INST737: Introduction to Data Science

Project Milestone-3: Report

Instructor: Dr. Vanessa Frias-Martinez

Team5 : Sadaf Nasir Davre, Tanya Gupta, Ushasri Bhogaraju

Term : Fall '23

Summary of the Datasets and Research:

We have three research questions, two of which are centered around building models to predict the 'Original Interest rate' variable, and a third research question aimed at predicting the class of the outcome variable 'Occupancy Status' accurately. To find answers to research questions 1 and 2, we started with a Fannie Mae Loan dataset in Milestone-1, into which we absorbed 3 more variables after performing linear and multivariate regression analysis in Milestone-2, as at that point we found out that none of the initial set of variables offered much predictive power. Using the new set of 15 IVs, we trained different linear regression, multivariate regression, Bagging and Random Forest models and used them to predict Original Interest Rate.

For the third research question, we found that using the 4 IVs identified in the research question, did not provide enough defining features for the models to predict the correct class of the DV Occupancy Status. Therefore in Run-2, we used all the 15 IVs and trained classifiers using Logistic regression, Naive Bayes models. For Decision Tree models with and without Boosting we used 6 IVs. When we used more variables, we had better success as our models could predict all the three classes of the DV which they could not initially. Our best model was the Decision Tree model with boosting10, which had 91% overall model accuracy. But, the NIR or the no information rate was very high at 90% meaning that if the model only predicted the most prevalent class, it would still be accurate 90% of the time. We therefore undersampled our most prevalent class for this part of the analysis.

Incorporating feedback:

1. Peer review: Reporting concisely
2. Peer review: Creating a smaller sample dataset using R, to reduce the number of values on our Graphs making them less cluttered.
3. Our own analysis: Ensuring equal representation of all classes of the outcome variable, 'Occupancy Status' in our dataset to train the models

Milestone-3 Report:

Data preprocessing:

We understood from Milestone-2 research that different models have different advantages and disadvantages because of the assumptions they make. The prediction accuracy of these models depends on their suitability to the data types of the variables and distribution of observations in the dataset. The Linear, Multivariate regression models cannot be trained if multicollinear variables are present in the

dataset whereas the Random forest model can handle them. The Logistic regression model assumes linearity and non-multicollinearity between the variables, and the Naive Bayes model assumes feature independence and normal distributions of features. These models underperform when these underlying assumptions do not hold good, such as when the dataset contains multicollinear variables. Also, we learned that, with a heavily imbalanced dataset such as ours, the models cannot be trained well to identify the under-represented classes of the outcome variable. Based on these learnings, we proceeded to

1. Eliminate multicollinear variables, benchMarkMortgageRate, No.of.Units (singular val)
2. Apply randomization and sampling techniques to create a more balanced, smaller dataset using R.
3. For training SVMs as classifiers, categorical IVs must be encoded. This was done using a one-hot encoding method.
4. For use in Neural Networks we normalized our numerical variables and encoded categorical IVs using one-hot encoding.

Base file: MS3sample.csv, dimensions 2760 obs of 14 variables (used for analysis)

Q1. Applying linear and non-linear SVMs to answer our research questions.

We propose to apply linear and non-linear SVMs as classifiers to answer the following *third* research question

3.'Can we accurately predict the class of Occupancy Status("I", "P" or 'S") in the Loan dataset, based on the IVs Original interest rate, bondRate, fedFundsRate, Borrower Credit Score and Original UPB? '

Note: This was our original third research question, to which the answers found in Milestone-2 were clear in that, the Logistic regression and Naive Bayes models could not even predict one instance of "S' class with the above IVs and the Decision Tree model with 50 boosting, predicted just 1 value for "S' class, indicating these IVs are inadequate to predict the outcome class. Since we have more variables in our dataset, we deployed all of them, and found that all the three models could predict some instances of "S" class correctly. This indicates that the defining features for "S" class are features other than those in the research question. With the additional feature deployment, our models could learn to predict all three classes, but, since our dataset was heavily imbalanced towards "P" class, our model accuracy was poor for I and S classes. We have rectified this aspect and created a more balanced dataset by undersampling the dominant classes in our dataset.

1.a. Train and test a linear SVM model.

Training and testing a linear SVM model as a Classifier to address research question No.3:

IVs and DV:

S.No	1	2	3	4	5	6	7	8	9	10	11	12	13	**14**
Variable Name	Original.Interest.Rate	bondRate	fedFundsRate	Original.UPB	Original.Loan.to.Value.Ratio..LTV.	Original.Combined.Loan.to.Value.Ratio..CLTV.	Number.of.Borrowers	Debt.To.Income..DTI.	Borrower.Credit.Score.at.Origination	Property.State	Seller.Name	Property.Type	Loan.Purpose	**Occupancy.Status**
Data type	Numeric (IV)	Numeric (IV)	Numeric (IV)	Numeric (IV)	Numeric (IV)	Numeric (IV)	Numeric (IV)	Numeric (IV)	Numeric (IV)	Nominal (IV)	Nominal (IV)	Nominal (IV)	Nominal (IV)	**Nominal (DV)**

Approach: There are two approaches when extending binary classifiers to solve multi-class problems. In the one-vs-one (OvO) approach, if we have N classes, we train N(N-1)/2 binary classifiers wherein, each classifier distinguishes between two specific classes. In the one-vs-all or the OvA approach, N binary classifiers are trained, where each classifier is responsible for distinguishing one specific class from all the other classes.

Since our outcome variable is multi-class with 3 classes, we used the ksvm() function from the kernlab package in R which uses the OvO approach. In this approach, a binary classifier is trained for every possible pair of classes, (3*(3-1)/2=3) in our case such as the following

1. Class “I” vs Class “P”
2. Class “I” vs Class “S”
3. Class “P” vs Class “S”

and the final decision is made based on the majority vote of the pairwise classifiers and the result is produced. We used the ‘vanilladot’ kernel for linear modeling of the SVM.

Data: We began with the base file MS3sample.csv with 2760 obs. of 14 variables. We encoded the categorical variables (other than the DV Occupancy status) listed above using one-hot encoding and arrived at the “final_MS3dataset.csv " dataset which we used for splitting into train and test datasets.

We trained our classifier on the train dataset with 2070 observations of 92 variables (encoded) and predicted using the test dataset with 690 observations of 92 variables.

```
> #### TRAINING LINEAR SVM CLASSIFIER USING VANILLADOT KERNEL
> library(kernlab)
> svm_classifier <- ksvm(Occupancy.Status~.,
+                        data = train,
+                        kernel = "vanilladot")
 Setting default kernel parameters
> svm_classifier
Support Vector Machine object of class "ksvm"

SV type: C-svc  (classification)
 parameter : cost C = 1

Linear (vanilla) kernel function.

Number of Support Vectors : 1630

Objective Function Value : -684.2233 -980.5347 -697.5321
Training error : 0.382609
```

Interpretation : During the training phase, the classifier used 1630 support vectors and the objective function values given in the output to decide the optimal hyperplane. Training error of the model is 0.3826 indicating that it misclassified 38.26% of the training data points.

Predicting the correct class using the above classifier:

```
> #PREDICTING USING THE CLASSIFIER
> svm_predictions <- predict(svm_classifier,test)
> ####confusion matrix
> table(svm_predictions, test$Occupancy.Status)

svm_predictions   I   P   S
              I 138  24  79
              P  37 165  36
              S  58  39 114
> agreement <- svm_predictions == test$Occupancy.Status
> table(agreement)
agreement
FALSE  TRUE
  273   417
> prop.table(table(agreement))
agreement
    FALSE      TRUE
0.3956522 0.6043478
>
```

Interpretation : The predictions table summarizes the predictions and the actual instances. The model could predict 138 instances of I class as I , 165 instances of P class as P and 114 instances of S class as S. The agreement table summarizes the count of correct predictions as 417 and incorrect predictions as 273 out of the total number of 690 observations in the test dataset. We find that the classifier correctly classified 60.43% of the class values and misclassified 39.56% of the class values.

We visualized the Confusion matrix to understand the details such as precision, recall, specificity and accuracy of the model.

```
> confusion_matrix <- confusionMatrix(svm_predictions, test$Occupancy.Status)
> confusion_matrix
Confusion Matrix and Statistics

          Reference
Prediction   I   P   S
         I 138  24  79
         P  37 165  36
         S  58  39 114

Overall Statistics

               Accuracy : 0.6043
                 95% CI : (0.5668, 0.641)
    No Information Rate : 0.3377
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.4065

 Mcnemar's Test P-Value : 0.1064

Statistics by Class:

                     Class: I Class: P Class: S
Sensitivity            0.5923   0.7237   0.4978
Specificity            0.7746   0.8420   0.7896
Pos Pred Value         0.5726   0.6933   0.5403
Neg Pred Value         0.7884   0.8606   0.7599
Prevalence             0.3377   0.3304   0.3319
Detection Rate         0.2000   0.2391   0.1652
Detection Prevalence   0.3493   0.3449   0.3058
Balanced Accuracy      0.6834   0.7828   0.6437
```

Interpretation: Classification is summarized in the confusion matrix output as prediction and reference values which are the same as the prediction output in the previous discussion.

Accuracy : Model accuracy is 60.43 % meaning that the model was successful in correctly predicting 60.43% of all instances. The kappa value of 0.40 indicates a fair level of agreement beyond what can be obtained by chance. The NIR or the no information rate, the case in which the model just predicts the most prevalent class is 33.77% (All our classes have equal representation in our dataset).

Precision (Positive Predictive Value) : This parameter provides the proportion of correct positive predictions to all the positives predicted for a class by this classifier. It is compiled using the formula Precision = True Positives / (True Positives + False Positives). In the confusion Matrix results we find that this model's precision rate is 57.26% for I class, 69.33% for P class and 54% for S class. These values are not high indicating that the precision of this model can be improved using different techniques such as feature selection.

Recall or Sensitivity (True Positive Rate): This parameter measures the proportion of correct positive predictions out of all actual positive instances of the class. It is given by the formula Recall = True Positives / (True Positives + False Negatives). In the confusion matrix results above, the recall rate is

59.23% for I class, 72.37% for P class and 49.78% for S class. This model has better values for recall than precision indicating that many of the positives this model identifies are actually negatives.

Specificity or True Negative Rate: Specificity is a measure of the classifier's ability to correctly exclude instances not belonging to a class. It answers the question, of all the actual negative instances for a class in the dataset, how many did this classifier predict as negatives. It is the ratio of true negatives to the total number of actual negative cases and is given by Specificity= True Negatives / (True Negatives + False Positives). In the confusion matrix results we can see that specificity of I class is 77.46%, for P class it is 84.2% and for S class it is 78.96%.

F1 measure : The F1 measure is the harmonic mean of precision and recall and is a single metric that combines precision and recall into a single score. It places equal importance on both Precision (the ability to make accurate positive predictions) and recall(the ability to capture all actual positive instances) and is used to provide a balanced assessment of the model's performance. The F1 measure is particularly useful when there is an uneven cost associated with false positives and false negatives. It emphasizes on the trade-off between precision and recall and is more important when a dataset is imbalanced. In our case, we have exactly the same number of class instances or samples for each class in our dataset and therefore there is no class imbalance that could skew the evaluation.

We calculated F1 using the formula, F1 = 2*(Precision * Recall)/(Precision + Recall) and obtained the values 0.5822834 for I class, 0.7081739 for P class and 0.51818 for S class. A higher F1 score indicates better performance for that class. Based on the scores, we can conclude that our model is best at predicting P class. Then it is better at predicting I class and then S class.

```
> ## CALCULATING F1 MEASURE
> #F1 <- 2*(Precision * Recall)/(Precision + Recall)
> #Precision
> p.i <- 0.5726
> p.p <- 0.6933
> p.s <-0.5403
> #Recall
> r.i <- 0.5923
> r.p <- 0.7237
> r.s <- 0.4978
> f1_i <- 2*(p.i * r.i)/(p.i + r.i)
> f1_i
[1] 0.5822834
> f1_p <- 2*(p.p * r.p)/(p.p + r.p)
> f1_p
[1] 0.7081739
> f1_s <- 2*(p.s * r.s)/(p.s + r.s)
> f1_s
[1] 0.51818
```

1.b. Kernels

We used the 'rbf' kernel to train a non-linear SVM model and made predictions using the test data. We used the same train and test datasets for the linear and non-linear models.

```
> svm_classifier_rbf <-ksvm(Occupancy.Status~., data = train, kernel = "rbfdot")
> svm_classifier_rbf
Support Vector Machine object of class "ksvm"

SV type: C-svc  (classification)
 parameter : cost C = 1

Gaussian Radial Basis kernel function.
 Hyperparameter : sigma =  0.00793768258980492

Number of Support Vectors : 1872

Objective Function Value : -734.639 -959.9742 -787.2531
Training error : 0.300966
```

The model used 1872 support vectors and misclassified 30% of the training data points. Essentially, when using the non-linear kernel, the support vectors used were more and the training error was less. However when predictions were made using the non-linear kernel, we found that the accuracies were slightly poorer..

```
> ##PREDICTING WITH RBF CLASSIFIER
> svm_predictions_rbf <- predict(svm_classifier_rbf, test)
> agreement_rbf <-svm_predictions_rbf == test$Occupancy.Status
> table(agreement_rbf)#Number of TRUE is more
agreement_rbf
FALSE  TRUE
  277   413
> prop.table(table(agreement_rbf))#TRUE percentage increased
agreement_rbf
    FALSE      TRUE
0.4014493 0.5985507
```

Interpretation : The agreement table summarizes the correct and incorrect predictions made by the classifier using the non-linear kernel. We find that the classifier correctly classified 413 instances (vs 417 for the linear model) and incorrectly classified 277 instances(273 for the linear model), which is given by 59.85% correct predictions and 40.14% of incorrect predictions out of the total 690 observations in the test dataset which is slightly lower compared to the linear kernel model.

We visualized the confusion matrix for analyzing various other parameters of the non-linear model.

```
> confusion_matrix_rbf <- confusionMatrix(svm_predictions_rbf, test$Occupancy.Status)
> confusion_matrix_rbf
Confusion Matrix and Statistics

          Reference
Prediction   I   P   S
         I 130  20  58
         P  39 163  51
         S  64  45 120

Overall Statistics

               Accuracy : 0.5986
                 95% CI : (0.5609, 0.6354)
    No Information Rate : 0.3377
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.398

 Mcnemar's Test P-Value : 0.07895

Statistics by Class:

                     Class: I Class: P Class: S
Sensitivity            0.5579   0.7149   0.5240
Specificity            0.8293   0.8052   0.7636
Pos Pred Value         0.6250   0.6443   0.5240
Neg Pred Value         0.7863   0.8513   0.7636
Prevalence             0.3377   0.3304   0.3319
Detection Rate         0.1884   0.2362   0.1739
Detection Prevalence   0.3014   0.3667   0.3319
Balanced Accuracy      0.6936   0.7601   0.6438
```

Interpretation: The model could predict 130 instances of I class as I (138 for linear model) , 163 instances of P class as P(165 for linear model) and 120 instances of S class as S(114 for linear model). The agreement table summarizes the count of correct predictions as 413 and incorrect predictions as 277 out of the total number of 690 observations in the test dataset. Classification is summarized in the confusion matrix output as prediction and reference values.

