

Loan Aggregator with Linear Regression in R

In this project we will use Linear regression to predict the most optimal loan for our clients.

Project Scenario

We are a loan aggregation organization. We receive applications seeking loan from different clients. Based on their information like income, etc., we determine which financial institution's loan would be the best for them in terms of interest rate. Usually this is done by people working at the institution but the process is not very efficient. We want to predict the interest rates that will be provided to an application based on its features so that we can narrow down our search to top 5 or so loans. This will make the process faster and consistent, thus bringing in more customers.

In this project, we will focus on the data of a specific financial institution and what kind of rates it will provide to an application.

Data Preparation & Manipulation

After setting the path and importing the dataset, we see that a lot of the attributes (columns) are characters.

We can also see that many columns like Amount.Requested, Amount.Funded, Interest.Rate, etc. are numbers. They have been converted to characters because some of their rows may have been "NA" because of lack of values or contain "%".

The screenshot shows the RStudio interface with the following components:

- Editor Pane:** Displays the R script `MyPracticeSheet.R`. The code sets the working directory, imports the dataset `Loans_data.csv`, and performs data manipulation using `dplyr` to convert columns to numeric types where applicable. A specific line of code is highlighted in blue: `mutate(Interest.Rate=as.numeric(gsub("%","",Interest.Rate)) , Debt.To.Income.Ratio=as.numeric(gsub("%","",Debt.To.Income.Ratio)) , Open.CREDIT.Lines=as.numeric(Open.CREDIT.Lines) , Amount.Requested=as.numeric(Amount.Requested) , Amount.Funded.By.Investors=as.numeric(Amount.Funded.By.Investors) , Revolving.CREDIT.Balance=as.numeric(Revolving.CREDIT.Balance))`.
- Console Pane:** Shows the output of running the script. It includes a warning message: `The following objects are masked from 'package:base': intersect, setdiff, setequal, union`. Below this, the `glimpse(l1d)` command is run, displaying the first few rows of the dataset `l1d`. The output shows various columns with their data types and values.
- Files Pane:** Displays a file tree under the path `D:/Edvancer Education Services`. The tree includes files like `winequality-white.csv`, `winequality-red.csv`, `Untitled.R`, `test.html`, `temp.Rdata`, `souvenircsv`, `skirts.csv`, `rain.csv`, `OnlineNewsPopularity.csv`, `namesbystate`, `loans data.csv`, `kings.csv`, `hsb2.sas7bdat`, `gaming1.sas7bdat`, `Existing Base.csv`, and `exercise.xls`.

