rows is used to split into 50-50 rather than first few or last few. This is to increase the chances that the bots records would be more evenly spread. This hypothesis is later confirmed by testing other splits which did not perform well.

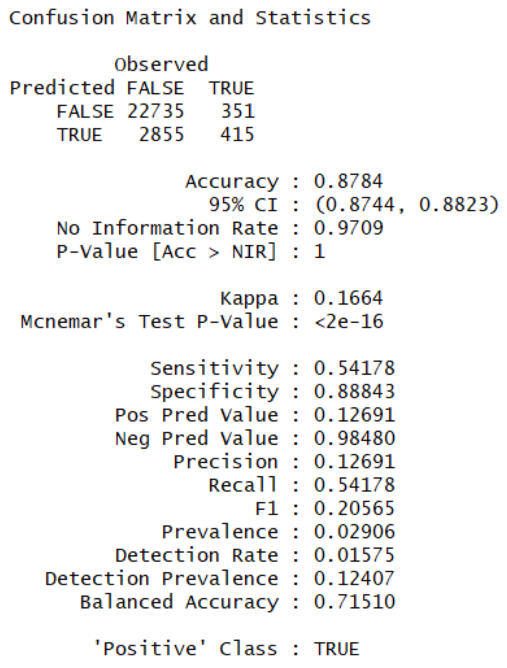
Apart from splitting to get evenly distributed values, upsampling of the bot records was also done. This made the data have equal amounts of bots and not bots by repeating the bots records at random and adding them to the dataset. This would help the training algorithm to learn to predict both bot and not bots accurately.

This time the fit control is also changed to 5 folds and 5 repeats. Repeats are lowered because the data is more due to upsampling and does not require as much data simulation as the regressor would. Also, this makes the training faster.

The pre-processing for the classifier is just scaling and centring due to reasons previously stated.

Glm is used as a classifier here because there are many variables and entries which sum up to a large number and glm is basically used for large datasets and perfectly gives high accuracy.

Then, Confusion matrix has been used for calibrating the output of a model and examining all possible outcomes of your predictions (true positive, false positive, true negative, false negative).[4] The matrix has been shown below which is basically used to determine the accuracy.



```
Confusion Matrix and Statistics

         Observed
Predicted FALSE  TRUE
    FALSE 22735   351
    TRUE   2855   415

               Accuracy : 0.8784
                 95% CI : (0.8744, 0.8823)
    No Information Rate : 0.9709
    P-Value [Acc > NIR] : 1

                  Kappa : 0.1664
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.54178
            Specificity : 0.88843
         Pos Pred Value : 0.12691
         Neg Pred Value : 0.98480
              Precision : 0.12691
                 Recall : 0.54178
                     F1 : 0.20565
             Prevalence : 0.02906
         Detection Rate : 0.01575
   Detection Prevalence : 0.12407
      Balanced Accuracy : 0.71510

       'Positive' Class : TRUE
```

****

Finally, we take the two prediction models and create the required predictions csv file. But to do this first we need to match up the given prediction set to the training set we used. Otherwise our models would not be able to treat it as input data to give predictions on. To do this the same cleaning and feature engineering are applied on the test data as before. This leaves us with 11842 values to predict on. The models are then used to generate the predictions and store them in the required csv file.

References:

[1] https://www.rdocumentation.org/packages/caret/versions/6.0-81/topics/preProcess

[2] https://www.datacamp.com/community/tutorials/preprocessing-in-data-science-part-1-centering-scaling-and-knn

[3] https://www.rdocumentation.org/packages/ranger/versions/0.11.2/topics/ranger

[4] https://www.datacamp.com/community/tutorials/confusion-matrix-calculation-r