Accuracy : Model accuracy is 59.86 % meaning that the model was successful in correctly predicting 59.86% of all instances. The kappa value of 0.398indicates a fair level of agreement beyond what can be obtained by chance. The NIR or the no information rate, is the same as in the linear model.

Precision (Positive Predictive Value) : This parameter provides the proportion of true positive predictions to all the instances predicted for the class. In the confusion Matrix results we find that this model is correctly predicting 62.5% of all the predicted class instances of I as I, 64.43% of all the predicted class instances of P as P and 52.4% of all the predicted class instances of S as S.

Recall or Sensitivity (True Positive Rate): This parameter measures the proportion of true positive predictions out of all actual instances of the class. In the confusion matrix results above, from the actual instances of the I class, the model is able to predict 55.79% of the instances correctly. From the actual

instances of P class it is able to predict 71.49% correctly and from the actual instances of S class, it is able to predict 52.4% of the instances correctly.

Specificity or True Negative Rate: Specificity is a measure of the classifier's ability to correctly exclude instances not belonging to a class. It is the ratio of true negatives to the total number of actual negative cases. In the confusion matrix results we can see that specificity of I class is 82.93%, for P class it is 80.52% and for S class it is 76.36%.

F1 measure : The F1 measure is the harmonic mean of precision and recall and is a single metric that combines precision and recall into a single score. We calculated F1 using the formula, F1 = 2*(Precision * Recall)/(Precision + Recall) and obtained the values 0.589547 for I class, 0.677766 for P class and 0.524 for S class. A higher F1 score indicates better performance for that class. Based on the scores, we can conclude that our model is best at predicting P class, but has improved in predicting S class from the linear model.

Summary:

Model Accuracy : Minor reduction in Model prediction accuracy noticed from 0.6043 to 0.5986 from linear to non-linear SVM models. The NIR remains the same between the two models, kappa value reduced marginally from 0.4065 to 0.398.

We have compiled a table to summarize the results from the confusion matrices of the two models.

Class	Precision		Recall		Specificity		F1 measure	
	linear	non-linear	linear	non-linear	linear	non-linear	linear	non-linear
I	0.5726	0.625	0.5923	0.5579	0.7746	0.8293	0.582283	0.589547
P	0.6933	0.6443	0.7237	0.7149	0.842	0.8052	0.708174	0.677766
S	0.5403	0.524	0.4978	0.524	0.7896	0.7636	0.51818	0.524

Interpretation : From the table above, we can see that the precision, specificity and F1 values for I class have improved from linear to non-linear models, whereas all the 4 parameters have decreased from linear to non-linear for the P class. For the S class only recall and F1 measure values have increased while other values have decreased. Overall, the non-linear kernel does not show significant improvement in performance over the linear kernel for our dataset, however it is better at predicting S class than the linear model.

Question 2. Neural Networks

We intend to explore Neural Networks to find out if they offer more accurate models for predicting the 'Original Interest Rate' of a loan from the loan dataset, and also find out the importance of the predictive features. We repeat the research questions to which we may find better answers from this part of the analysis.

1.'Can we accurately predict the 'Interest Rate' applied to a mortgage loan by a lending institution, using criterion such as the Quantum of loan, Loan-To-Value ratio, Debt-To-Income ratio, Loan Purpose, Number of Borrowers and Credit Score of the Borrower/s, using the Fannie Mae Acquisition and Performance 2022Q4 dataset?'

2.'Do all the borrower criteria such as the Quantum of loan, Loan-To-Value ratio, Debt-To-Income ratio, Loan Purpose, Number of Borrowers and Credit Score of the Borrower/s have equal influence over 'Interest Rate'? If not, which criteria have more effect on Interest Rate?'

Note : The answers to the above questions from Milestone-2 analysis were that the variables in the Fannie Mae dataset were not having adequate predictive power. We therefore added bondRate, fedFundsRate and benchMarkMortgageRate variables. For MS3 we dropped benchMarkMortgageRate as it had almost perfect negative correlation with fedFundsRate variable and therefore will not add value. From MS-2 analysis, we found that the most predictive features are bondRate and fedFundsRate followed by Borrower Credit score and Loan quantum.

IVs and DV:

S.No	**1**	2	3	4	5	6	7	8	9	10	11	12	13	14
Variable Name	**Original.Interest.Rate**	bondRate	fedFundsRate	Original.UPB	Original.Loan.to.Value.Ratio..LTV.	Original.Combined.Loan.to.Value.Ratio..CLTV.	Number.of.Borrowers	Debt.To.Income..DTI.	Borrower.Credit.Score.at.Origination	Property.State	Seller.Name	Property.Type	Loan.Purpose	Occupancy.Status
Data type	**Numeric (DV)**	Numeric (IV)	Numeric (IV)	Numeric (IV)	Numeric (IV)	Numeric (IV)	Numeric (IV)	Numeric (IV)	Numeric (IV)	Nominal (IV)	Nominal (IV)	Nominal (IV)	Nominal (IV)	Nominal (IV)

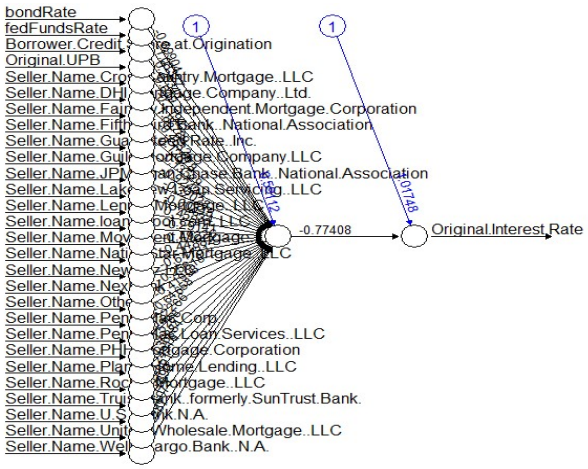
Data : To suit the SVM classifiers, we had encoded all our categorical variables except the outcome variable 'Occupancy Status'. We now have to encode that variable and normalize all the numeric variables including the DV interest rate. We achieved this with R code. And then we split the encoded dataset into training and test datasets.

```
> data <- read.csv("Normalized_MS3dataset.csv")
> dim(data)
[1] 2760   93

> train_NN <- data[1:2070, ]
> dim(train_NN)
[1] 2070   93
> test_NN <- data[2071:2760, ]
> dim(test_NN)
[1] 690  93
```

Approach: We wanted to find the best NN model for our dataset and then deploy hidden layers technique on it. Therefore, we first trained a model on 5 IVs, *fedfundsRate, bondRate, Original UPB, Borrower Credit Score and SellerName* that were found to be most effective in predicting interest rate both for multivariate regression analysis and Random Forest analysis and deployed the model on the test dataset to obtain the following results.

```
> model_NN_4var_SN_results <-compute(model_NN_4var_SN,test_NN[, c(2:5,10:33)])
> predicted_IR_4var_SN <- model_NN_4var_SN_results$net.result
> cor(predicted_IR_4var_SN, test_NN$Original.Interest.Rate)
          [,1]
[1,] 0.6948693
> mse <- mean((predicted_IR_4var_SN - test_NN$Original.Interest.Rate)^2)
> mse
[1] 0.02007664
> rmse <- sqrt(mse)
> rmse
[1] 0.1416921
```

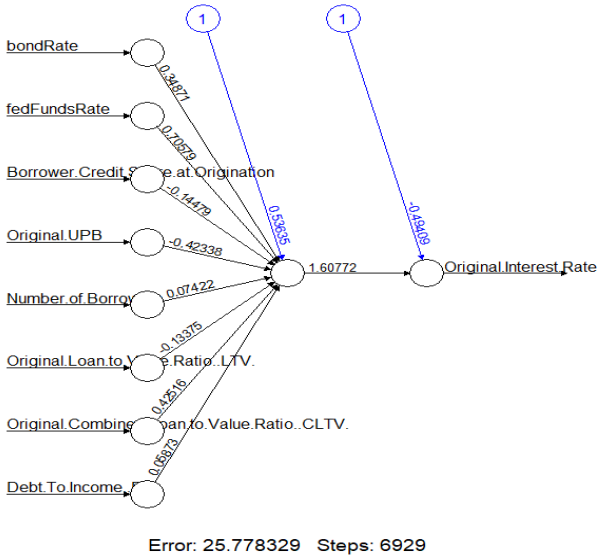
We then plotted this model. The model plot reveals one hidden layer with one neuron and the encoded variables as the number of neurons in the input layer and the neuron corresponding to the DV Original interest rate in the output layer



We then trained a model we called model _NN_numvar where we tried using *all the numerical variables in the dataset and deployed it on the test dataset to make predictions. Here are the results*

```
> model_NN_numvar_results <-compute(model_NN_numvar,test_NN[2:9])
> predicted_IR_numvar <- model_NN_numvar_results$net.result
> cor(predicted_IR_numvar, test_NN$Original.Interest.Rate)
          [,1]
[1,] 0.6598189
> mse <- mean((predicted_IR_numvar - test_NN$Original.Interest.Rate)^2)
> mse
[1] 0.02196935
> rmse <- sqrt(mse)
> rmse
[1] 0.1482206
>
```

We plotted the model to visualize the structure. *Clearly, without the Seller Name variable, the results are poorer.*



Hence we trained the model we called model_NN_SN and deployed it on test data to make predictions and found better results.

```
> model_NN_SN_results <-compute(model_NN_SN,test_NN[2:33])
> predicted_IR_SN <- model_NN_SN_results$net.result
> cor(predicted_IR_SN, test_NN$Original.Interest.Rate)
          [,1]
[1,] 0.7039427
> mse <- mean((predicted_IR_SN - test_NN$Original.Interest.Rate)^2)
> mse
[1] 0.01958945
> rmse <- sqrt(mse)
> rmse
[1] 0.1399623
>
```

Finally we trained a model using all the variables, that is 92 encoded IVs which we called model_NN and captured the model output to a text file. In this model, we first used the neuralnet() function from the neuralnet package without specifying the 'hidden' parameter and used all the variables in our encoded dataset. We have 92 IVs and 1 DV interest rate after encoding all levels of the 5 categorical variables in the dataset.

```
$call
neuralnet(formula = Original.Interest.Rate ~ ., data = train_NN)

$response
  Original.Interest.Rate
1             0.50000000
2             0.47222222
3             0.44444444
4             0.44444444
5             0.80555556
6             0.91666667
7             0.43333333
8             0.52555556
```

The model output was captured into a text file and interpreted, as it was very long to view in the R studio environment for interpretation.

Some important elements found in the Model_NN outputs were:

1. $response : The prediction values output by the model based on training data.

```
$call
neuralnet(formula = Original.Interest.Rate ~ ., data = train_NN)

$response
  Original.Interest.Rate
1             0.50000000
2             0.47222222
3             0.44444444
4             0.44444444
5             0.80555556
6             0.91666667
7             0.43333333
8             0.52555556
9             0.75000000
```

2. $net.result : Calculated results of each neuron in the output layer of the neural network. In a regression problem like ours with a single DV, we have a single column in this matrix.

```
$net.result
$net.result[[1]]
          [,1]
1    0.6205681
2    0.4621687
3    0.4709806
4    0.5179645
5    0.7138083
6    0.8071609
7    0.5003152
8    0.5720053
9    0.7560592
10   0.6159259
11   0.3892620
12   0.5676266
```

3. $Weights : The weights object provides information about the weights which are the connections between the neurons and the biases, also known as offsets for each layer. In this model, we have 92 weights associated with each neuron in the input layer, one weight associated with the neuron in the hidden layer, totalling 93 weights. There is one bias term associated with the neuron in the hidden layer for this model.

```
$weights
$weights[[1]]
$weights[[1]][[1]]
              [,1]
 [1,]  1.748430797
 [2,] -0.840026267
 [3,] -1.205262373
 [4,]  0.589900481
 [5,]  0.584734760
 [6,] -0.069904155
 [7,]  0.403189197
 [8,] -1.615515820
 [9,] -0.007075371
[10,] -0.678688237
[11,]  2.055960700
[12,] -0.633071354
[13,]  0.106802548
```

4. $Startweights : The startweights represent the strength of connections between the input layer and output layer. These are the initial weights for each of the 92 neurons that are randomly assigned by the model. As the model trains, it adjusts the weights through back propagation to minimize the error function and make better predictions for the task on hand.

```
$startweights
$startweights[[1]]
$startweights[[1]][[1]]
              [,1]
 [1,]  2.417174507
 [2,] -0.481722623
 [3,] -0.418759641
 [4,]  1.278793403
 [5,] -1.343225813
 [6,]  0.077453309
 [7,]  1.016275415
 [8,]  0.471444378
 [9,]  0.845016206
```

5. Error function $err.fct : It is the error function that calculates the SSE, sum of squared errors for a regression model. The goal during training is to minimize this function so that predicted values are closer to actual values. *Upon deploying the values for the above model we obtained the value of 11.4204 for the sse. This was calculated using R code.*

> model_NN$err.fct

function (x, y) { 1/2 * (y - x)^2 }

<bytecode: 0x000002a0d4ce52d8>

<environment: 0x000002a0d4815ae8> attr(,"type") [1] "sse"

6. Activation function : The activation function in the model_NN output shows that the model used a linear sigmoid function to introduce non-linearity to the model. The logistic sigmoid function maps any input value to a value between 0 and 1 to suit binary classification. *The value of the activation function depends on the weighted input to each neuron. For a value of 'x' for weighted sum, the activation function is given by 1/(1 + exp(-x))*

> model_NN$act.fct

function (x) { 1/(1 + exp(-x))}

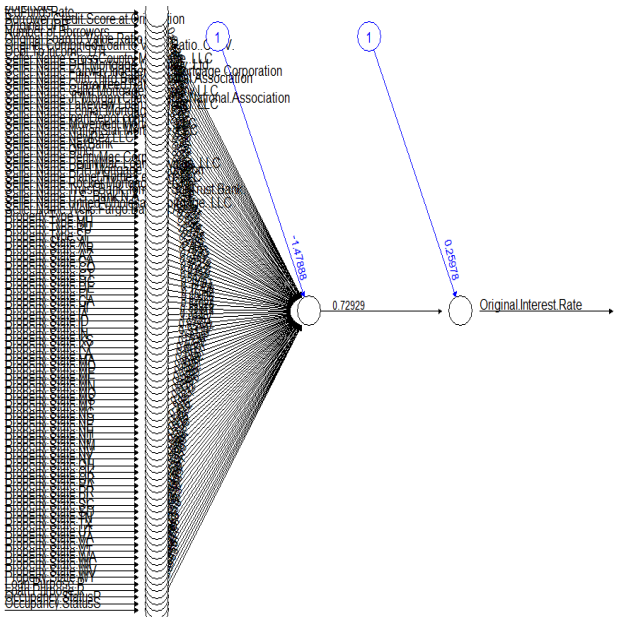
<bytecode: 0x00000134172e0fa0>

<environment: 0x00000134172e1908>

attr(,"type")

[1] "logistic"

We visualized the structure of the neural network by plotting it. The model created a single layer neural network with 92 input neurons corresponding to each IV in our encoded dataset. According to this article, on training neural networks by Guenther and Fritsch (journal.r-project.org archives, 2010-11) the default for the neuralnet() function when no hidden layers are specified is one hidden layer with one neuron. This is evident from the plot where the 92 neurons in the input layer connect to the one neuron in the hidden layer which is again connected to one neuron in the output layer.