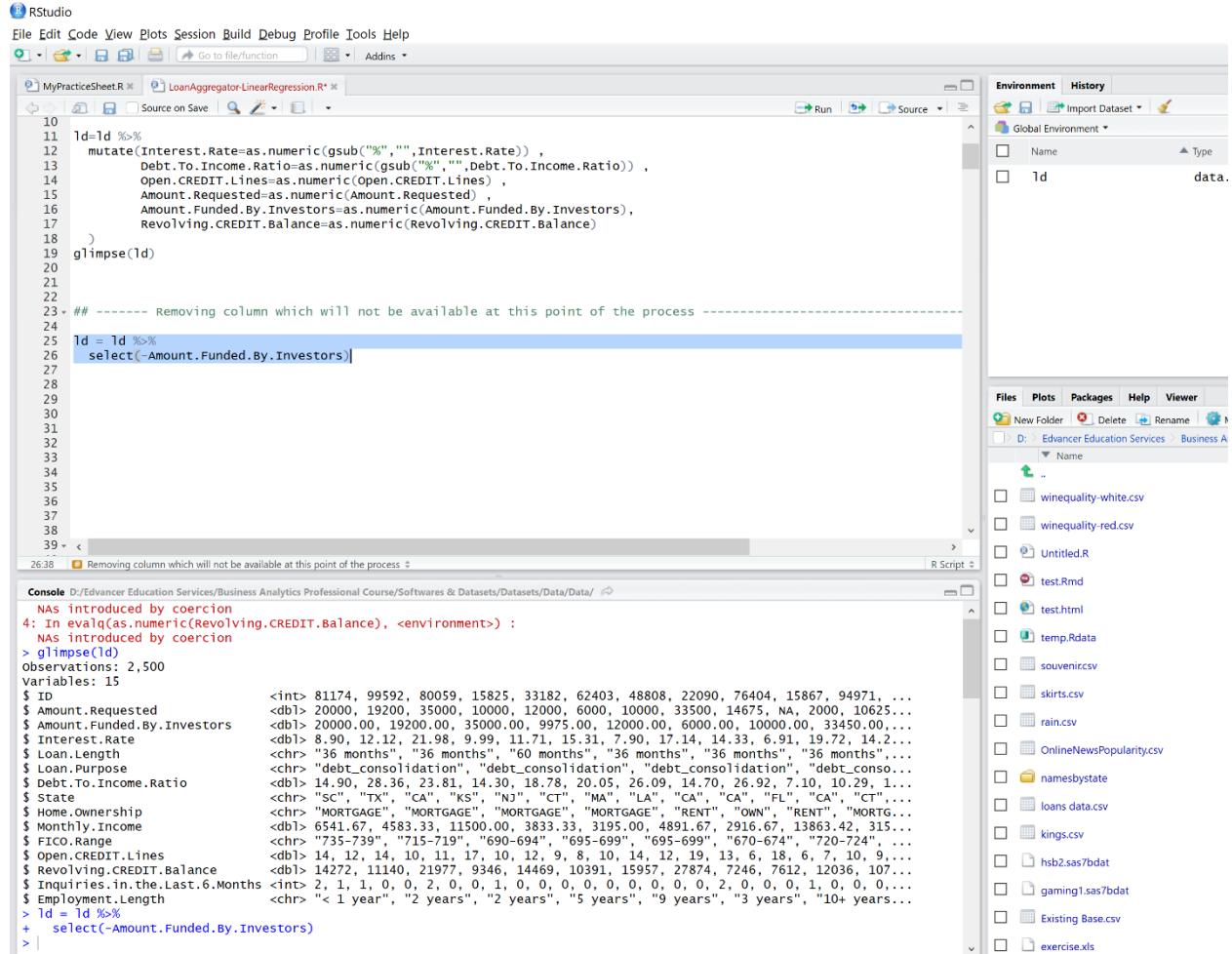
So, we will remove the "%" from the values of some of the columns like Interest Rate and Debt-to-Income ratio, also we will convert some of the character columns into numeric. We may get a warning because we are converting values that have "NA", but they can be ignored.

We can use glimpse to see that the rows we had targeted have become numbers (or double).

The screenshot shows the RStudio interface with the following details:

- Top Bar:** RStudio, File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Left Panel:** MyPracticeSheet.R, LoanAggregator-LinearRegression.R, Source on Save, Run, Source.
- Code Editor:** R script pane containing R code for data processing and model building. The code includes reading a CSV file, setting working directory, creating a new dataset 'ld' with various financial metrics, and performing a left join ('ld %>% select(-Amount.Funded.By.Investors)'). Lines 18-30 show a warning message about coercion.
- Console:** Displays the output of the R code, including the resulting dataset structure and a warning message.
- Environment:** Global Environment pane showing objects 'Name' and 'ld'.
- Files:** Files pane listing various datasets and files used in the session.

We have a column Amount Funded by Investors, but this is not a part of our process right now as we are assuming that the application is new and not historical data. So, we will remove it from our dataframe.



We have another column called FICO Range, which basically gives the range of Fico Score that an applicant has. This is a categorical data, but it has a numerical importance too i.e. higher range means better financial status of the client. So, we cannot completely ignore this aspect of the column too. The solution to this is to convert the range given for each applicant into a mean Fico score. After calculation, we can remove the intermediate and parent column.

