Pre-processing and Modelling using Caret Package in R

Ajeet Kumar Rai
Department of Mathematics
NIT Warangal
Telangana, India

ABSTRACT
To Implement Machine learning algorithms there are a number of Tools like python, R, Apache Mahout, Cloud-based services etc. In some tools, there are different packages that help us in data preprocessing and implementing machine learning algorithms. So, in this paper, aim to discuss how can use caret package in R software to implement machine learning techniques.

General Terms
R Programming, Caret Package, Statistical analysis, Data Pre-Processing.

Keywords
Titanic Dataset, Machine Learning, Decision Tree, Random Forest, Confusion Matrix, ROC Curve.

1. INTRODUCTION
Since in python scikit-learn is a famous library for implementing machine learning algorithms. Similarly, for R users there is a package called Caret. It not only implement models but also helps in pre-processing. So, first starting with data cleansing, handling factors and then dividing the Titanic dataset into training data and test data. After doing above process can be to implement different models like a Decision tree, random Forest etc. Lastly analyzing evaluation error metrics and performance on the unknown test data. The dataset used here is Titanic data has been taken from Kaggle.com containing 892 rows 12 variables. In this dataset survived is target variable containing values 0 and 1.

2. CARET PACKAGE
Name Caret comes from Classification And Regression Tree. R users can use this package for data splitting, Pre-processing, feature engineering, parameter tuning, modeling and variable importance estimation. But, here topic restrict to use splitting, pre-processing and modeling functions only.

3. METHODOLOGY
The main objective not to just implement the models, but to show how caret package useful in data wrangling as well. Mainly focused on the following:

- Replacing Missing terms with appropriate values.
- Dealing with factors.
- Data Splitting.
- Implementing models like Decision tree and Random Forest.

So, now starting by loading dataset in R studio /R and here data named as titanic _dataset.

3.1 Replacing Missing Terms With Appropriate Values
Implementing models by ignoring missing values may lead to an error, of course, some algorithm like decision tree is not sensitive to missing values. So, to avoid that error first should check if there are any missing values. In Titanic dataset, there are total 177 missing values. In Caret package there is function preProcess that help us to remove missing values. PreProcess takes two values first data that is titanic dataset and method. In method it takes three values as a vector that is scale, center and medianImpute/knnImpute. Depending on the dataset we choose medianImpute/knnImpute which replace NA values either by taking Median or applying Knn algorithm. The scale will set all attributes on the same scale by either dividing maximum value or calculating mean deviation and divided by the maximum value. In preProcess there can be more input which are optional for this dataset. After implementing the above function there are no missing values.

3.2 Dealing With Factors
In R, attributes with factor datatype that is categorical variables again may lead to an error if values in columns are more than the maximum value of nlevels. So, there is the dummyVars function in caret that can helps to handle these categorical attributes. It makes a new column for each value in categorical variables. dummyVars takes three values that are ~., data and fullRank. Here tilde "~." will take all the attributes. fullRank should be true and it takes (n-1) columns for attributes with factor with n different levels.

3.3 Data Splitting
In kaggle competition they already have test data excluding target variable. but here using the same titanic _dataset for validation as well, by dividing data into the training and test dataset. Here, it's better to divide dataset three times in different proportion and then applying models on three different test data. Proportion will be 60% and 40%, 50% and 50%, 40% and 60%.In Caret createDatapartition function will divide the data in train and test dataset and have to use this function three times in different proportions. CreateDatapartition takes three values that are data, the percentage of the training set and the list that should be False. For first proportion it takes 60% that is p=0.60 remaining dataset will automatically use for validation and repeating the same thing for remaining two proportions. while doing above process data should be replaced every time for each proportion.

4. MODELING
After doing data Pre-processing now ready to apply different Machine learning algorithms to see the predictions on the test data. Machine Learning problem can be categorized in supervised, unsupervised and reinforcement learning. if the dataset has both predictors and target variable than it can be classified as supervised learning. In the absence of target variable, it classified as unsupervised learning. In supervised learning, there are another two types of problem that is Regression and Classification problem. There are many regression problems like linear regression, multiple linear regression, ridge regression, lasso regression etc. and some classification problems are a logistic regression, support

vector machine, Naive Bayes, decision tree, random forest, boosting etc. Before applying algorithms it's good to use feature engineering, but escaping that part and moving to the first model Decision tree.