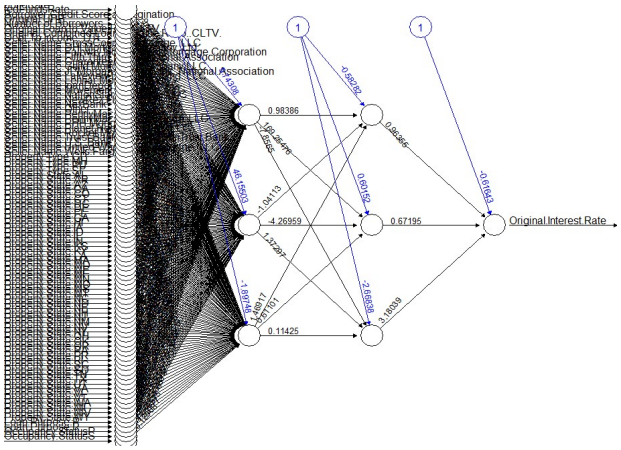
We then deployed the trained model_NN to predict our outcome variable using the test data and better results than any other model we trained so far.

```
> model_NN <- neuralnet(formula = Original.Interest.Rate~.,data = train_NN)
> model_NN_results <-compute(model_NN,test_NN[2:93])
> predicted_IR <- model_NN_results$net.result
> cor(predicted_IR, test_NN$Original.Interest.Rate)
          [,1]
[1,] 0.7575084
> mse <- mean((predicted_IR - test_NN$Original.Interest.Rate)^2)
> mse
[1] 0.0165513
> rmse <- sqrt(mse)
> rmse
[1] 0.1286519
>
```

Since we have identified the best performing model, we proceeded to deploy it by specifying different values for the number of hidden layers. The first variant with 2 hidden layers with 3 neurons each which we called model_NN_H2, provided inferior results to the default model with 1 hidden layer and one neuron

```
> ###### USING HIDDEN LAYERS=2 specified by c(3,3)###############
> model_NN_H2 <- neuralnet(formula = Original.Interest.Rate~.,data = train_NN, hidden=c(3,3))
> model_NN_H2_results <- compute(model_NN_H2, test_NN[2:93])
> predicted_IR_H2 <- model_NN_H2_results$net.result
> cor(predicted_IR_H2, test_NN$Original.Interest.Rate)
          [,1]
[1,] 0.6426574
> mse <- mean((predicted_IR_H2 - test_NN$Original.Interest.Rate)^2)
> mse
[1] 0.0244129
> rmse <- sqrt(mse)
> rmse
[1] 0.1562463


> residuals <- predicted_IR_H2 - test_NN$Original.Interest.Rate
> sse <- sum(residuals^2)
> sse
[1] 16.8449
>
```

Plotting model_NN_H2



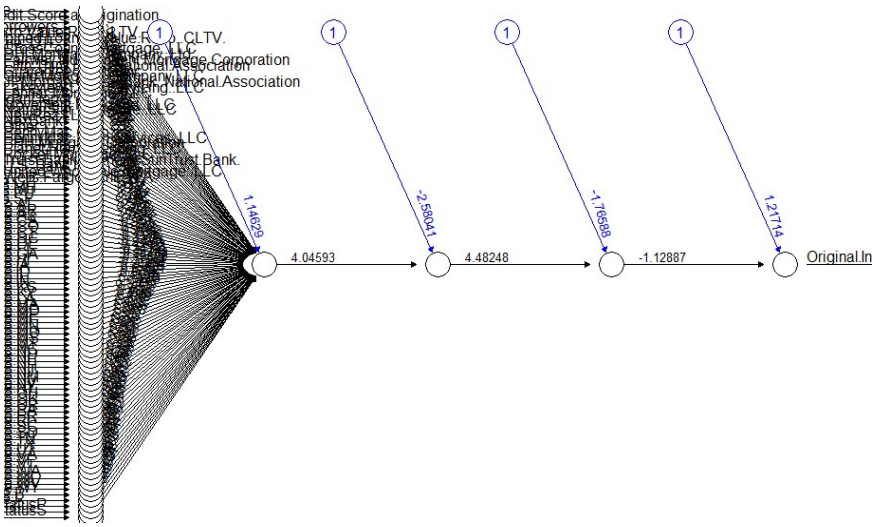
We produce hereunder, the summary of performance results. *There is one neuron in the Output layer for all the models which is not shown separately here.*

Default Model_Name: model_NN No of IVs 92 (encoded and normalized 13 variables), Performance with different hidden layers									
No of hidden layers and neurons in each layer specified using hidden parameter in the neuralnet() function	Correlation bet. predicted & actual values	RMSE	error function (SSE)	No of neurons in					
				input layer	hidden layer 1	hidden layer 2	hidden layer 3	hidden layer 4	hidden layer 5
Default (1 hidden layer and 1 neuron)	0.7575084	0.1286519	11.4204	92	1	-			
hidden layers 2, with 3 neurons in each layer specified by c(3,3)	0.6426574	0.1562463	16.8449	92	3	3			
hidden layers 3, with 3 neurons in first hidden layer, 2 neurons in the second hidden layer and 1 neuron in the 3rd hidden layer specified by c(3,2,1)	0.6606546	0.1513675	15.80936	92	3	2	1		
hidden layers 3, with 3 neurons in first two layers and 2 neuron in third hidden layer, specified by c(3,3,1)	0.7079605	0.1406101	13.64214	92	3	3	1		
hidden layers 3, with 2 neurons in the first and one each in 2nd and 3rd hidden layer specified by c(2,1,1)	0.6828302	0.14577	14.66173	92	2	1	1		
hidden layers 3, with one neuron in each hidden layer, specified by c(1,1,1)	**0.7575341**	**0.1286459**	**11.41934**	**92**	**1**	**1**	**1**		
hidden layers 4, with one neuron per layer, specified by c(1,1,1,1)	0.7574792	0.1286595	11.42176	92	1	1	1	1	
hidden layers 5, with one neuron per layer, specified by c(1, 1,1,1,1)	0.7574216	0.1286724	11.42404	92	1	1	1	1	1

Our Best performing model:

This model is the one with 3 hidden layers, with one neuron in each hidden layer. In this model there are 97 neurons in all, 92 in the input layer, 3 in the hidden layers and one in the output layer. There are 96 weights associated with the neurons and three bias terms for the layers in the hidden neurons. The error function is minimal at 11.41934 and the correlation between predicted and actual values of the outcome variable on unseen test data is 0.7575341. The RMSE value is also minimal at 0.1286459.

We plotted this model to obtain this structure.



Question 3. Clustering

Our dataset comprises 9 numeric and 5 nominal variables. Therefore we used Approach 2 on the dataset and computed Gower distances to proceed to perform the PAM clustering, hierarchical clustering and DBSCAN clustering on the dataset.

1.PAM clustering method: Partitioning around medoids method is a clustering technique that uses medoids to define cluster centers. On our data, the algorithm begins by assigning medoids initially randomly and then assigning each data point to the nearest medoid based on the Gower distance as our dataset is of mixed data types. It then recalculates the medoid so as to minimize the dissimilarity or the distance between the other points in the cluster after adding the datapoint. It continues the process till all data points are assigned to one cluster or the other.

We used the dissimilarity matrix from the daisy function, (an acronym for Day's Euclidean, Manhattan, and Canberra DisSimilarity) as input to the pamk() function, which automatically clusters the dataset around the optimal number of clusters.

1. We found these values for our PAM clusters.

```
> # calculating GOWER DISTANCE
> g.dist = daisy(data, metric="gower", type=list())
> #  Using Gower distance matrix, determining optimal number of clusters
> pc <- pamk(g.dist, krange = 1:5, criterion = "asw")
> optimal_clusters <- pc$nc
> optimal_clusters
[1] 2
>
```

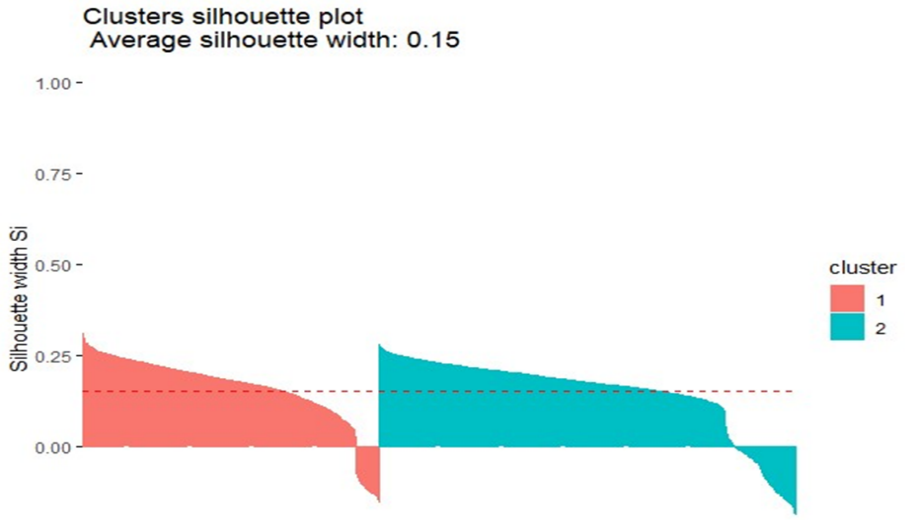
Interpretation: The best number of clusters using PAM clustering method is 2 for our dataset

2. We then proceeded to understand the number of elements in each cluster.

```
> # Performing PAM clustering with the optimal number of clusters
> pam_result <- pam(g.dist, k = optimal_clusters)
> #number of elements percluster
> cluster_counts <- table(pc$pamobject$clustering)
> print(cluster_counts)

   1    2
1145 1615
>
```

Interpretation: We find that in Cluster-1 we have 1145 elements and in cluster-2 we have 1615 elements for the total 2760 observations in our dataset.

3. Structural plots: PAM clusters do not inherently produce structural plots, therefore we visualized silhouette plots that show the clusters returned by the pam() function.



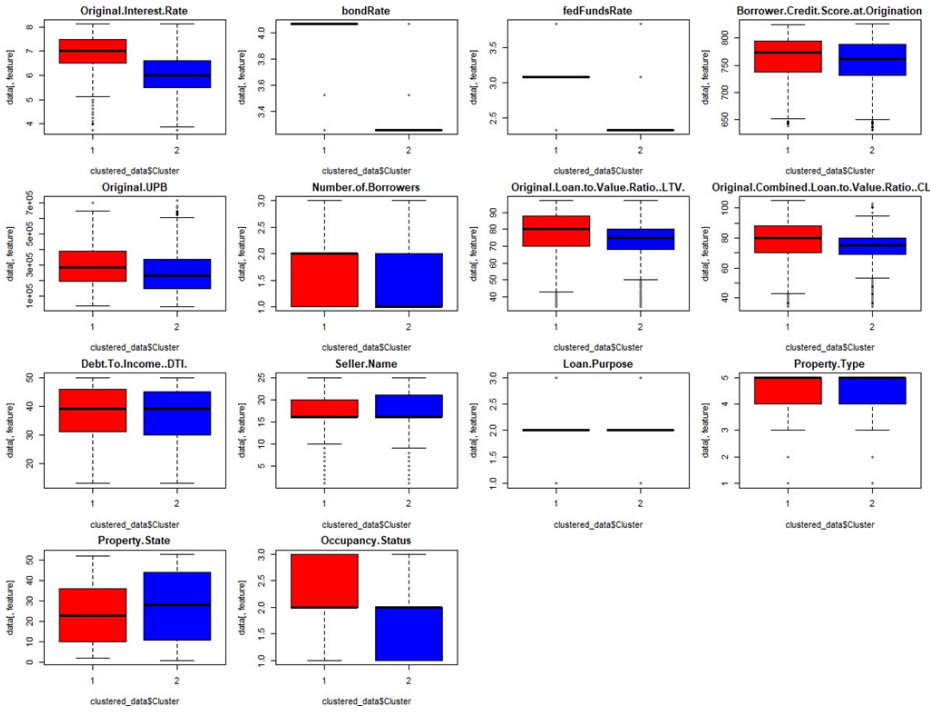
Interpretation : The algorithm could iteratively bring together data points calculating and recalculating medoids to arrive at two distinct clusters, one smaller than the second cluster with a larger number of elements.

4. a. Computing mean values for the features in the dataset for comparison. Also comparing the means for the DV Original interest rate in both the clusters.

```
> ##### COMPARING CLUSTERS, CALCULATING CLUSTER MEANS
> # Accessing cluster assignments
> cluster_assignments <- pam_result$clustering
> # Combine original data and cluster assignments
> clustered_data <- cbind(data, Cluster = cluster_assignments)
> # Calculating mean values for each cluster
> cluster_means <- aggregate(. ~ Cluster, data = clustered_data, FUN = mean)
> # Print the cluster means
> print(cluster_means)
  Cluster Original.Interest.Rate bondRate fedFundsRate
1       1               6.913827 3.930166     3.131747
2       2               6.090871 3.347771     2.525046
  Borrower.Credit.Score.at.Origination Original.UPB Number.of.Borrowers
1                             763.2568     300711.8            1.572926
2                             756.6173     255419.2            1.389474
  Original.Loan.to.Value.Ratio..LTV. Original.Combined.Loan.to.Value.Ratio..CLTV.
1                           75.83319                                      75.92926
2                           73.55232                                      73.76285
  Debt.To.Income..DTI. Seller.Name Loan.Purpose Property.Type Property.State
1             37.55459    16.29607     1.914410      4.172926       23.06463
2             36.93189    16.19505     1.853251      4.208669       26.90526
  Occupancy.Status
1         2.292576
2         1.792570
>
```

Interpretation: Comparing means of each feature, we can see that the means are higher for Cluster -1 and significantly lower for Cluster-2 . For our original DV Original Interest Rate, the means differ in the two clusters by a significant value of 0.82%.