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

MyPracticeSheet.R LoanAggregator-LinearRegression.R*

Source on Save Run Source Addins

```
21
22
23 ## ----- Removing column which will not be available at this point of the process -----
24
25 ld = ld %>%
26   select(-Amount.Funded.By.Investors)
27
28
29 ## ----- Converting FICO range into a Mean Score to get better utilisation -----
30
31 ld= ld %>%
32   mutate(f1=as.numeric(substr(FICO.Range,1,3)),
33         f2=as.numeric(substr(FICO.Range,5,7)),
34         fico=0.5*(f1+f2)
35   ) %>%
36   select(-FICO.Range,-f1,-f2)
37 glimpse(ld)
38
39
40
41
42
43
44
45
46
47
48
49
50 <
```

Converting FICO range into a Mean Score to get better utilisation ▾

Console D:/Educator Education Services/Business Analytics Professional Course/Softwares & Datasets/Datasets/Data/Data/ ↵

```
> ld= ld %>%
+   mutate(f1=as.numeric(substr(FICO.Range,1,3)),
+         f2=as.numeric(substr(FICO.Range,5,7)),
+         fico=0.5*(f1+f2)
+   ) %>%
+   select(-FICO.Range,-f1,-f2)
> glimpse(ld)
Observations: 2,500
Variables: 14
$ ID              <int> 81174, 99592, 80059, 15825, 33182, 62403, 48808, 22090, 76404, 15867, 94971, ...
$ Amount.Requested <dbl> 40000, 19200, 35000, 10000, 12000, 6000, 10000, 33500, 14675, NA, 2000, 10625, ...
$ Interest.Rate    <dbl> 8.90, 12.12, 21.98, 9.99, 11.71, 15.31, 7.90, 17.14, 14.33, 6.91, 19.72, 14.2, ...
$ Loan.Length      <chr> "36 months", "36 months", "60 months", "36 months", "36 months", "36 months", ...
$ Loan.Purpose     <chr> "debt_consolidation", "debt_consolidation", "debt_consolidation", "debt_cons...
$ Debt.To.Income.Ratio <dbl> 14.90, 28.36, 23.81, 14.30, 18.78, 20.05, 26.09, 14.70, 26.92, 7.10, 10.29, 1...
$ State            <chr> "SC", "TX", "CA", "KS", "NJ", "CT", "MA", "LA", "CA", "FL", "CA", "CT", ...
$ Home.Ownership    <chr> "MORTGAGE", "MORTGAGE", "MORTGAGE", "MORTGAGE", "RENT", "OWN", "RENT", "MORT...
$ Monthly.Income    <dbl> 6541.67, 4583.33, 11500.00, 3833.33, 3195.00, 4891.67, 2916.67, 13863.42, 315.1...
$ Open.CREDIT.Lines <dbl> 14, 12, 14, 10, 11, 17, 10, 12, 9, 8, 10, 14, 12, 19, 13, 6, 18, 6, 7, 10, 9, ...
$ Revolving.CREDIT.Balance <dbl> 14272, 11140, 21977, 9346, 14469, 10391, 15957, 27874, 7246, 7612, 12036, 107...
$ Inquiries.in.the.Last.6.Months <int> 2, 1, 1, 0, 0, 2, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, ...
$ Employment.Length <chr> "1 year", "2 years", "2 years", "5 years", "9 years", "3 years", "10+ years"...
$ fico              <dbl> 737, 717, 692, 697, 697, 672, 722, 707, 687, 717, 672, 667, 672, 737, 727, 73...
```

We have another column Employment Length, which contains number of years the applicant has been employed. It is a categorical variable, with columns like "<1 years" and "10+ years". We will convert this into a numerical column and convert the "<1 years" and "10+ years" columns into 0 and 10 respectively. At the same time we will omit the parent column and also remove rows with No Values.

