Credit Card Payment Predictive Models

Ensemble v/s Non-Ensemble methods

Dataset: CreditCard Clients Dataset
https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients

Build model to predict whether the credit card user will pay the next installment amount or default it based upon last six months of payment trends.

24 attributes – Credit limit, age, education, sex, marriage, payment history details
30000 observations
Default Next Payment - Yes = 1 and No = 0

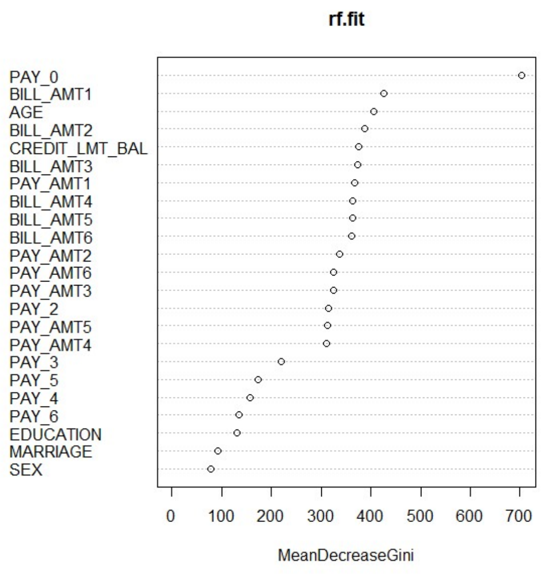
Divided data into test and train
Training Data: 21000
Test Data: 9000

Random Forest:

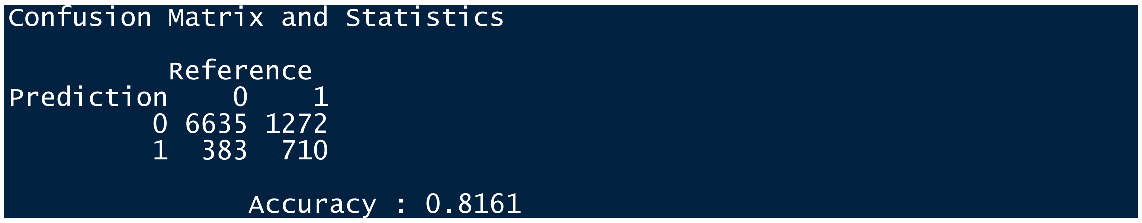
Creates a random sample of data and fits a tree for each sample of data. It results in multiple tress based on random sample of data so it's called random forest. Then on each test data is applied on each tree and the results are averaged to derive the final value in case of regression and for classification it derives the final value based on the number of votes.

Now let's build random forest model for the given data with 100 trees:

Variable importance: shows as age, credit limit, first installment amount



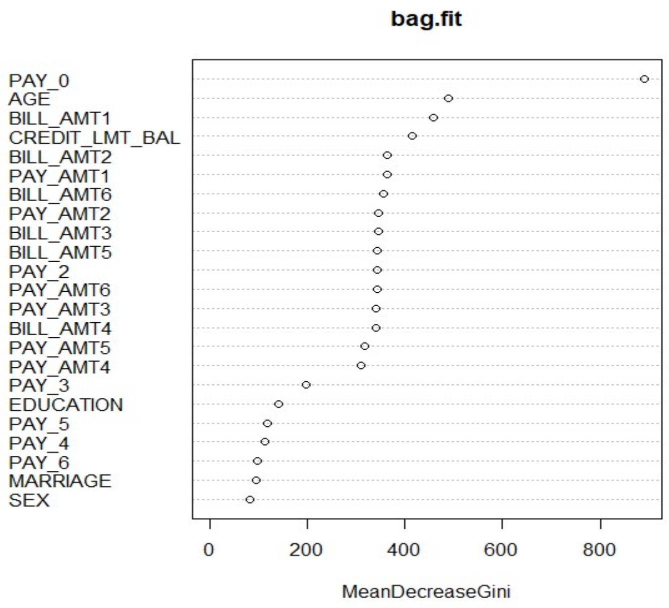
Accuracy: 81.61

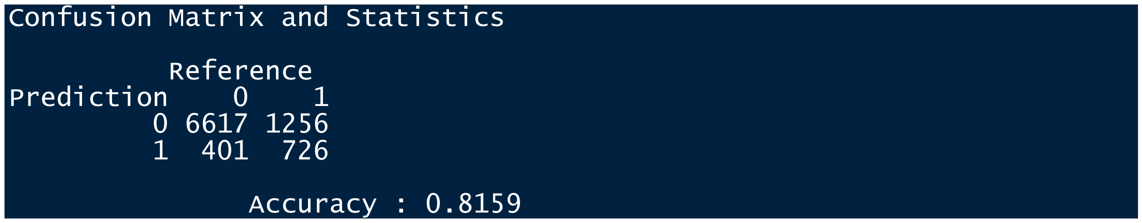


```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 6635 1272
         1  383  710

               Accuracy : 0.8161
```

Bagging:

We are using bagging in random forest – to randomly select the features for each split there by it results in unique trees of random forest. We are trying with 10 features to randomly select from for each split.



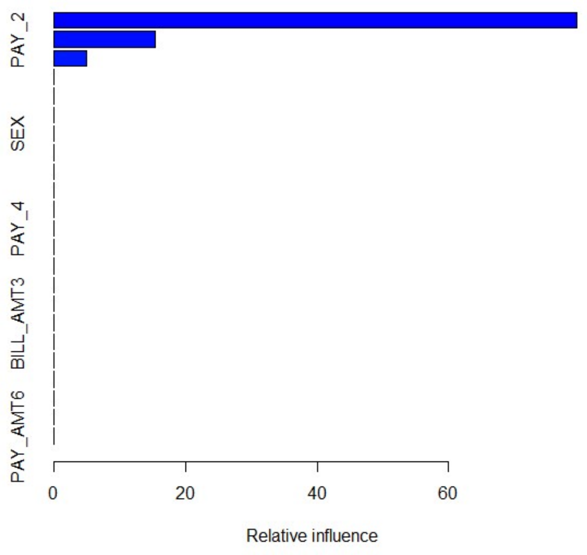


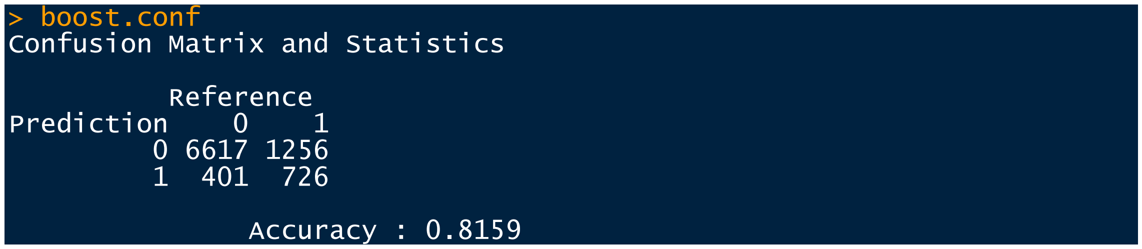
```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 6617 1256
         1  401  726

               Accuracy : 0.8159
```

Boosting:

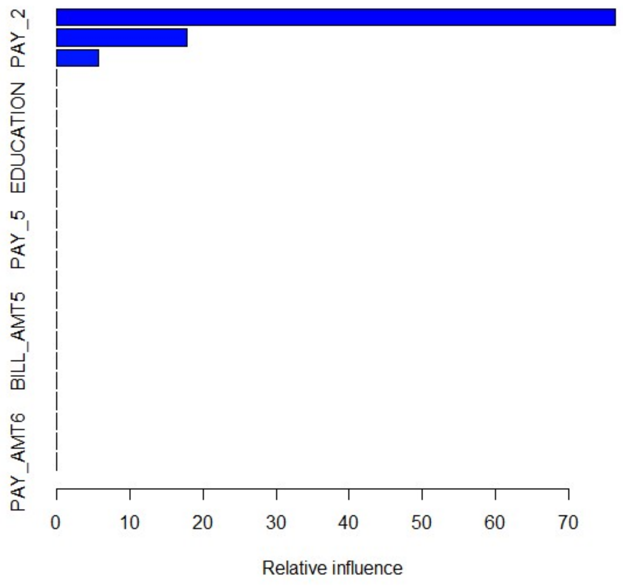
Boosting builds a strong classifier from the series of weak classifiers. It builds model and then iterate on the model to improve the performance of the model resulting in better model iteration is stopped when no further improvement of the tree is possible or when it reaches the maximum number of trees. We here use adaboost – binary classification boosting algorithm with max trees as 1000 and shrinkage of 0.1 and 0.6 this is at each iteration the performance should increase by shrinkage parameter.