4. b. Creating box plots for each feature to compare and contrast.

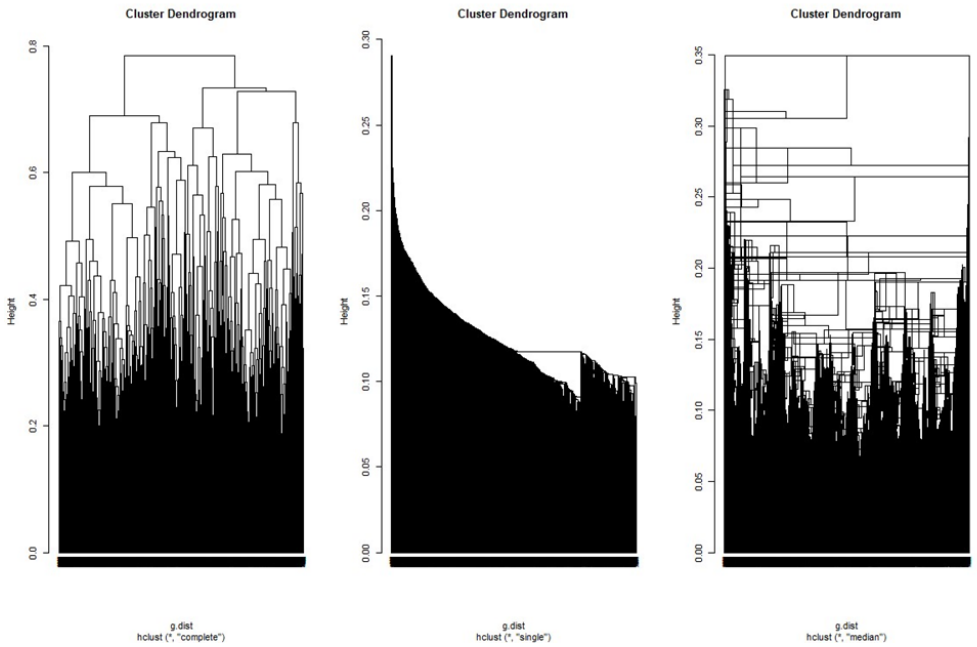


Interpretation : We find that while some features such as Original Interest Rate, Occupancy status, bondRate, fedFundsRate have more distinctly different means, other features such as DTI, Number of Borrowers and Property type variables have closer means between the clusters.

2. Hierarchical clustering : There are two approaches to hierarchical clustering, which basically is an algorithm to iteratively put together all the data points in the dataset into a single cluster by following certain rules to do the same, which are specified in the method of linkage. These are bottom-up (agglomerative) or top-down(divisive) approaches. We are using the agglomerative approach here. This technique involves grouping similar data points iteratively into clusters in a tree-like structure called a dendrogram. After computing dissimilarities or the distances between data points using an appropriate function, in our case, we have the Gower distances calculated using the daisy function, we can choose the linkage method to link the data points such as 'complete', 'single', 'average', median, 'Ward's' etc. These methods determine how the distance between clusters are calculated when they are being brought together.

1. Structural Images of hierarchical clustering with different linkage methods:

'Complete' linkage method 'single' linkage method 'median' linkage method

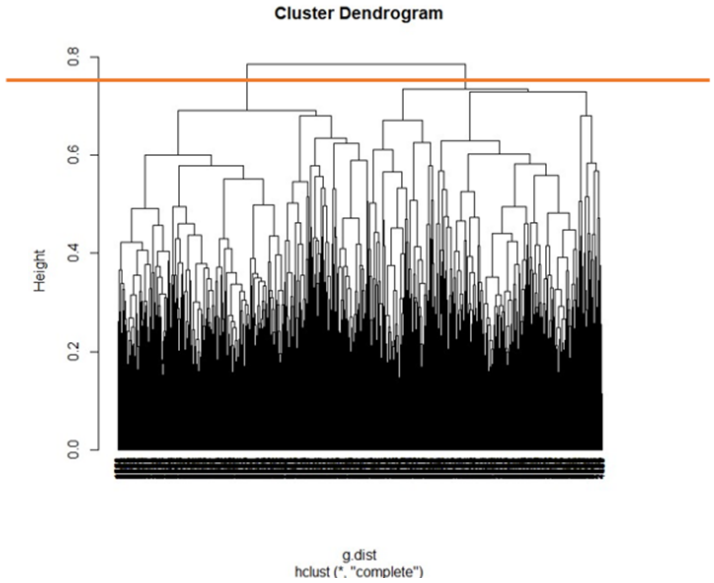


The Complete Linkage method: In this method, we define the distance between clusters to be the maximum distance between any two points in the clusters. This method tends to produce compact, spherical clusters that are sensitive to outliers.

The Single Linkage: In this method the minimum distance between two clusters is defined as the minimum distance between any two points in the clusters. The effect is that this option tends to produce elongated clusters as can be seen in the image. This method is more sensitive to noise and outliers.

The Median linkage : In this method, the distance between two clusters is defined as the median distance between all pairs of points in the clusters. This method balances the effects of complete and single linkage and is less sensitive to outliers.

2. Determination of optimal number of clusters through visual inspection of the dendrogram. Taking up the dendrogram derived from the complete method of linkage, for optimal clusters, in our case, visual inspection reveals that the tree shows a significant increase in height after the root node at the second level. Therefore, we cut our dendrogram to height = 0.75 to obtain 2 clusters.



3. Counting elements in each cluster :

```
> clusters <- cutree(hc.c, h = 0.75)
> clusters_elements <- table(clusters)
> print(clusters_elements)
clusters
   1    2
1330 1430
>
```

Interpretation: The number of elements in the clusters are 1330 and 1430 respectively, which are different from those obtained from the PAM clustering technique.

4. *Comparing features from clusters, specifically our original DV Original Interest Rate:*

```
> clusters <- cutree(hc.c, h = 0.75)
> # Combining original data and clusters
> clustered_data <- cbind(data, Cluster = clusters)
> # Calculating mean values for each cluster
> cluster_means <- aggregate(. ~ Cluster, data = clustered_data, FUN = mean)
> # Print the cluster means
> print(cluster_means)
  Cluster Original.Interest.Rate bondRate fedFundsRate
1       1              6.993960 3.942917     3.253684
2       2              5.909875 3.260566     2.333147
  Borrower.Credit.Score.at.Origination Original.UPB Number.of.Borrowers
1                             759.9406     277218.0            1.464662
2                             758.8427     271410.5            1.466434
  Original.Loan.to.Value.Ratio..LTV.
1                           75.10075
2                           73.93846
  Original.Combined.Loan.to.Value.Ratio..CLTV. Debt.To.Income..DTI. Seller.Name
1                                     75.18797             37.54211    16.31353
2                                     74.17203             36.86294    16.16573
  Loan.Purpose Property.Type Property.State Occupancy.Status
1     1.897744      4.231579       24.56165         2.028571
2     1.860839      4.158741       26.00979         1.973427
>
```

Interpretation: Just as in the case of PAM clustering, we find that for numerical variables, Cluster-1, the means are higher and for Cluster-2 they are all lower, whereas for categorical variables such as Property state, it is not the case. The mean for the DV Original Interest rate in Cluster-1 here is 6.993960 whereas in PAM technique it was 6.913827 and for cluster-2 the mean from hierarchical cluster is 5.90 whereas for PAM it was 6.09. The difference in the mean for Original Interest Rate between two clusters in hierarchical clusters is 1.09% whereas it was 0.82% in PAM clustering. The larger difference in means of Original Interest Rate between clusters in hierarchical clustering indicates that this method has identified more distinct groups based on the overall characteristics of the dataset.

3. .DBSCAN clustering technique: In this technique, the algorithm performs density-based scanning and clustering based on the parameters we provide. We must provide values for two parameters namely 'epsilon' and 'minimum points' to the algorithm. It then works by randomly selecting a starting point and scans around it for points within epsilon distance to itself. These points that are within the epsilon of the starting point are known as core points. With the minimum number of core points including the starting point, a cluster is formed. The algorithm output is fine-tuned using different epsilon values and minimum points to obtain the desired number of clusters.

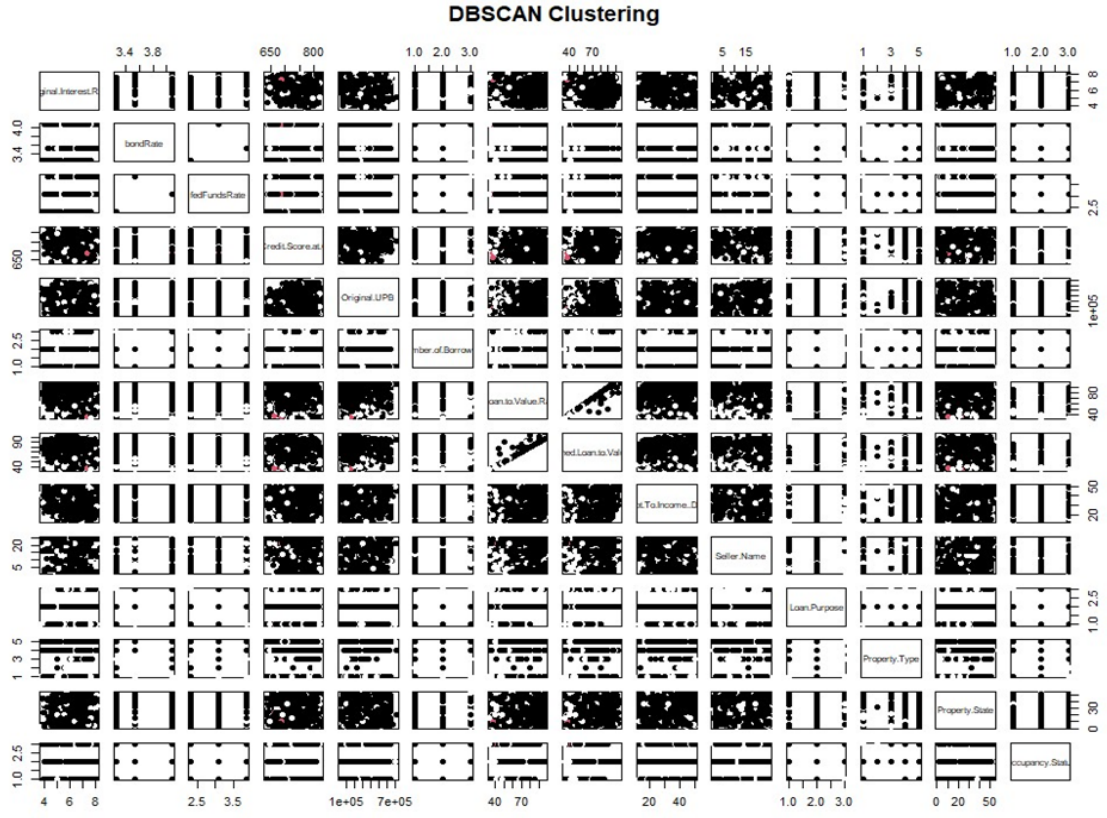
1. *Optimal clusters and number of elements in each cluster :* Selecting optimal clusters in this algorithm involves identifying epsilon and minimum points that reflect desired number of clusters and minimum noise points.

```
  2   94 1321    0
> dbc2 = dbscan(g.dist, eps=.18, minPts= 7) ###2 FINAL
> dbc2
DBSCAN clustering for 2760 objects.
Parameters: eps = 0.18, minPts = 7
Using unknown distances and borderpoints = TRUE
The clustering contains 2 cluster(s) and 174 noise points.

   0    1    2
 174 2581    5

Available fields: cluster, eps, minPts, dist, borderPoints
>
```

Interpretation: After about 50 attempts, we finalized the epsilon value of 0.18 with minimum points 7, to arrive at the DBScan clusters with the least number of noise points of 174. In all other cases, the noise points were at high levels.

2. The scatter plot matrix or pairs plot form the DBscan algorithm: We have 14 features in our dataset and therefore, the DBSCAN plot returns a matrix of scatter plots with each individual feature name as the diagonal of the matrix. Each scatter plot within the matrix structure represents the relationship between a pair of features in the dataset. The diagonal plots show the distribution of each individual feature whereas the off-diagonal plots identify trends or clusters in the relationship between pairs of features.

   Interpretation of the plot: The points in the scatter plot are ideally colored based on their cluster assignments. Here for the most part we see only one color, (there are a few other colors of course), indicating that most of the points fall in the same cluster.



4. Comparing means of individual features in the DBSCAN clusters: Since there are noise points in the clusters returned by the DBScan algorithm, the following output shows means for the noise points also.

```
> ##COMPUTING MEANS FOR FEATURES IN THE CLUSTER
> dbscan_clustered_data <- data.frame(data, Cluster = dbc2$cluster)
> # CalculatING mean values for each cluster
> dbscan_cluster_means <- aggregate(. ~ Cluster, data = dbscan_clustered_data, FUN = mean)
> # Display the cluster means
> print(dbscan_cluster_means)
  Cluster Original.Interest.Rate bondRate fedFundsRate Borrower.Credit.Score.at.Origination
1       0              6.360149 3.609138     2.890345                              745.7931
2       1              6.436097 3.587117     2.768493                              760.3747
3       2              6.971000 4.070000     3.080000                              714.2000
  Original.UPB Number.of.Borrowers Original.Loan.to.Value.Ratio..LTV.
1     286540.2            1.557471                           61.36207
2     273606.4            1.458349                           75.44479
3     156200.0            2.000000                           43.20000
  Original.Combined.Loan.to.Value.Ratio..CLTV. Debt.To.Income..DTI. Seller.Name Loan.Purpose
1                                     61.69540             36.41954    15.78736     1.896552
2                                     75.59667             37.23789    16.25804     1.879117
3                                     43.20000             39.40000    21.00000     1.000000
  Property.Type Property.State Occupancy.Status
1      3.396552       23.87931          2.270115
2      4.246416       25.41534          1.979853
3      4.800000       21.80000          3.000000
> |
```

Interpretation: We can see that the means for each feature are distinct between clusters 1 & 2. For the Original interest rate variable, the mean for cluster 1 is 6.43 and for cluster-2 it is 6.97. The difference in means is 0.54% which is less than both the PAM clustering and hierarchical clustering algorithm outputs.