The screenshot shows the RStudio interface with the following components:

- Top Bar:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Left Panel:** MyPracticeSheet.R (active), Source on Save, Run, Source.
- Code Editor:**

```

33 f2<-as.numeric(substr(FICO.Range,5,7)),
34 ) %>%
35   fico=0.5^(f1-f2)
36   select(-FICO.Range,-f1,-f2)
37   glimpse(l1d)
38
39
40 ## ----Converting work years into numerical variable -----
41 l1d<-l1d %>%
42   mutate(el=ifelse(substr(Employment.Length,1,2)=="10",10,Employment.Length),
43         el=ifelse(substr(Employment.Length,1,1)<"0",el),
44         el=gsub("years","",el),
45         el=gsub("year","",el),
46         el=as.numeric(el))
47 ) %>%
48   select(-Employment.Length) %>%
49   na.omit()
50
51 glimpse(l1d)
52
53
54
55
56
57
58
59
60
61
62
  
```
- Console:**

```

+ el=as.numeric(el)
+ ) %>%
+   select(-Employment.Length) %>%
+   na.omit()
Warning message:
In evalq(as.numeric(el), <environment>) : NAs introduced by coercion
> glimpse(l1d)
observations: 2,396
variables: 14
$ ID              <int> 81174, 99592, 80059, 15825, 33182, 62403, 48808, 22090, 76404, 94971, 36911, ...
$ Amount.Requested <dbl> 20000, 19200, 35000, 10000, 12000, 6000, 10000, 33500, 14675, 2000, 10625, 28...
$ Interest.Rate    <dbl> 8.90, 12.12, 21.98, 9.99, 11.71, 15.31, 7.90, 17.14, 14.33, 19.72, 14.27, 21...
$ Loan.Length      <chr> "36 months", "36 months", "60 months", "36 months", "36 months", "36 months"...
$ Loan.Purpose     <chr> "debt_consolidation", "debt_consolidation", "debt_consolidation", "debt_conso...
$ Debt.To.Income.Ratio <dbl> 14.90, 28.36, 23.81, 14.30, 18.78, 20.05, 26.09, 14.70, 26.92, 10.29, 12.54, ...
$ State            <chr> "SC", "TX", "CA", "KS", "NJ", "CT", "MA", "LA", "CA", "FL", "CA", "CT", "CT", ...
$ Home.Ownership    <chr> "MORTGAGE", "MORTGAGE", "MORTGAGE", "MORTGAGE", "RENT", "OWN", "RENT", "MORTG...
$ Monthly.Income    <dbl> 6541.67, 4583.33, 11500.00, 3833.33, 3195.00, 4891.67, 2916.67, 13863.42, 315...
$ Open.CREDIT.Lines <dbl> 14, 12, 14, 10, 11, 17, 10, 12, 9, 10, 14, 12, 19, 13, 6, 18, 6, 7, 9, 12, 11...
$ Revolving.CREDIT.Balance <dbl> 14272, 11140, 21977, 9346, 14469, 10391, 15957, 27874, 7246, 12036, 10767, 10...
$ Inquiries.in.the.Last.6.Months <int> 2, 1, 1, 0, 0, 2, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 1, 0, 0, 2, 0, ...
$ fico              <dbl> 737, 717, 692, 697, 697, 672, 722, 707, 687, 672, 667, 672, 737, 727, 732, 69...
$ el               <dbl> 0, 2, 2, 5, 9, 3, 10, 10, 8, 6, 0, 1, 1, 0, 9, 7, 9, 3, 10, 6, 2, 7, 3, 7, 6, ...
  
```
- Environment:** Global Environment, Name, Type, l1d, data.f
- Files:** Files, Plots, Packages, Help, Viewer, D:\Edvancer Education Services\Business Analytics Professional Course\Softwares & Datasets\Datasets\Data\Data/, winequality-white.csv, winequality-red.csv, Untitled.R, test.Rmd, test.html, temp.rdata, souvenircsv, skirts.csv, rain.csv, OnlineNewsPopularity.csv, namesbystate, loans data.csv, kings.csv, hsb2.sas7bdat, gaming1.sas7bdat, Existing Base.csv, exercise.xls

We have a categorical variable Home Ownership which has 3 categories - Mortgage, Rent, Own and Other. for these we will create dummy variables, where the value will be 1 for that category and 0 otherwise. We will not consider the Other category as it could be anything and has less impact than the others.