4.1 Decision Tree

Decision tree works with both regression and classification problem. Since in dataset target variable has two values 0 and 1 that is its binary classification problem. A Decision tree is nothing but the graphical representation of solutions based on the certain conditions. It has a root node, internal node, and leaf node as shown in below diagram.

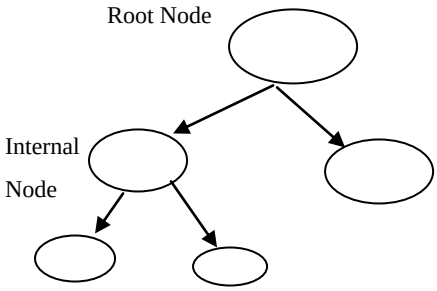

Fig 1: Decision tree

It uses the information gain/Entropy to select the most appropriate variables. The decision tree is not sensitive with anomalous and can handle the missing values. Its split variables in such way that leaf node contain only homogeneous data. Entropy is calculates as:

Entropy= $-p*\log_2 p - (1-p)*\log_2(1-p)$, where p is probability

One disadvantage of using this model is that it's generally overfitting. So, in machine learning there is method called ensemble method which can help us to deal with overfitting. Ensemble method can be cauterized in Bagging, Boosting and Stacking. Now, we will use bagging technique and one of the bagging technique is Random Forest.

4.2 Random Forest

Random Forest also works with both Regression and Classification problems. It builds a number of the Decision tree and adds them together to get a more effective result. It can also be used for variable importance estimation. In this Algorithm, each tree uses 63.2% of data for training with replacement. Remaining 36.8% data used to calculate out of the bag error i.e. misclassification rate. even in the random forest there is no need of validation data since OOB rate itself enough for analyzing unknown data. To choose random variable we can use mtry value or by default, it takes a square root of a total number of predictors in classification problem or it divides the total number of predictors by 3 for regression problem. Random Forest is a good choice if the model is suffering from the High Variance problem.

5. COMPLETE CODE IN R USING CARET PACKAGE

```
# Loading datadata=read.csv("titanic_dataset.csv",header=TRUE,strings AsFactors = F)

#installing and loading caret package

Install.packages("Caret")

library(caret)

preprocvalues=preProcess(data,method=c("medianImpute","center","scale"))

 #taking median for all NA values with respective variables & adjusting variables on same scale.

library(RANN)

data_pro=predict(preprocvalues,data)

sum(is.na(data_pro))

[1] 0                          #total 0 NA values

#removing unnecessary variables, of course these variables can be used for feature engineering.

data_pro=subset(data_pro,select=-c(Name))

data_pro=subset(data_pro,select=-c(PassengerId))

data_pro=subset(data_pro,select=-c(Ticket))

str(data_pro)

'data.frame': 891 obs. of  9 variables:

 $ Survived: num  -0.789 1.266 1.266 1.266 -0.789 ...

 $ Pclass  : num  0.827 -1.565 0.827 -1.565 0.827 ...

 $ Sex     : chr  "male" "female" "female" "female" ...

 $ Age     : num  -0.53 0.571 -0.255 0.365 0.365 ...

 $ SibSp   : num  0.433 0.433 -0.474 0.433 -0.474 ...

 $ Parch   : num  -0.473 -0.473 -0.473 -0.473 -0.473 ...

 $ Fare    : num  -0.502 0.786 -0.489 0.42 -0.486 ...

 $ Cabin   : chr  "" "C85" "" "C123" ...

 $ Embarked: chr  "S" "C" "S" "S" ...

# Now having only 9 variables including target variable.

dv=dummyVars("~.",data_pro,fullRank = T)
data_tran=data.frame(predict(dv,data_pro))

str(data_tran)

'data.frame': 891 obs. of  157 variables:

 $ Survived          : num  -0.789 1.266 1.266 1.266 -0.789 ...

 $ Pclass            : num  0.827 -1.565 0.827 -1.565 0.827 ...

 $ Sexmale           : num  1 0 0 0 1 1 1 1 0 0 ...

 $ Age               : num  -0.53 0.571 -0.255 0.365 0.365 ...

#here just showing few lines of code. Now there are 157 variables and all are numeric datatype including target variable. But need to convert again target variable into factor datatype.

data_tran$Survived=as.factor (data_tran$Survived

set.seed(5)

index <- createDataPartition(data_tran$Survived, p=0.60, list=FALSE)      #Data Splitting

train <- data_tran [ index,]      #Training data=60%

test<- data_tran [-index,]        #Test data=40%

#Implementing Decision Tree

model_rpart<-train(Survived~.,test,method='rpart')
```

prediction=predict.train(model_rpart,test,type="raw")

confusionMatrix(table(predict(model_rpart,test),test$Survived))

Accuracy on validation dataset.