Comparison of clusters from different clustering techniques: We compared clustering results from the three different clustering algorithms. The legend here is hclusters_single represents clustering results with hierarchical clustering single linkage method, hclusters_complete represents complete linkage method and pam_result$clustering and dbc2$cluster represent clusters obtained from the PAM and DBSCAN algorithms.

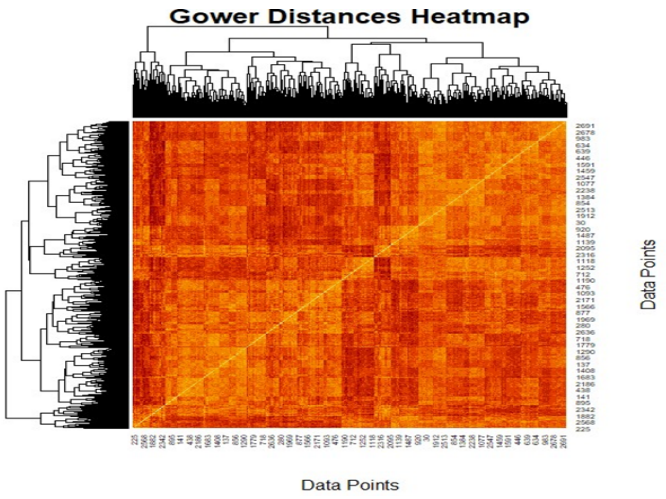
```
> # comparing the clustering results
> table(hclusters_single, hclusters_complete)
                hclusters_complete
hclusters_single    1    2
               1 1329 1430
               2    1    0
> table(hclusters_complete, pam_result$clustering)

hclusters_complete    1    2
                  1 1057  273
                  2   88 1342
> table(hclusters_single, dbc2$cluster)

hclusters_single    0    1    2
               1  173 2581    5
               2    1    0    0
> table(pam_result$clustering, hclusters_single)
   hclusters_single
      1    2
  1 1144    1
  2 1615    0
```

Interpretation:

1. The hclusters_single and hclusters_complete clusters agree on 1329 data points in their cluster-1. For cluster-2 they do not agree on any data point.
2. The hclusters_complete and pam_results$clustering results agree on 1057 data points in their cluster-1. In their cluster-2 they agree on 1342 data points.

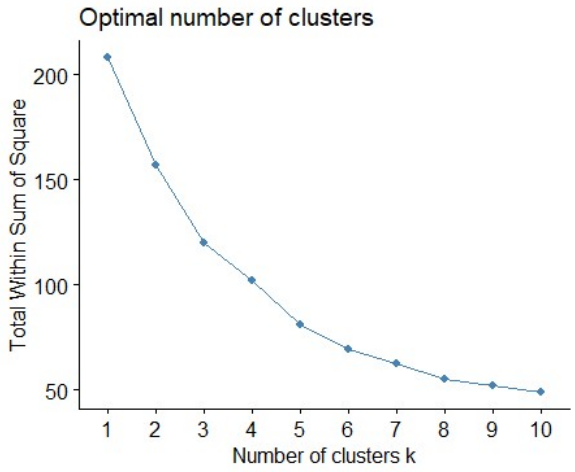
3. The hclusters_single and dbc2$cluster agree on 2581 points in their cluster-1, whereas they agree on 0 points for cluster 2.
4. The pam_results$clustering and hclusters_single results agree on 1144 data points in their cluster-1, whereas they agree on 0 points for cluster-2.

We tried to investigate the reasons for this disagreement phenomenon using the Gower distance map. Darker colors in the map indicate higher dissimilarity or distance between data points and lighter colors indicate lesser dissimilarity. We can see that the map has lighter colors than darker, indicating that the data points do not have clear dissimilarities for the algorithms to work with.



Ancillary research: We wanted to investigate if we will get better clustering results with approach-1 for our dataset. Therefore, we reduced our dataset with one unique row per property state and retained Original Interest Rate, Original Loan value, Borrower credit score and CLTV variables to deploy the kmeans(), hclust() and dbscan() algorithms. We converted the 'Property State' variable to rownames and normalized our data. The dimensions of this dataset are 53 observations of 4 variables. We first deployed the kmeans() algorithm on this dataset.

1. Optimal clusters: We plotted the 'within sum of squares' against 'the number of clusters' to determine the optimal clusters. The point at which the rate of decrease of WCSS slows down, is said to be the elbow point. For this dataset, there is no clear elbow point, however, k=2 appears to be the beginning of a decreasing trend.

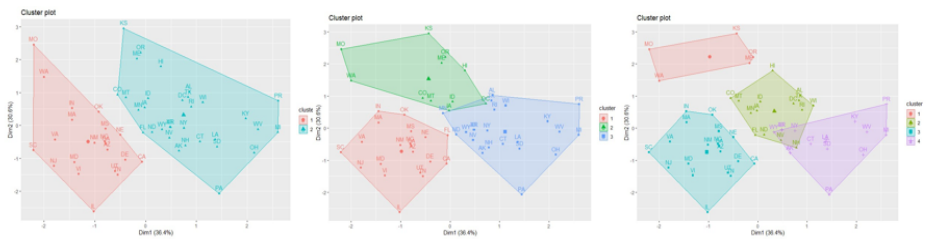


2. Checking the values for k = 2 number of clusters, we found that the tot.withinss value to be 154. This value represents the compactness of the cluster. We then proceeded to check these values for k=3 and k =4 and found that though the compactness was increasing, with tot.withinss values 119 and 97.2 respectively, the cluster visualizations showed that the clusters were overlapping.

3.

```
> k2 <- kmeans(data, centers = 2, nstart = 100)
> ### Checking the tot.withinss (sum of squared error)
> str(k2)
List of 9
 $ cluster     : Named int [1:53] 2 1 2 2 1 2 1 2 1 2 ...
  ..- attr(*, "names")= chr [1:53] "KS" "CA" "WI" "TX" ...
 $ centers     : num [1:2, 1:4] -0.853 0.56 0.759 -0.498 0.371 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:2] "1" "2"
  .. ..$ : chr [1:4] "Original.Interest.Rate" "Original.UPB" "Borrower.
Credit.Score.at.Origination" "Original.Combined.Loan.to.Value.Ratio..CL
TV."
 $ totss       : num 208
 $ withinss    : num [1:2] 63.1 91
 $ tot.withinss: num 154
 $ betweenss   : num 53.8
 $ size        : int [1:2] 21 32
 $ iter        : int 1
 $ ifault      : int 0
 - attr(*, "class")= chr "kmeans"
>
```

3. Plotting the Clusters: We used the fviz_cluster() function with different values of k, to visualize the cluster arrangements and found that when k=2, we have distinct clusters.



Interpretation: Though there are some clear differences between the clusters for values k = 3 and k =4, we decided to proceed with k=2 as it provided 2 clear clusters.

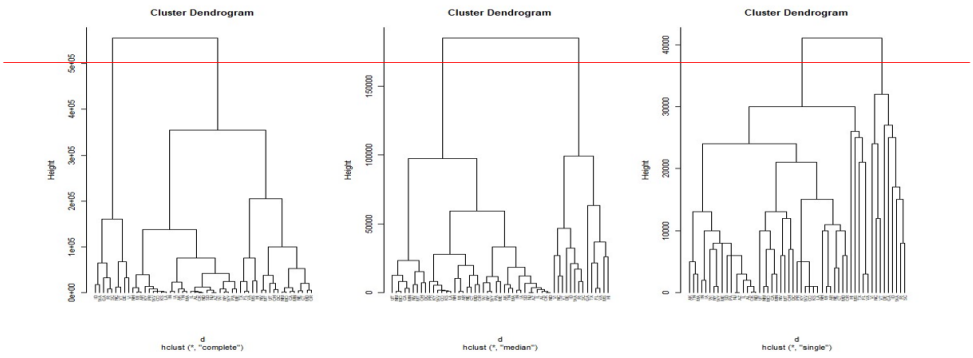
4. Comparing means: To deploy the kmeans() algorithm, we standardized our data. When we standardize data, we are transforming the variables to have a mean of '0' and a standard deviation of '1' , therefore we do not find the actual mean values of the features in this output.

```
> k2 <- kmeans(data, centers = 2, nstart = 100)
> #### OBTAINING MEAN VALUES FOR EACH FEATURE
> k_cluster_centers <- k2$centers
> # Creating a data frame with cluster centers and feature names
> k_cluster_means_df <- data.frame(Cluster = 1:nrow(cluster_centers), cluster_centers)
> print(k_cluster_means_df)
  Cluster Original.Interest.Rate Original.UPB Borrower.Credit.Score.at.Origination
1       1             -0.1119648    0.3894698                           0.3181056
2       2              0.2003581   -0.6969459                          -0.5692416
  Original.Combined.Loan.to.Value.Ratio..CLTV.
1                                    0.4994637
2                                   -0.8937772
> |
```

Interpretation : From the above output, we can conclude that the mean values of features between clusters are different from 0, and from each other, indicating that the clusters are distinct.

2. *Approach1, hierarchical clustering & DBscan* : Since we have analyzed in a detailed manner, using our original dataset, hierarchical clustering and DBscan clustering techniques, we are only briefly producing the results, retaining focus on whether or not the different clustering techniques agree with each other on this reduced dataset.

Cutting the dendrograms from visual inspection at appropriate height to obtain optimal number clusters, :



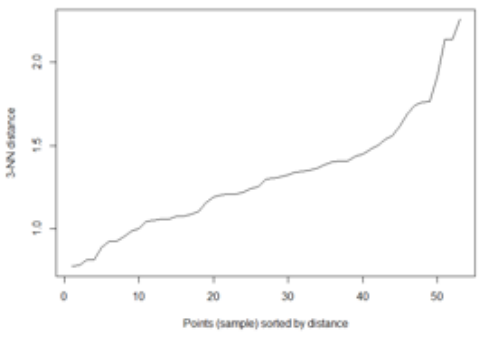
Obtaining the number of elements of each cluster from the hclust() algorithm with different methods of linkage such as complete, median and single, we find that the clusters formed from complete and single methods of linkage are similar.

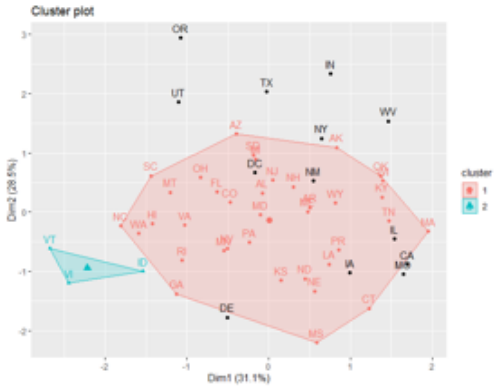
```
> print(hc.c.elements)
hc.c.clusters
 1  2
44  9
> print(hc.m.elements)
hc.m.clusters
 1  2
14 39
> print(hc.s.elements)
hc.s.clusters
 1  2
44  9
```

Calculating means: We then compared the means for each cluster formed using the 'complete' method of linkage with the hclust() function and find that the means are distinct for each feature within the cluster

```
> # Calculating mean values for each cluster
> h_cluster_means <- aggregate(. ~ Cluster, data = h_clustered_data, FUN = mean)
> h_cluster_means
  Cluster Original.Interest.Rate Original.UPB
1       1              6.450636     193250.0
2       2              6.638889     533222.2
  Borrower.Credit.Score.at.Origination
1                             746.1818
2                             774.8889
  Original.Combined.Loan.to.Value.Ratio..CLTV.
1                                     76.25000
2                                     80.55556
>
```

DBScan algorithm, visualizing elbow point for optimal epsilon, we find that there is no clear elbow. We tested with different values to find the best value of epsilon at 1.25 and minimum points at 3.





Comparing clustering results: The legend here is hc.c.clusters represents clustering results with hierarchical clustering complete linkage method, hc.m.clusters represents median linkage method and hc.s.clusters represents single linkage method of assigning data points to different clusters. So also, k2clusters are the clustering assignment of data points by the k2means algorithm and db3cluster is the clustering assignment by the DBSCAN algorithm..

```
> ##comparing clustering results
> table(hc.c.clusters, hc.s.clusters)
             hc.s.clusters
hc.c.clusters  1  2
            1 44  0
            2  0  9
> table(hc.m.clusters, hc.c.clusters)
             hc.c.clusters
hc.m.clusters  1  2
            1  5  9
            2 39  0
> table(k2clusters,hc.c.clusters)
          hc.c.clusters
k2clusters  1  2
         1 28  6
         2 16  3
> table(k2clusters, db3$cluster)

k2clusters  0  1  2
         1  5 27  2
         2  8 10  1
> |
```

Interpretation:

1. Both the hc.c.cluster and hc.s.cluster assignment are in complete agreement with each other, assigning 44 data points to the first cluster and 9 to the second cluster.
2. The hc.m.clusters and hc.c. clusters agree on 5 points in the first cluster and 0 points on the second cluster.
3. The k2clusters and hc.c.clusters both agree on 28 data points and assign them to cluster 1 and agree on only 3 data points while assigning them to cluster 2.
4. The k2clusters and db3cluster agree on 27 points and assign them to cluster-1 whereas for cluster 2 they agree only on 1 point.

Question 4. Comparative Analysis : Our third research question for our classifier problems is, *'Can we accurately predict the class of Occupancy Status("I", "P" or 'S") in the Loan dataset, based on the IVs Original interest rate, bondRate, fedFundsRate, Borrower Credit Score and Original UPB? '*

IVs and DV for classifiers:

S.No	1	2	3	4	5	6	7	8	9	10	11	12	13	**14**
Variable Name	Original.Interest.Rate	bondRate	fedFundsRate	Original.UPB	Original.Loan.to.Value.Ratio..LTV.	Original.Combined.Loan.to.Value.Ratio..CLTV.	Number.of.Borrowers	Debt.To.Income..DTI.	Borrower.Credit.Score.at.Origination	Property.State	Seller.Name	Property.Type	Loan.Purpose	**Occupancy.Status**
Data type	Numeric (IV)	Numeric (IV)	Numeric (IV)	Numeric (IV)	Numeric (IV)	Numeric (IV)	Numeric (IV)	Numeric (IV)	Numeric (IV)	Nominal (IV)	Nominal (IV)	Nominal (IV)	Nominal (IV)	**Nominal (DV)**

Comparing performance of models using the CARET package: Using the CARET package in R, we were able to run different classification techniques such as Decision Trees and Random Forests from Q3 Milestone-2 and SVM and Neural Network classifiers from Q1&2 Milestone3 and perform three different cross-validations using exhaustive and non-exhaustive methods easily with a few lines of code and compare results. There are 3 important functions in the CARET package with which we accomplished this task. The train() function, the trainControl() function and the resamples() function.