The screenshot shows the RStudio interface with the following details:

- Top Bar:** RStudio, File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Left Panel:** MyPracticeSheet.R, LoanAggregator-LinearRegression.R*
- Code Area:**

```

44 el=year %>%
45   el=gsub("years","",el),
46   el=as.numeric(el)
47 ) %>%
48 select(-Employment.Length) %>%
49 na.omit()
50
51 glimpse(l1d)
52
53 ## -----Converting Home Ownership into Dummy Variable -----
54 table(l1d$Home.ownership)
55
56
57 l1d %>%
58   mutate(HW_RENT=as.numeric(Home.ownership=="RENT"),
59         HW_MORT=as.numeric(Home.ownership=="MORTGAGE"),
60         HW_OWN=as.numeric(Home.ownership=="OWN")) %>%
61   select(-Home.ownership)
62
63 glimpse(l1d)
64
65
66
67
68
69
70
71
72
73 <-- Converting Home Ownership into Dummy Variable
    
```
- Console Area:**

```

  Console D:/Edvancer Education Services/Business Analytics Professional Course/Softwares & Datasets/Datasets/Data/Data/
> Table(l1d$Home.ownership)

  MORTGAGE OTHER OWN RENT
1 1098      5 186 1107

>
> l1d %>%
+   mutate(HW_RENT=as.numeric(Home.ownership=="RENT"),
+         HW_MORT=as.numeric(Home.ownership=="MORTGAGE"),
+         HW_OWN=as.numeric(Home.ownership=="OWN")) %>%
+   select(-Home.ownership)
> glimpse(l1d)
Observations: 2,396
Variables: 16
 $ ID                  <int> 81174, 99592, 80059, 15825, 33182, 62403, 48808, 22090, 76404, 94971, ...
 $ Amount.Requested     <dbl> 20000, 19200, 35000, 10000, 12000, 6000, 10000, 33500, 14675, 2000, 10625, ...
 $ Interest.Rate        <dbl> 8.90, 12.12, 21.98, 9.99, 11.71, 15.31, 7.90, 17.14, 14.33, 19.72, 14.27, 21.71, ...
 <chr> "36 months", "36 months", "61 months", "36 months", "36 months", "36 months", ...
 <chr> "debt_consolidation", "debt_consolidation", "debt_consolidation", "debt_consolidation", ...
 <dbl> 14.90, 28.36, 23.81, 14.30, 18.78, 20.05, 26.09, 14.70, 26.92, 10.29, 12.54, ...
 <chr> "SC", "TX", "CA", "KS", "N", "CT", "MA", "LA", "CA", "FL", "CA", "CT", "CT", ...
 <dbl> 6541.67, 4583.33, 11500.00, 3833.33, 3195.00, 4891.67, 2916.67, 13863.42, 315.00, ...
 <dbl> 14, 12, 14, 10, 11, 17, 10, 12, 9, 10, 14, 12, 19, 13, 6, 18, 6, 7, 9, 12, 11, ...
 <dbl> 14272, 11140, 21977, 9346, 14469, 10391, 15957, 27874, 7246, 12036, 10767, 10446, ...
 $ Inquiries.in.the.last.6.Months <int> 2, 1, 1, 0, 0, 2, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 2, 0, ...
    
```
- Right Panels:**
 - Environment:** Global Environment, Name, 1d
 - History:** History
 - Files:** Files, Plots, Packages, Help, Viewer, New Folder, Delete, Rename, D:/Edvancer Education Services/Business Analytics Professional Course/Softwares & Datasets/Datasets/Data/Data/, winequality-white.csv, winequality-red.csv, Untitled.R, test.Rmd, test.html, temp.Rdata, souvenircsv, skirts.csv, rain.csv, OnlineNewsPopularity.csv, namesbystate, loans data.csv, kings.csv, hsb2.sas7bdat, gaming1.sas7bdat, Existing Base.csv, exercise.xls

The Loan Purpose also contains 14 different categorical variables. We can convert them all to Dummy Variables but that would be unnecessary because most of these categories have very few rows. In our project, we will consider and dummify only 3 categories - Credit Card, Debt Consolidation and Other.