Accuracy : 0.8394

95% CI : (0.797, 0.8761)

No Information Rate : 0.6169

P-Value [Acc > NIR] : < 2e-16

Kappa : 0.6531

Mcnemar's Test P-Value : 0.06369

Sensitivity : 0.9041

Specificity : 0.7353

Pos Pred Value : 0.8462

Neg Pred Value : 0.8264

Prevalence : 0.6169

Detection Rate : 0.5577

Detection Prevalence : 0.6592

Balanced Accuracy : 0.8197

'Positive' Class : -0.788829297442119

using another evaluation error metrics ROC curve.

for_auc=predict(model_rpart,test,type="prob")

library(pROC)

area_under_curve=auc(test$Survived,for_auc[,2])

area_under_curve

Area under the curve: 0.8379

plot(roc(test$Survived,for_auc[,2]),main="Decision Tree",col="green")

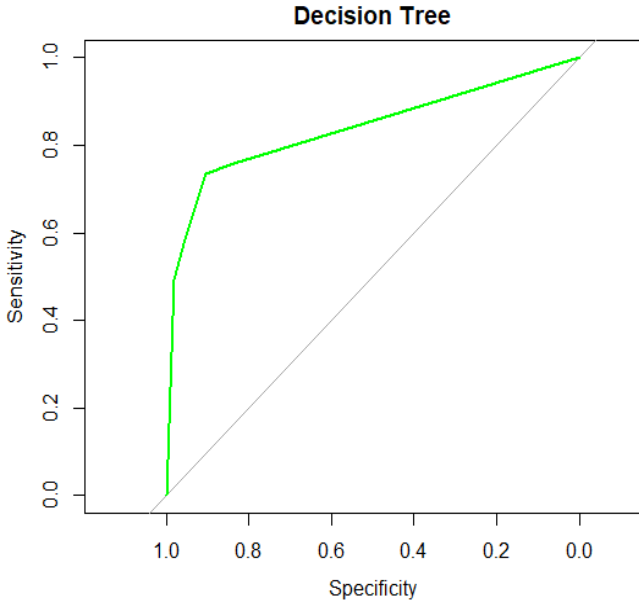

Fig 2: ROC Curve for Decision tree

#Now implementing random Forest.

Set.seed(100)

model_rf<-train(Survived~.,test,method='rf')

prediction=predict.train(model_rf,test,type="raw")

confusionMatrix(table(predict(model_rf,test),test$Survived))

Accuracy : 0.9775

95% CI : (0.9636, 0.9938)

No Information Rate : 0.6169

P-Value [Acc > NIR] : <2e-16

Kappa : 0.9641

Mcnemar's Test P-Value : 0.6831

Sensitivity : 0.9909

Specificity : 0.9706

Pos Pred Value : 0.9819

Neg Pred Value : 0.9851

Prevalence : 0.6169

Detection Rate : 0.6113

Detection Prevalence : 0.6225

Balanced Accuracy : 0.9807

'Positive' Class : -0.788829297442119

#ROC Curve

for_auc=predict(model_rf,test,type="prob")

library(pROC)

area_under_curve=auc(test$Survived,for_auc[,2])

area_under_curve

Area under the curve: 0.9975

plot(roc(test$Survived,for_auc[,2]),main="Random Forest",col="red")

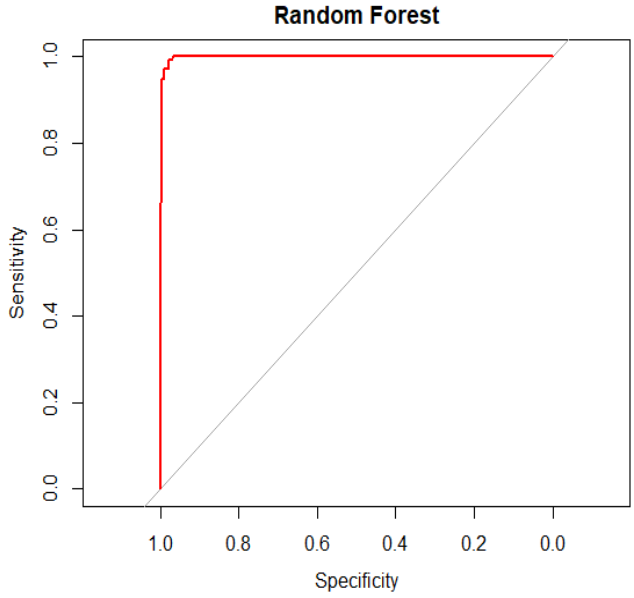

Fig 3: ROC Curve for Random Forest

#after applying random forest new accuracy is 0.9775, which shows random forest is better than decision tree.