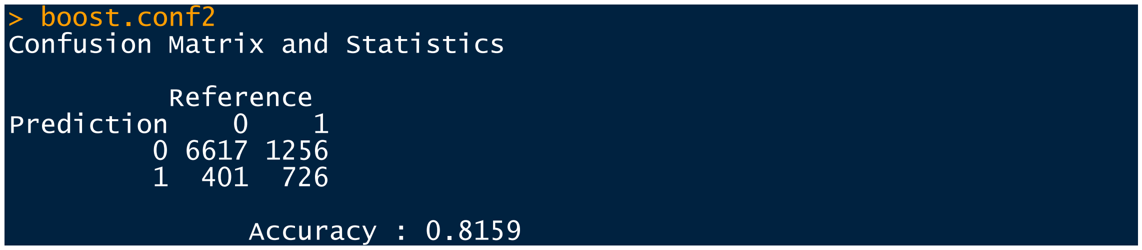
Shrinkage- 0.1




```
> boost.conf
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 6617 1256
         1  401  726

               Accuracy : 0.8159
```

Shrinkage – 0.6

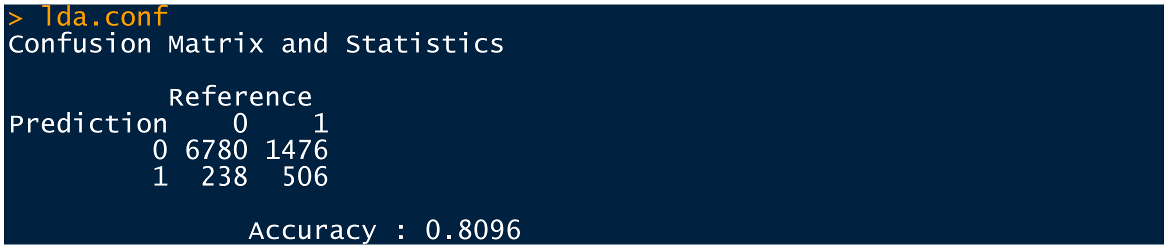



```
> boost.conf2
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 6617 1256
         1  401  726

               Accuracy : 0.8159
```

Non-Ensemble Methods – Simplistic methods – LDA, Logistic, k-NN

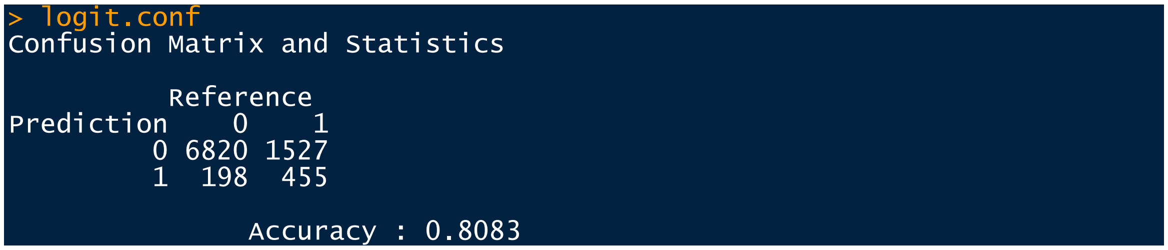
LDA: Linear discriminant Analysis – When data is continuous and numeric LDA can be used to perform classification as it depends on the class mean and all classes will have same variance. Based on this we predict the class of the data by drawing a true decision boundary b/n class.