The screenshot shows the RStudio interface with the following components:

- Code Editor:** Displays an R script titled "LoanAggregator-LinearRegression.R" containing code to convert categorical variables into dummy variables. The code includes filtering for "MORTGAGE" and "OWN", selecting "Home.ownership", and then dummifying "Loan.Purpose" into three categories: "credit_card", "debt_consolidation", and "other".
- Environment View:** Shows the global environment with a variable "ld" selected.
- Console View:** Shows the results of the R commands run in the script. It includes a table of counts for various loan purposes and the resulting dummified data frame "ld".
- File Explorer:** Shows the file structure under "D:\Edvancer Education Services\Business Analytics Professional Course\Softwares & Datasets\Datasets\Data\Data/".

```

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
MyPracticeSheet.R  LoanAggregator-LinearRegression.R*
59 HW_MORT=as.numeric(Home.Ownership=="MORTGAGE"),
60 HW_OWN=as.numeric(Home.Ownership=="OWN") %>%
61 select(-Home.ownership)
62
63 glimpse(ld)
64
65 ## ----- Converting Loan Purpose into Dummy Variables
66 ## ----- Will only consider 3 categories to dummify -----
67
68 table(ld$Loan.Purpose)
69
70 ld=ld %>%
71   mutate(LP_cc=as.numeric(Loan.Purpose=="credit_card"),
72         LP_dc=as.numeric(Loan.Purpose=="debt_consolidation"),
73         LP_other=as.numeric(Loan.Purpose=="other"))
74 ) %>%
75 select(-Loan.Purpose)
76
77 glimpse(ld)
78
79
80
81
82
83
84
85
86
87
88
77:12 Will only consider 3 categories to dummify : 

```

```

Console D:\Edvancer Education Services\Business Analytics Professional Course\Softwares & Datasets\Datasets\Data\Data/
$ HW_OWN
> table(ld$Loan.Purpose)
      car credit_card debt_consolidation educational home_improvement house
        49          426           1261          14            144          20
  major_purchase medical moving other renewable_energy small_business
        98            28           28          184             4           82
    vacation wedding
        20            38
> ld=ld %>%
+   mutate(LP_cc=as.numeric(Loan.Purpose=="credit_card"),
+         LP_dc=as.numeric(Loan.Purpose=="debt_consolidation"),
+         LP_other=as.numeric(Loan.Purpose=="other"))
+ ) %>%
+   select(-Loan.Purpose)
> glimpse(ld)
Observations: 2,396
Variables: 18
$ ID
$ Amount.Requested
$ Interest.Rate
$ Loan.Length
$ Debt.To.Income.Ratio
$ State
$ Monthly.Income

```

Loan Length has just 2 Categories - "36 months" and "60 months". There is another category but it is just a missing value. Because there are just 2 categories, we need just 1 dummy variable. The State variable does not contain important information so we will discard it in this project.

The screenshot shows the RStudio interface with several panes open:

- Script Editor:** Displays a script named `loanAppFromLinearRegression.R` containing R code for data manipulation and analysis.
- Environment:** Shows the global environment with variables like `ld` (a data frame) and `ld$loan.length`.
- Files:** Browsing the "D:\Ghancer Education Services\Business Analytics Professional Course\Software & Datasets\Datasets\Datasets\Datasets" directory, listing files such as `whitequality-white.csv`, `whitequality-red.csv`, `Untitled.R`, `test.html`, `test.Rmd`, `temp.Rdata`, `souvenirs.csv`, `skirts.csv`, `rain.csv`, `OnlineNewsPopularity.csv`, `namesdaysstate`, `loans-data.csv`, `kings.csv`, `robobanfblobat`, `gamingfblobat`, `Existing_Bank.csv`, and `exercises.xls`.