1. The k-fold train control object

```
> ##Creating a train control object for k-fold cross validation
> train_control <- trainControl(method="cv", number=10)
> train_control
$method
[1] "cv"

$number
[1] 10

$repeats
[1] NA
```

Cross-validation method

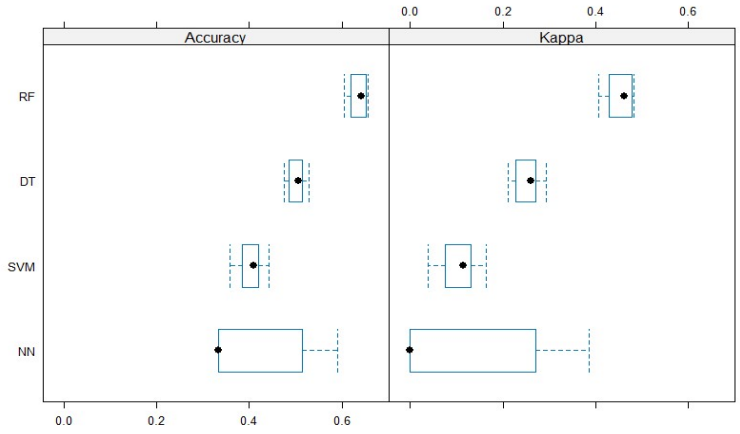
1. k-fold method: k-fold is a non-exhaustive cross-validation technique where in, dataset is divided into k folds, and k-1 folds are used for training and the remaining fold is used for testing.

Summary:

```
> results <- resamples(list(DT = model.DT, RF = model.RF, SVM = model.SVM, NN = model.NN))
> ### summary of accuracy distributions as percentiles, boxplots and dotplots
> summary(results)

Call:
summary.resamples(object = results)

Models: DT, RF, SVM, NN
Number of resamples: 10

Accuracy
        Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
DT  0.4746377 0.4882246 0.5072464 0.5039855 0.5144928 0.5289855    0
RF  0.6050725 0.6222826 0.6413043 0.6362319 0.6521739 0.6557971    0
SVM 0.3586957 0.3885870 0.4094203 0.4032609 0.4184783 0.4420290    0
NN  0.3333333 0.3333333 0.3333333 0.3978261 0.4692029 0.5905797    0

Kappa
         Min.    1st Qu.    Median       Mean   3rd Qu.      Max. NA's
DT  0.21195652 0.23233696 0.2608696 0.25597826 0.2717391 0.2934783    0
RF  0.40760870 0.43342391 0.4619565 0.45434783 0.4782609 0.4836957    0
SVM 0.03804348 0.08288043 0.1141304 0.10489130 0.1277174 0.1630435    0
NN  0.00000000 0.00000000 0.0000000 0.09673913 0.2038043 0.3858696    0
```

The cross-validation summary shows that the Random Forest (RF) model outperforms the others with the highest mean accuracy, indicating it's likely the best predictor for our dataset. The Decision Tree (DT) model, while fairly accurate, varies more across different folds. Support Vector Machines (SVM) and Neural Networks (NN) lag behind in mean accuracy, with NN particularly inconsistent, suggesting it might be overfitting or improperly tuned. Overall, RF is the standout model, combining accuracy with stability across the data.

Box Plot

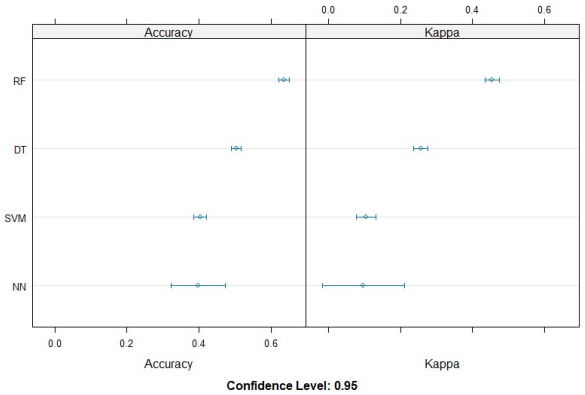


The box plot visualizes the cross-validation results for four machine learning models: Random Forest (RF), Decision Trees (DT), Support Vector Machines (SVM), and Neural Networks (NN). Focusing on accuracy, RF and DT display higher median values, with RF having a slight edge and less variability. SVM and NN show lower accuracy, with NN particularly spread out, indicating inconsistent performance.

In terms of the Kappa statistic, which adjusts accuracy for chance, the pattern is similar. RF shows a high median Kappa with tight interquartile range, reflecting strong and consistent agreement between predictions and actual values. DT follows but with more variability. SVM's Kappa is lower, yet consistent, while NN's spread suggests fluctuating model agreement across folds.

Overall, RF appears to be the most robust model, offering a compelling balance of accuracy and reliability.

Dot Plot



The dot plot summarizes the k-fold cross-validation performance for four models. Random Forest (RF) leads with the highest median accuracy and also scores well on the Kappa statistic, suggesting its predictions are both accurate and consistent. Decision Trees (DT) come next, a bit more spread out, which might mean performance varies more from fold to fold.

Support Vector Machines (SVM) and Neural Networks (NN) show lower median accuracies, but SVM is more consistent than NN, as seen by the tighter clustering of dots. NN's wide range in both accuracy and Kappa could imply that it's not as stable across different subsets of data. The plot points to RF as the most reliable choice for our data, with DT as a secondary option. SVM could be a contender with some tuning, while NN might need a more thorough look to iron out its variability.

2. k-fold with repeated CVs: When we use the "repeated CV" method, we specify that this process is to be repeated for the specified number of runs. CARET package provides optimal models with 3 runs for 10-folds each.

1. The k-fold repeated train control object: We created the following train control object for 3 runs with 10 folds each.

```
> #### comparing accuracy across classifiers
> ###Creating the train control object with 3 runs of 10-folds each for optimal models with caret package
> control <- trainControl(method="repeatedcv", number=10, repeats=3)
> control
$method
[1] "repeatedcv"

$number
[1] 10

$repeats
[1] 3
```

2. The train function: Using the train() function we could train our 4 classifier models, specifying our control object with the trControl parameter and the algorithm with the method parameter.

```
model.DT <- train(Occupancy.Status ~., data = data, trControl = control, method = "rpart" )
model.RF<- train(Occupancy.Status ~., data = data, trControl = control, method = "rf" )
model.SVM<- train(Occupancy.Status ~., data = data, trControl = control, method = "svmRadial", scale = FALSE )
model.NN <- train(Occupancy.Status ~., data = data, trControl = control, method = "nnet" )
```

3. The resamples Object: The resamples() function returns an object with results from comparison of different models. This is the summary of the distributions of accuracy and kappa values across models.

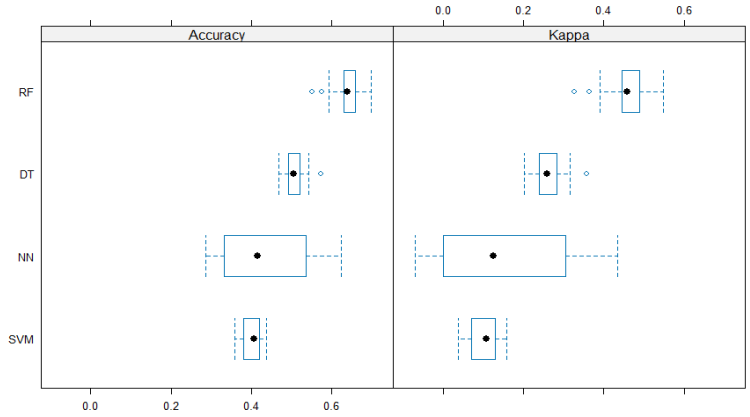
Summary:

The cross-validation summary highlights Random Forest (RF) as the top model with the highest accuracy and consistent performance across folds. Decision Trees (DT) also perform well but with slightly more variability. Support Vector Machines (SVM) and Neural Networks (NN) show lower accuracies, with NN particularly inconsistent, as suggested by its negative Kappa score.

In summary, Random Forest stands out as the most accurate and reliable model for this dataset, while Neural Networks might require a deeper dive into the model's architecture and training process to improve its performance.

```
> results <- resamples(list(DT = model.DT, RF = model.RF, SVM = model.SVM, NN = model.NN))
> summary(results)

Call:
summary.resamples(object = results)

Models: DT, RF, SVM, NN
Number of resamples: 30

Accuracy
        Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
DT  0.4673913 0.4927536 0.5054348 0.5076087 0.5208333 0.5724638    0
RF  0.5507246 0.6304348 0.6394928 0.6410628 0.6594203 0.6992754    0
SVM 0.3586957 0.3813406 0.4057971 0.4012077 0.4193841 0.4384058    0
NN  0.2862319 0.3333333 0.4166667 0.4332126 0.5353261 0.6231884    0

Kappa
            Min.     1st Qu.    Median      Mean   3rd Qu.      Max. NA's
DT   0.20108696 0.23913043 0.2581522 0.2614130 0.2812500 0.3586957    0
RF   0.32608696 0.44565217 0.4592391 0.4615942 0.4891304 0.5489130    0
SVM  0.03804348 0.07201087 0.1086957 0.1018116 0.1290761 0.1576087    0
NN  -0.07065217 0.00000000 0.1250000 0.1498188 0.3029891 0.4347826    0
```

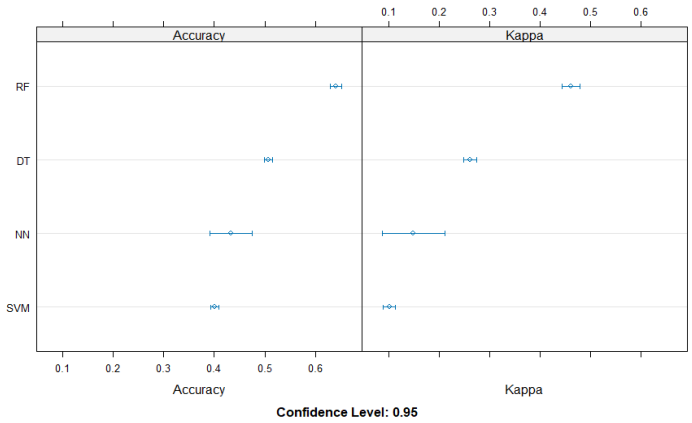
Box Plot



The box plot compares the accuracy and Kappa scores from repeated k-fold cross-validation for four models. Random Forest (RF) and Neural Networks (NN) have the highest median accuracy, but RF shows greater variability. Decision Trees (DT) have consistent but lower accuracy. Support Vector Machines (SVM) have the lowest accuracy but are consistent. RF and NN lead in Kappa scores, suggesting better prediction reliability.

In summary, RF seems to offer the best balance of accuracy and agreement in predictions, with NN as a close competitor, especially given its consistency. DT and SVM might require more investigation or parameter tuning to enhance their predictive capabilities.

Dot Plot



The dot plot illustrates the accuracy and Kappa statistics for four predictive models evaluated using k-fold cross-validation with repeated runs. Random Forest (RF) and Decision Trees (DT) display higher accuracies with a narrow confidence interval, hinting at reliable performance. Neural Networks (NN) and Support Vector Machines (SVM) show lower accuracies with wider confidence intervals, indicating less reliability and more variation in their predictive performance.

For the Kappa statistic, RF and DT again show higher values, suggesting their predictions are not only accurate but also consistent across different folds. NN and SVM have lower Kappa values, with SVM's range particularly wide, which might suggest its performance could vary significantly with different data samples.

In essence, RF and DT are the top performers in terms of both accuracy and consistency, while NN and SVM may require further tuning to improve and stabilize their predictions. The confidence level at 0.95 adds robustness to these insights, providing a high degree of trust in the cross-validation process and its results.

3. Bootstrapping

1. Bootstrapping train control object : for this model was defined the method as “boot’, number of times to be run as 30 and percentage of split as 80%-20%

```
> control <- trainControl(method="boot", number= 30,  p = 0.8)
> control
$method
[1] "boot"

$number
[1] 30

$repeats
[1] NA

$search
[1] "grid"

$p
[1] 0.8
```

Summary:

```
> results <- resamples(list(DT = model.DT, RF = model.RF, SVM = model.SVM, NN = model.NN))
> ### summary of accuracy distributions as percentiles, boxplots and dotplots
> summary(results)

Call:
summary.resamples(object = results)

Models: DT, RF, SVM, NN
Number of resamples: 30

Accuracy
        Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
DT  0.4837117 0.4981335 0.5108601 0.5104056 0.5216052 0.5393701    0
RF  0.5900196 0.6199060 0.6266773 0.6268685 0.6335800 0.6546341    0
SVM 0.3330059 0.3595672 0.3754845 0.3742154 0.3873178 0.4122367    0
NN  0.3001988 0.4006666 0.5104485 0.4680543 0.5354276 0.5866142    0

Kappa
         Min.   1st Qu.     Median       Mean    3rd Qu.      Max. NA's
DT  0.23335960 0.2514393 0.26760385 0.26818773 0.28446707 0.3130374    0
RF  0.38533243 0.4295995 0.44024968 0.44042423 0.44988782 0.4821563    0
SVM 0.01687977 0.0527379 0.07009412 0.07133024 0.08753516 0.1188156    0
NN  0.00000000 0.1169491 0.26676587 0.20658060 0.30237900 0.3790680    0
```

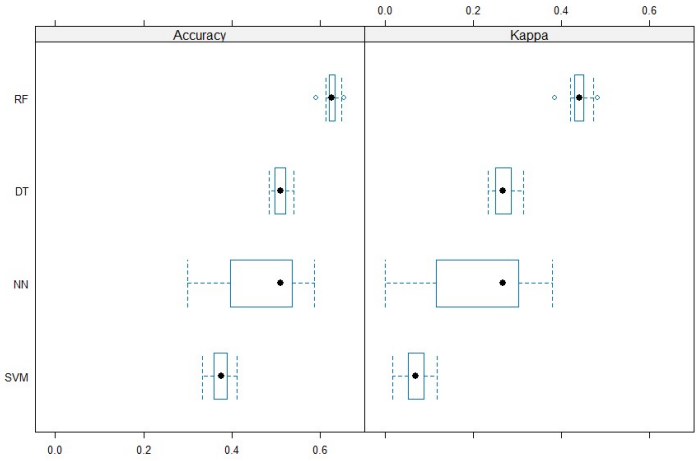
The summary table showcases the bootstrapping cross-validation results for four models. Random Forest (RF) shows a solid lead in accuracy, boasting the highest median and mean, indicative of its reliable predictive power. Decision Trees (DT) are close, but with a broader spread, suggesting some swings in performance.

Support Vector Machines (SVM) and Neural Networks (NN) lag behind RF and DT in accuracy. Notably, NN has the widest range, signaling some inconsistency across different samples.