We use class prior calculated from the dataset:
Yes- 1- 0.22
No-0 – 0.78



```
> lda.conf
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 6780 1476
         1  238  506

               Accuracy : 0.8096
```

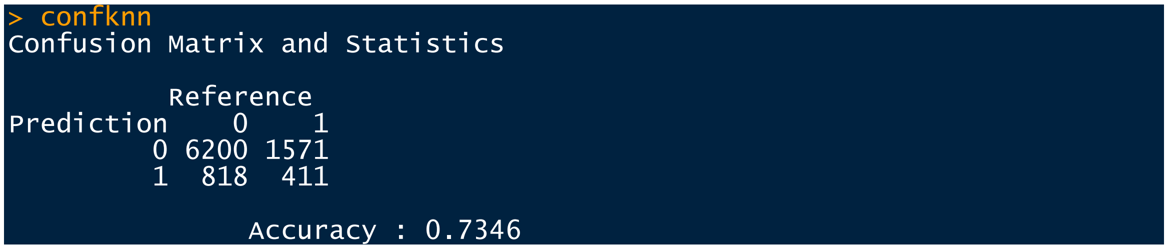
Logistic Regression: It fits data to the linear equation and then the output of linear equation passed thru the softmax function to get the probability of the class if greater than 0.5 then belongs to class1 else class2.



```
> logit.conf
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 6820 1527
         1  198  455

               Accuracy : 0.8083
```

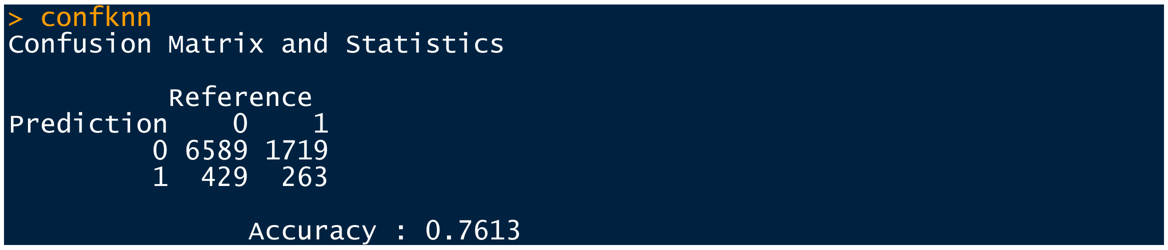
k-Nearest Neighbors: Deriving the class of the test data based on the k neighbors if number of neighbors are more than 1 based on votes the class is derived its purely based on distance metric.

We use k=3 and see how model behaves.



```
> confknn
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 6200 1571
         1  818  411

               Accuracy : 0.7346
```

k-10 and see how model behaves.



```
> confknn
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 6589 1719
         1  429  263

               Accuracy : 0.7613
```

<u>Conclusion:</u>

		Accuracy	Complexity Parameter
Ensemble methods	**Random Forest**	81.61	number of trees
	Bagging	81.59	number of trees, number of features for each split
	Boosting	81.59	shrinkage, number of trees, interaction depth
Non-Ensemble methods	**LDA**	80.96	number of features
	Logistic	80.83	number of features
	k-NN	76.13	number of neighbors

Ensemble methods are computationally intensive as multiple trees to be built and the results are averaged from different models we can't really explain how the model derives the value just by looking variable importance we could figure out which feature is more prominent and along with that the complexity parameters for these methods we have to run multiple trails to identify optimal value to tune the model's complexity parameter.

Non-ensemble methods are simplistic and less complexity parameters mainly depends on the feature selection and building model with features correlated with the response variable for better performance. LDA is simple enough just with all features, mean and covariance calculated form data it could differentiate classes nearly to ensemble methods. If LDA is used with subset of features it would have performed better than ensemble methods.

Its not always wise to choose the ensemble methods over simplistic non-ensemble methods. The methods should be evaluated for performance and simple interpretable models which are easy to tune parameters of the models and easy to scale for the dataset in hand. Non-ensemble methods always provide a go-to option due to its simplistic and interpretable nature.