Training the Model

Before we start our analysis, we need to divide our dataset into Training and Testing parts. The division that we are using is 70:30.

The screenshot shows the RStudio interface with the following details:

- Code Editor:** Displays the script `LoanAggregator.LinearRegression.R`. The code performs the following steps:
 - Converts the "Loan.Length" column into dummy variables.
 - Removes the "State" column from the dataset.
 - Creates a table for the state categories.
 - Creates a variable `ld` by selecting all columns except "State".
 - Breaks the data into training and test datasets. It uses `set.seed(2)` and `s = sample(1:nrow(ld), 0.7 * nrow(ld))` to generate a sample index. Then, it creates `ld_train` as the subset of `ld` where the index is in `s`, and `ld_test` as the subset where the index is not in `s`.
 - Prints a summary of the training dataset (`glimpse(ld_train)`).
- Console:** Shows the execution of the R code. It includes the creation of a seed, sampling rows, defining training and test datasets, and printing the first few rows of the training dataset.
- Environment:** Shows the global environment with variables `ld`, `ld_test`, `ld_train`, and `s`.
- Files:** Shows the file structure in the current directory, including CSV files like `winequality-white.csv` and `winequality-red.csv`, and R scripts like `Untitled.R` and `test.Rmd`.

We use the linear regression function with a focus on predicting "Interest Rate" keeping all the predictors except ID, as it will not contribute in any way to our prediction of rates. We specify the dataset on which we have to perform logistic regression (`ld_train`) and store the output in the variable "fit".

We can check the VIF values of all the predictors using the `vif()` function present in the `car` library and use it on the variable `fit`. (NOTE: You may have to install the `car` library from CRAN before you use VIF)

The screenshot shows the RStudio interface with the following components:

- Script Editor:** Displays the code for `LoanAggregator-LinearRegression.R`. The code includes:


```

99 glimpse(ld_train)
100
101 ## ----- Training the linear regression model on train data and storing it in fit -----
102
103 fit=lm(Interest.Rate~. -ID,data=ld_train)
104
105 ## ---- checking VIF values of variables -----
106 library(car)
107 vif(fit)
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127 "
128
129 Checking VIF values of variables
      
```
- Environment Browser:** Shows the global environment with objects like `fit`, `ld`, `ld_test`, `ld_train`, and `s`.
- Console:** Displays the output of the R session, including the results of the `vif(fit)` command.

```

> fit=lm(Interest.Rate~. -ID,data=ld_train)
> library(car)
> vif(fit)
      
```

Amount.Requested	Debt.To.Income.Ratio	Monthly.Income
1.601573	1.364146	1.402768
open.CREDIT.Lines	Revolving.CREDIT.Balance	Inquiries.in.the.Last.6.Months
1.344688	1.318871	1.045984
fico	eI	HW.RENT
1.153243	1.090594	105.600550
HW_MORT	HW.OWN	LP_CC
106.113545	29.936906	1.644014
LP_dc	LP_other	LL_36
1.833158	1.291791	1.257889

The variable `HW_MORT` has a very high VIF value (more than 5) so we will remove it from the next regression we perform. We then check the VIF again and see that they are well below the cautionary level of 5.

The screenshot shows the RStudio interface with the following components:

- Script Editor:** Displays the code for "MyPracticeSheet.R". The code includes training a linear regression model on the full dataset and then checking VIF values again after removing the "HW_MORT" variable.
- Environment Browser:** Shows the global environment with objects like `fit`, `ld`, `ld_test`, `ld_train`, and `s`.
- Console:** Shows the output of the R code. It includes the results of the lm() function and the vif() function, which prints a table of VIF values for various variables.

```

96 ld_train=ld[,]
97 ld_test=ld[-s,]
98 glimpse(ld_train)
99
100 ## ----- Training the linear regression model on train data and storing it in fit -----
101 fit=lm(Interest.Rate~. -ID,data=ld_train)
102
103 ## ---- Checking VIF values of variables -----
104 library(car)
105 vif(fit)
106
107 ## -----Removing HW_MORT and checking the VIF values again -----
108 fit=lm(Interest.Rate~. -ID - HW_MORT,data=ld_train)
109 vif(fit)
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125 < Removing HW_MORT and checking the VIF values again >
1119