#repeating the same code two times more with different proportions of training and test dataset.

Now, comparing the results.

Table 1: Algorithm and Percentage of accuracy

Algorithms	Train-60% Test-40%	Train-40% Test-60%	Train-50% Test-50%
Decision Tree	0.8394	0.8146	0.7978
Random Forest	0.9775	0.9738	0.964

6. CONCLUSION

After doing the pre-processing and modelling and presenting the results in the above table. Results clearly show that random forest performs better than a decision tree. Also pre-processing help us in avoiding errors. There are other packages also those help R users for data wrangling and statistical analysis without doing complex coding, but by studying caret package came to know that this one package is enough to building machine learning system.

7. ACKNOWLEDGEMENTS

My special thanks to my parents, Prof. J.V Ramana Murthy Sir and my friends who encourage me to write this research paper.

8. REFERENCES

[1] Hastie T, Tibshirani R, Friedman JH (2001). The Elements of Statistical Learning. SpringerVerlag, New York. URL http://www-stat.stanford.edu/~tibs/ElemStatLearn/.

[2] Titanic: Machine Learning from disaster https://www.kaggle.com/c/titanic Algorithm-d457d499ffcd

[3] Non-Linear Classification in R with Decision Trees. (2016, September 21). Retrieved February 23, 2018, from https://machinelearningmastery.com/non-linear-classification-in-r-with-decisiontrees/

[4] Kuhn, M. (2017). The Caret Package. GitHub. Retrieved 14 December 2017, from https://topepo.github.io/Caret/

[5] The CARET package, Max Kuhn. Contributions from Jed Wing, Steve Weston, Andre Williams, Chris Keefer and Allan Engelhardt (2012). caret: Classification and Regression Training. R package version 5.15-044.

[6] Kaggle, Data Science Community, [Online]. Available: http://www.kaggle.com/ [Accessed: 2-Jun-2017

[7] Prediction of Survivors in Titanic Dataset: A Comparative Study using Machine Learning Algorithms Tryambak Chatterjee* Department of Management Studies, NIT Trichy, Tiruchirappalli, Tamilnadu, India

[8] Statistics review 13: Receiver operating characteristic curves. Critical Care (London, England), 8(6), 508512. http://dx.doi.org/10.1186/cc3000

[9] X. Wu, V. Kumar and J. R. Quinlan, "Top 10 algorithms in data mining", Knowledge and Information Systems, vol. 14, no. 1, (2008), pp. 1-37.

[10] N. Bissantz and J. Hagedorn, "Data mining", Business and Information Systems Engineering, vol. 1, (2009), pp. 118-122

[11] Eric Lam, Chongxuan Tang. Titanic – Machine LearningFromDisaster.AvailableFTP: cs229.stanford.edu Directory: proj2012 File: LamTang-TitanicMachineLearningFromDisaster.pdf

[12] Trevor Stephens. (2014). Titanic: Getting Started With R - Part 3: Decision Trees [Online]. Available: http://trevorstephens.com/kaggletitanic-tutorial/r-part-3-decision-trees/

[13] Robnik M, Sikonja, (2004): Improving Random Forests, J F Boulicaut et al (eds): Machine Learning, ECML 2004 Proceedings, Springer, Berlin

[14] Zhang H, Wang M, (2009): Search for the smallest Random Forest, Statistics and Its Interface Volume.2, pp 381-388.

[15] Bradley, A.P., 1997. The use of the area under the ROC curve in the evaluation of machine learning algorithms. Pattern Recogn. 30 (7), 1145–1159.

[16] Breiman, L., Friedman, J., Olshen, R., Stone, C., 1984. Classification and Regression Trees. Wadsworth International Group, Belmont, CA.

[17] Fawcett, T., 2001. Using rule sets to maximize ROC performance. In: Proc. IEEE Internat. Conf. on Data Mining (ICDM-2001), pp. 131– 138.