Kappa values, reflecting accuracy adjusted for chance, align with these trends. RF maintains a strong median Kappa, confirming its dependable performance. DT is less consistent, with a lower median Kappa. SVM's Kappa scores are modest and also show considerable spread. NN's Kappa spans from zero to moderate, underscoring its unstable predictive behavior.

Overall, RF stands out as the most robust model, while NN, despite its potential, might need reevaluation or retraining for stability.

Box Plot

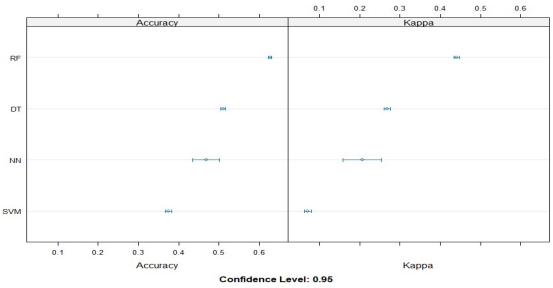


The box plot provides an informative comparison of bootstrapping cross-validation results for Random Forest (RF), Decision Trees (DT), Neural Networks (NN), and Support Vector Machines (SVM). RF and DT showcase high accuracy, but RF has a slight edge with a tighter interquartile range, pointing to consistent performance. NN, while achieving moderate accuracy, exhibits less variability than SVM, which has the lowest accuracy and a wider spread indicating inconsistency.

In terms of Kappa, which adjusts for random chance, RF and DT have comparably high scores, reinforcing their robustness. NN's Kappa is moderate, suggesting fair agreement, whereas SVM's lower Kappa aligns with its accuracy findings, hinting at less reliable predictions.

Overall, RF appears to be the most dependable model, with DT also showing promise. NN could be a viable option with some refinements, while SVM might require significant tuning to achieve competitive performance.

Dot Plot



The dot plot illustrates the bootstrapping cross-validation results for four models at a 95% confidence level. Random Forest (RF) and Decision Trees (DT) are performing well, with RF showing particularly high accuracy and Kappa values, suggesting it's not only making accurate predictions but also with consistent reliability. DT, while also strong, has slightly wider confidence intervals, indicating a bit more variability.

Neural Networks (NN) and Support Vector Machines (SVM) are yielding moderate results, with NN having slightly narrower confidence intervals, hinting at more stable predictions than SVM. However, both have lower Kappa values compared to RF and DT, which means their predictions are less consistent.

In brief, RF seems to be the most reliable model with its high and tight scoring in both accuracy and Kappa. NN and SVM might need some optimization to reach their full potential.

Final Conclusion

Considering all the cross-validation summaries, box plots, and dot plots for the Random Forest (RF), Decision Trees (DT), Neural Networks (NN), and Support Vector Machines (SVM) models, a coherent conclusion emerges. Random Forest consistently shows strong performance with high accuracy and Kappa scores across all methods, indicating its reliable predictive power. Decision Trees, while slightly less accurate than RF, provide a good balance of accuracy and consistency.

Neural Networks, despite potential volatility, demonstrate reasonable accuracy but may suffer from overfitting, as indicated by their wider confidence intervals and varying Kappa scores. Support Vector Machines generally trail behind the other models, with lower accuracy and Kappa scores, suggesting that they may not be as well-suited for this particular dataset or require more fine-tuning.

In summary, the Random Forest model is the standout choice for robustness and reliability, whereas the Support Vector Machine model appears to be the least favorable based on the given data and cross-validation methods used.

Question 5. Feature Selection

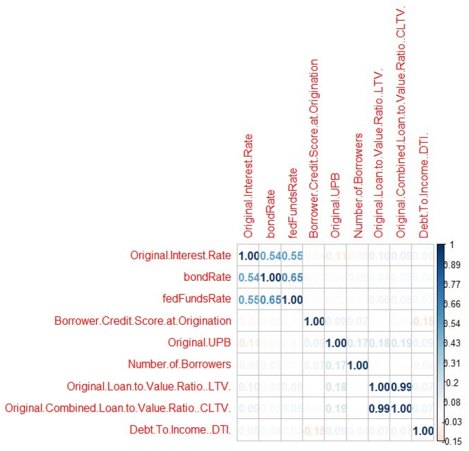
1. **FILTER FEATURE SELECTION ON LINEAR MULTIVARIATE REGRESSION**

Correlation Matrix

The visualization presents a correlation matrix for a set of variables in a dataset, presumably aimed at informing feature selection for a linear multivariate regression analysis. Each square in the matrix represents the correlation coefficient between two variables; the closer the value is to 1 or -1, the stronger the linear relationship.

Focusing on the `Original.Interest.Rate`, which is our variable of interest, we see moderate to strong correlations with `bondRate` and `fedFundsRate`. This suggests a potential predictive relationship, as changes in these rates might reflect in the interest rate being studied. Notably, `Original.Loan.to.Value.Ratio..LTV.` and `Original.Combined.Loan.to.Value.Ratio..CLTV.` display an almost perfect correlation, hinting at redundancy; one of them could likely be omitted from the regression model without losing much information. Other variables, such as `Debt.To.Income..DTI.` and `Borrower.Credit.Score.at.Origination`, show weaker correlations with `Original.Interest.Rate`. This could imply a less direct influence on the interest rate, or perhaps their impact is more nuanced and not purely linear.

Given the aim of building a regression model, we'd consider the strength of the correlations, but also be wary of multicollinearity, particularly between `LTV` and `CLTV`. We'd keep variables that have a strong relationship with our target but are not too inter-correlated, ensuring our model remains robust and interpretable. So, it is inferred from this correlation matrix that our dependent variable `Original.Interest.Rate`, is significantly correlated with the independent variables `bondRate` and `fedFundsRate`.



Summary of the model to check for significance of variables

The regression summary reveals that `bondRate` and `fedFundsRate` are strong and statistically significant predictors of the `Original.Interest.Rate`, indicated by their low p-values. The `Borrower.Credit.Score.at.Origination` is marginally significant, and while `Original.UPB` shows a significant inverse relationship, its practical impact on the interest rate might be limited. Variables like `Original.LTV` and `Debt.To.Income..DTI.` are not significant, suggesting they have little to no influence on the interest rate in this model. With an R-squared value of approximately 0.3889, the

model captures some, but not all, of the variability in the interest rate. We refined the model by removing the insignificant predictors to potentially improve its explanatory power.

```
> # Build the linear regression model using only numeric features
> model1 <- lm(Original.Interest.Rate ~ ., data = train_data)
> # Summary of the model to check for significance of variables
> summary(model1)

Call:
lm(formula = Original.Interest.Rate ~ ., data = train_data)

Residuals:
     Min       1Q   Median       3Q      Max
-3.02398 -0.42714  0.06073  0.55118  1.53313

Coefficients:
                                          Estimate Std. Error t value Pr(>|t|)
(Intercept)                              2.353e+00  3.453e-01   6.815 1.21e-11 ***
bondRate                                 7.474e-01  5.246e-02  14.246  < 2e-16 ***
fedFundsRate                             6.412e-01  3.910e-02  16.397  < 2e-16 ***
Borrower.Credit.Score.at.Origination    -8.941e-04  3.816e-04  -2.343   0.0192 *
Original.UPB                            -6.578e-07  1.111e-07  -5.920 3.72e-09 ***
Number.of.Borrowers                      4.169e-02  2.933e-02   1.421   0.1554
Original.Loan.to.Value.Ratio..LTV.      -2.866e-04  6.399e-03  -0.045   0.9643
Original.Combined.Loan.to.Value.Ratio..CLTV.  5.311e-03  6.340e-03   0.838   0.4022
Debt.To.Income..DTI.                     1.146e-03  1.657e-03   0.692   0.4893
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.7074 on 2199 degrees of freedom
Multiple R-squared:  0.3889,    Adjusted R-squared:  0.3867
F-statistic: 174.9 on 8 and 2199 DF,  p-value: < 2.2e-16

> |
```

Summary of the model after removing the statistically insignificant features

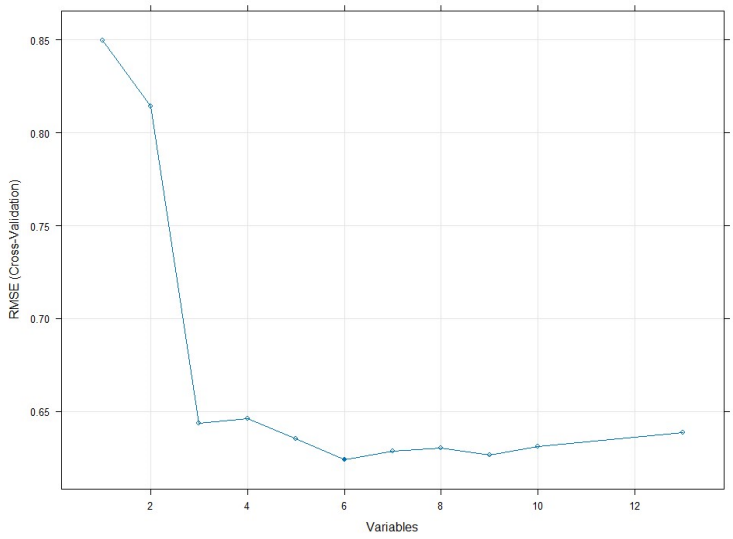
After refining our model by keeping only `bondRate` and `fedFundsRate`, the summary shows both predictors are statistically significant with p-values less than 0.05, indicated by triple asterisks. The R-squared value of around 0.3729 means the model explains approximately 37% of the variation in the `Original.Interest.Rate`, which is decent for financial data known for its complexity and influenced by many factors. The coefficients for both `bondRate` and `fedFundsRate` are positive, suggesting that as these rates increase, so does the `Original.Interest.Rate`. This trimmed model is more focused, using only variables that have a clear and significant relationship with our target variable with a minor sacrifice in R2 value.

```
> # Summary of the updated model to see the changes
> summary(model2)

Call:
lm(formula = Original.Interest.Rate ~ bondRate + fedFundsRate,
    data = train_data)

Residuals:
    Min      1Q  Median      3Q     Max
-3.1086 -0.3999  0.0901  0.5717  1.3501

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)   1.98378    0.14625   13.56   <2e-16 ***
bondRate      0.73894    0.05304   13.93   <2e-16 ***
fedFundsRate  0.64686    0.03949   16.38   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.7157 on 2205 degrees of freedom
Multiple R-squared:  0.3729,    Adjusted R-squared:  0.3723
F-statistic: 655.5 on 2 and 2205 DF,  p-value: < 2.2e-16

> |
```

2. HYBRID METHOD OF FEATURE SELECTION, DEPLOYMENT ON RANDOM FOREST MODEL

The RFE is a hybrid feature selection method that employs elements of both filter and wrapper methods. We trained a RFE Control function using Caret package and used the rfe() function in R to deploy RFE .The model began with our 14 IVs and recursively eliminated 8 features, in the order of their importance in predicting Original Interest Rate.RMSE values are used as this model is a regression Random Forest Model

The graph shows the RMSE cross-validation the model employed to arrive at the best set of variables. As we can see from the graph, for 6 variables, the RMSE is lowest.



The model eliminated 8 features and retained Occupancy Status, Seller Name, bondRate, fedFundsRate, Original CLTV and Original LTV as the most important in predicting the outcome variable.

```
> # list the chosen features
> predictors(result)
[1] "Occupancy.Status"                          "Seller.Name"
[3] "bondRate"                                  "fedFundsRate"
[5] "Original.Combined.Loan.to.Value.Ratio..CLTV." "Original.Loan.to.Value.Ratio..LTV."
>
```

Our first Random Forest model with all the features produced a correlation of 0.69 between predicted and actual values with a RMSE of 0.63.

```
[1] 2070   14
> train_control <- trainControl(method="cv", number=10)
> model.RF<- train(Original.Interest.Rate ~., data = train, trControl = train_control, method = "rf" )
> predictions.RF <- predict(model.RF, newdata = test)
> ### correlation between actual and predicted values for original model
> cor(predictions.RF, test$Original.Interest.Rate)
[1] 0.692315
> mse <- mean((predictions.RF - test$Original.Interest.Rate)^2)
> mse
[1] 0.4011501
> rmse <- sqrt(mse)
> rmse
[1] 0.6333641
```

The model retaining only the most predictive features derived from the results of RFE method produced a correlation of 0.67 with an RMSE of 0.43

```
> model.RF.E <- train(Original.Interest.Rate ~ Occupancy.Status+
+                       Seller.Name+
+                       bondRate+ fedFundsRate+
+                       Original.Combined.Loan.to.Value.Ratio..CLTV. +
+                       Original.Loan.to.Value.Ratio..LTV.,
+                       data = train, trControl = train_control, method = "rf" )
> predictions.RF.E <- predict(model.RF.E, newdata = test)
> ### correlation between actual and predicted values for original model
> cor(predictions.RF.E, test$Original.Interest.Rate)
[1] 0.6659492
> mse <- mean((predictions.RF.E - test$Original.Interest.Rate)^2)
> rmse <- sqrt(mse)
> rmse
[1] 0.6601713
> mse
[1] 0.4358262
```

Summary: The slight compromise of reduction in correlation of 0.02 gave us an improvement in RMSE of 0.2 with a great reduction in model complexity from 14 features to 6 features.

3. EMBEDDING METHOD – REGULARIZATION ON NEURAL NETWORK MODEL

In reviewing the application of the embedding feature selection method through regularization on a neural network model, we have two distinct snapshots that describe the process and outcomes of this sophisticated machine learning technique.

The first snapshot showcases the initial neural network model without regularization. The model is trained on the encoded version of our entire dataset that has 2760 observations of 93 variables. We normalized the numerical variables and encoded the categorical variables and then used the model to predict the `Original.Interest.Rate` on a separate test set. The correlation between the actual and predicted interest rates is approximately 0.706, which is a respectable figure, indicating a strong positive relationship between the model's predictions and the actual data.

```
> ### REGULARIZATION ON NEURAL NETWORKS
> data <- read.csv("Normalized_MS3dataset.csv")
> train <- data[1:2070, ]
> test <- data[2071:2760, ]
> model.NN <- neuralnet(Original.Interest.Rate ~., data = train )
> predictions.NN <- predict(model.NN, newdata = test)
> ### correlation between actual and predicted values for original model
> cor(predictions.NN, test$Original.Interest.Rate)
          [,1]
[1,] 0.706223
```

The second snapshot introduces a neural network model with regularization—specifically, a resilient backpropagation with weight decay (Rprop-WD) algorithm. This advanced method adapts the learning

rates during backpropagation to minimize predictive error, with the weight decay component acting as a countermeasure against overfitting by penalizing larger weights.

```
> model.NNR <- neuralnet(
+   formula,
+   data = train,
+   linear.output = TRUE,
+   act.fct = "logistic",
+   algorithm = "rprop+",  # Using resilient backpropagation with weight decay
+   stepmax = 1e6 #Setting a large number of steps
+ )
> predictions.NNR <- predict(model.NNR, newdata = test)
> ### correlation between actual and predicted values for original model
> cor(predictions.NNR, test$Original.Interest.Rate)
          [,1]
[1,] 0.7063019
```

Upon application of the Rprop-WD model, we observe a slight improvement in the correlation between predicted and actual values, increasing to approximately 0.7063. This increment, while seemingly minor, suggests that the regularization technique has enhanced the model's generalization capabilities, ensuring that it performs slightly better with the data it hasn't been trained on.

In technical and practical terms, the report highlights the effectiveness of regularization techniques like Rprop-WD in neural network models. The use of weight decay as a regularization method appears to have contributed to a model that can predict interest rates with a slightly higher degree of accuracy and reliability. This kind of nuanced improvement is crucial in the field of data science, where even small percentage points can have significant impacts, particularly in large-scale or sensitive financial applications.

This comparative analysis underscores the importance of regularization in complex models and provides a clear direction for further model refinement and evaluation against additional data or through cross-validation to confirm the consistency of these results.

The comparison of results before and after applying the regularization technique in neural network modeling reveals a nuanced but noteworthy enhancement in performance. The modest increase in the correlation coefficient after regularization signals an improvement in the model's ability to make predictions that align closely with actual outcomes. This improvement, although slight, is indicative of the positive impact that feature selection through regularization can have on a model's predictive accuracy. It suggests that by carefully tuning the complexity of the model and mitigating overfitting, we can achieve a more refined and possibly more reliable predictive tool. Such improvements, particularly in fields like

finance or healthcare, where precision is paramount, can translate into significant benefits and more informed decision-making based on the model's output.

4. WRAPPER- SFS- METHOD ON ENTIRE DATASET REGRESSION MODEL

We created a base model with only the intercept value for the outcome variable, another model with all variables and a step model with forward selection method to analyze the improvement of the model using Wrapper methods. The Step() function in R searches for the best possible model by iteratively selecting features, to arrive at a model with the lowest possible AIC. When comparing models with different subsets of features, the lowest AIC is the best model.

List of shortlisted variables by the step model: These variables are better predictive features for the outcome variable.

```
> base.mod <- lm(Original.Interest.Rate ~ 1, data=train)
> # Step 2: Full model with all predictors
> all.mod <- lm(Original.Interest.Rate ~ . , data= train)
> # Step 3: Perform step-wise algorithm . direction=both, forward, backward,
> stepMod <- step(base.mod, scope = list(lower = base.mod, upper = all.mod), direction = "forward", trace = 0, steps = 1000)
> shortlistedVars <- names(unlist(stepMod[[1]]))
> shortlistedVars <- shortlistedVars[!shortlistedVars %in% "(Intercept)"]
> shortlistedVars <- shortlistedVars[!shortlistedVars %in% "(Intercept)"]
> # remove intercept
> print(shortlistedVars)
 [1] "fedFundsRate"                                    "Occupancy.StatusP"
 [3] "Occupancy.StatusS"                               "bondRate"
 [5] "Seller.NameCrossCountry Mortgage, LLC"           "Seller.NameDHI Mortgage Company, Ltd."
 [7] "Seller.NameFairway Independent Mortgage Corporation"  "Seller.NameFifth Third Bank, National Association"
 [9] "Seller.NameGuaranteed Rate, Inc."                "Seller.NameGuild Mortgage Company LLC"
[11] "Seller.NameJPMorgan Chase Bank, National Association" "Seller.NameLakeview Loan Servicing, LLC"
[13] "Seller.NameLennar Mortgage, LLC"                 "Seller.NameloanDepot.com, LLC"
[15] "Seller.NameMovement Mortgage, LLC"               "Seller.NameNationStar Mortgage, LLC"
[17] "Seller.NameNewRez LLC"                           "Seller.NameNexBank"
[19] "Seller.NameOther"                                "Seller.NamePennyMac Corp."
[21] "Seller.NamePennyMac Loan Services, LLC"          "Seller.NamePHH Mortgage Corporation"
[23] "Seller.NamePlanet Home Lending, LLC"             "Seller.NameRocket Mortgage, LLC"
[25] "Seller.NameTruist Bank (formerly SunTrust Bank)" "Seller.NameU.S. Bank N.A."
[27] "Seller.NameUnited Wholesale Mortgage, LLC"       "Seller.NameWells Fargo Bank, N.A."
[29] "Original.Combined.Loan.to.Value.Ratio..CLTV."    "Borrower.Credit.Score.at.Origination"
[31] "Property.TypeCP"                                 "Property.TypeMH"
[33] "Property.TypePU"                                 "Property.TypeSF"
[35] "Original.UPB"                                    "Number.of.Borrowers"
[37] "Loan.PurposeP"                                   "Loan.PurposeR"
>
```

Final Model with short listed variables :

```
> summary(stepMod)

Call:
lm(formula = Original.Interest.Rate ~ fedFundsRate + Occupancy.Status +
    bondRate + Seller.Name + Original.Combined.Loan.to.Value.Ratio..CLTV. +
    Borrower.Credit.Score.at.Origination + Property.Type + Original.UPB +
    Number.of.Borrowers + Loan.Purpose, data = train)

Residuals:
     Min       1Q   Median       3Q      Max
-2.82231 -0.32735  0.07307  0.40729  1.72333

Coefficients:
                                                 Estimate Std. Error t value Pr(>|t|)
(Intercept)                                     2.655e+00  3.609e-01   7.359 2.69e-13 ***
fedFundsRate                                    5.719e-01  3.585e-02  15.955  < 2e-16 ***
Occupancy.StatusP                              -6.771e-01  3.572e-02 -18.954  < 2e-16 ***
Occupancy.StatusS                              -8.355e-02  3.425e-02  -2.439 0.014797 *
bondRate                                        7.856e-01  4.736e-02  16.588  < 2e-16 ***
Seller.NameCrossCountry Mortgage, LLC           7.702e-01  2.127e-01   3.621 0.000301 ***
Seller.NameDHI Mortgage Company, Ltd.          -6.780e-01  2.249e-01  -3.015 0.002600 **
Seller.NameFairway Independent Mortgage Corporation  7.438e-01  2.171e-01   3.426 0.000625 ***
Seller.NameFifth Third Bank, National Association    1.107e-01  2.450e-01   0.452 0.651347
Seller.NameGuaranteed Rate, Inc.                5.445e-01  2.375e-01   2.292 0.021985 *
Seller.NameGuild Mortgage Company LLC           5.654e-01  2.236e-01   2.529 0.011523 *
Seller.NameJPMorgan Chase Bank, National Association  5.035e-01  2.247e-01   2.241 0.025160 *
Seller.NameLakeview Loan Servicing, LLC         5.665e-01  2.304e-01   2.458 0.014038 *
Seller.NameLennar Mortgage, LLC                -5.801e-02  2.345e-01  -0.247 0.804669
Seller.NameloanDepot.com, LLC                   4.801e-01  2.148e-01   2.235 0.025523 *
Seller.NameMovement Mortgage, LLC               6.233e-01  2.063e-01   3.021 0.002554 **
Seller.NameNationStar Mortgage, LLC             5.360e-01  2.280e-01   2.351 0.018824 *
Seller.NameNewRez LLC                           7.793e-01  2.106e-01   3.701 0.000220 ***
Seller.NameNexBank                              1.117e-01  2.792e-01   0.400 0.689066
Seller.NameOther                                4.262e-01  1.933e-01   2.204 0.027604 *
Seller.NamePennyMac Corp.                       5.966e-01  2.114e-01   2.822 0.004815 **
Seller.NamePennyMac Loan Services, LLC          3.804e-02  2.118e-01   0.180 0.857498
Seller.NamePHH Mortgage Corporation             6.534e-01  2.108e-01   3.100 0.001960 **
Seller.NamePlanet Home Lending, LLC             7.604e-01  2.277e-01   3.340 0.000854 ***
Seller.NameRocket Mortgage, LLC                 3.013e-01  1.970e-01   1.530 0.126266
Seller.NameTruist Bank (formerly SunTrust Bank)  2.100e-01  2.171e-01   0.968 0.333396
Seller.NameU.S. Bank N.A.                       2.256e-01  2.209e-01   1.022 0.307056
Seller.NameUnited Wholesale Mortgage, LLC       5.486e-01  1.985e-01   2.763 0.005771 **
Seller.NameWells Fargo Bank, N.A.               2.987e-01  2.052e-01   1.456 0.145667
Original.Combined.Loan.to.Value.Ratio..CLTV.    1.047e-02  1.137e-03   9.214  < 2e-16 ***
Borrower.Credit.Score.at.Origination           -2.055e-03  3.525e-04  -5.830 6.42e-09 ***
Property.TypeCP                                -5.160e-01  3.548e-01  -1.455 0.145948
Property.TypeMH                                 2.348e-01  1.212e-01   1.938 0.052740 .
Property.TypePU                                -1.301e-02  4.811e-02  -0.270 0.786893
Property.TypeSF                                 1.048e-01  4.437e-02   2.361 0.018333 *
Original.UPB                                   -3.267e-07  1.051e-07  -3.109 0.001901 **
Number.of.Borrowers                             5.349e-02  2.672e-02   2.002 0.045420 *
Loan.PurposeP                                  -8.639e-02  4.331e-02  -1.995 0.046194 *
Loan.PurposeR                                  -9.941e-02  7.954e-02  -1.250 0.211504
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.6055 on 2031 degrees of freedom
Multiple R-squared:  0.5555,    Adjusted R-squared:  0.5472
F-statistic: 66.79 on 38 and 2031 DF,  p-value: < 2.2e-16
```

Summary:

Our regression analysis incorporated a stepwise feature selection method to refine the predictive model for our dataset. This iterative process aimed to optimize the Akaike Information Criterion (AIC), a measure of the model's quality, balancing complexity and fit.

Initial Observations

The original model included all predictors, which led to an inflated number of variables, some of which likely did not significantly contribute to the model's predictive power.

The all-encompassing approach, while comprehensive, risked overfitting and reduced interpretability due to the excess of variables.

Stepwise Forward Selection

Through the stepwise forward selection, we systematically built up a model starting with no variables and adding one at a time. Each addition was validated by its impact on lowering the AIC, a method favoring parsimony. This method highlighted the most impactful variables, such as 'bondRate' and 'fedFundsRate', which showed strong statistical significance in the final model.

Final Model Insights

The final model presented a more streamlined selection of variables, emphasizing those with substantial influence on the 'Original.Interest.Rate'.Notable variables like 'Occupancy.Status' and 'bondRate' persisted in the final model, suggesting their stronger relationship with the interest rate. Seller names, surprisingly, figured prominently, indicating possible patterns in interest rates associated with different lenders or financial entities.

This process underscores the practicality of stepwise selection in navigating vast datasets, allowing us to distill a model that is both accurate and interpretable.

Question 6. Extra Credit

Reflecting on the ethical considerations of the proposed research questions, here's a synthesis of potential issues based on the discussion in Class 10:

Predicting Interest Rate:

Data Consent and Anonymity: If the Fannie Mae dataset includes any personal identifiers, it was crucial to obtain consent from individuals for the use of their data in such analysis. An ethical concern arises if the data was used without explicit permission or if there is potential for re-identification.

Variable Influence and Discriminatory Bias: For instance, if the Debt-To-Income ratio has a higher influence on interest rates and this variable correlates with certain socioeconomic classes, models might inadvertently perpetuate class-based discrimination. This could lead to higher interest rates disproportionately affecting lower-income applicants.The selection of the 15 IVs, as discussed in the second milestone, was carried out with an awareness of their potential to embed socioeconomic biases. We evaluated each variable's influence, especially in our logistic regression models, to mitigate the risk of discriminatory bias, which is essential for the equitable application of predictive analytics in lending practices.

Influence of Criteria on Interest Rate:

Implicit Bias and Variable Weighting: If the model finds that the number of borrowers significantly affects the interest rate, this could disadvantage single applicants or those who are unmarried, potentially reflecting a societal bias towards traditional family structures.Our findings, particularly from the SVM and neural network models, revealed the significant

predictive power of borrower numbers. We critically assessed this variable's weight against societal norms and biases that could be perpetuated through our model, as these insights could have profound ethical implications.

Lending Institutions' Evaluation Criteria:

Fair Representation and Missing Data: The 'unknown' category could mask important trends and biases. For example, if minority-owned lending institutions are disproportionately categorized as 'other', the model may not accurately represent their lending practices, leading to less effective predictions for those institutions.During our Data Categorization Process, the issue of 'unknown' institutions was addressed to ensure our models did not misrepresent lending practices due to incomplete data. The handling of missing data was carefully designed to ensure a fair representation of all data points.

Consistency in Criteria Evaluation: Different institutions may weigh criteria differently, leading to varied interest rates. If the model fails to capture this nuance, it may suggest a homogeneity in lending practices that does not exist, potentially misguiding applicants about the best institution for their needs. Our Model Evaluation Metrics were consistently applied across different criteria, including precision, recall, and F1 scores. This uniformity was crucial to guarantee that the performance metrics did not favor any particular group or institution.

Predicting Occupancy Status:

Variable Selection and Socioeconomic Bias: The choice of variables like Borrower Credit Score could embed systemic biases, as credit scores may be lower in communities with historical disadvantages. A model that heavily relies on this variable could unfairly predict occupancy status, impacting those communities negatively.The Clustering Techniques and Variable Impact section discussed the potential for socioeconomic bias in variable selection. It was imperative to reflect on how these selections might affect different groups and the ethical implications of such effects on predictions, particularly for Occupancy Status.

Model Fairness Across Demographics: If the model is more accurate for one demographic over another, it may lead to unequal treatment. For example, if the model is less accurate in predicting occupancy status for lower-income individuals, they might be unfairly classified, affecting their loan terms.The accuracy of our models was analyzed across different demographic groups - We

acknowledged where performance disparities occurred, understanding the importance of developing models that are fair and equitable across all demographics.

The ethical implications are not just theoretical; they have real-world consequences. For instance, a model that inaccurately predicts higher interest rates for certain demographics could lead to systemic financial disadvantages. Hence, ethical considerations must be at the forefront of the entire modeling process, from data collection to model deployment.

Contributions

All the 3 team members made equal contributions for research, analysis, code creation, PPT and Video presentation for Milestone-3.

References

http://journal.r-project.org/archive/2010-1/RJournal_2010-1_Guenther+Fritsch.pdf
https://www.pluralsight.com/guides/encoding-data-with-r
https://www.rdocumentation.org/packages/cluster/versions/2.1.4/topics/daisy