```

Variable	Amount.Requested	Debt.To.Income.Ratio	Monthly.Income
Open.CREDIT.Lines	1.600472	1.362425	1.402430
fico	1.341196	Revolving.CREDIT.Balance	Inquiries.in.the.Last.6.Months
HW_OWN	1.151463	1.316826	1.045602
LP_other	1.104588	e1	HW.RENT
	1.291791	1.089516	1.251635
		LP_cc	LP_dc
		1.642075	1.832890
		LL_36	
		1.257672	

Next we check the p values of all the variables through the summary() function and remove those variables that have a p-value greater than 0.05 (Our acceptance level is still 0.95). This is easy when we have less variables but becomes tedious when we have hundreds of variables to choose from.

The screenshot shows the RStudio interface with the following components:

- Script Editor:** Displays a script named "MyPracticeSheet.R" containing R code for linear regression analysis.
- Environment Pane:** Shows the global environment with objects like `fit`, `ld`, `ld_test`, `ld_train`, and `s`.
- Console Pane:** Displays the output of the R code, including the coefficient table, standard error, t-value, p-value, and AIC values.

```

103 #> fit<-lm(Interest.Rate~.-ID-HW_MORT,data=ld_train)
104 ## ---- checking VIF values of variables -----
105 library(car)
106 vif(fit)
107
108 ## -----Removing HW_MORT and checking the VIF values again -----
109 fit<-lm(Interest.Rate~.-ID,data=ld_train)
110 vif(fit)
111
112 ## ----- checking summary for p values of variables -----
113 summary(fit)
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132 <
114:13 Checking summary for p values of variables <

```

Console Output:

```

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 7.612e+01 1.140e+00 66.795 < 2e-16 ***
Amount.Requested 1.601e-04 8.057e-06 19.867 < 2e-16 ***
Debt.To.Income.Ratio 1.683e-03 7.770e-03 0.217 0.82858
Monthly.Income -1.839e-05 1.392e-05 -1.321 0.18661
Open.CREDIT.Lines -3.744e-02 1.285e-02 -2.914 0.00361 **
Revolving.CREDIT.Balance -4.814e-06 3.093e-06 -1.557 0.11977
Inquiries.in.the.Last.6.Months 3.216e-01 4.201e-02 7.655 3.27e-14 ***
fico -8.799e-02 1.514e-03 -58.136 < 2e-16 ***
el 1.239e-02 1.460e-02 0.849 0.39601
HW.RENT 2.529e-01 1.124e-01 2.250 0.02461 *
HW.OWN 4.803e-01 2.009e-01 2.390 0.01694 *
LP_CC -5.136e-01 1.697e-01 -3.026 0.00252 **
LP_dc -3.590e-01 1.357e-01 -2.646 0.00823 **
LP_other 1.406e-01 2.090e-01 0.673 0.50111
LL_36 -3.161e+00 1.343e-01 -23.543 < 2e-16 ***
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.048 on 1662 degrees of freedom
Multiple R-squared: 0.7658, Adjusted R-squared: 0.7638
F-statistic: 388.1 on 14 and 1662 DF, p-value: < 2.2e-16

```

What makes this step automated is the function `step()`, which checks on the AIC score of each variable.

$$AIC = \text{Loss function} + 2 * \text{No. of variables}$$

The Loss function we use here is Error Sum of Squares (SSE). Higher number of variables will lead to explanation of variance and thus decrease in Loss Function. In a nutshell, lower AIC value variables will be kept and higher ones will be discarded. Once all the variables have been dropped, we can check the summary again and drop the variables with higher p-values.

Rather than writing variables manually in the linear regression function, we can use the formula() function to get the dependent and predictor variables. We can copy paste these variables to the linear regression function and then remove the variables with high p values. Here, we will remove the Monthly Income variable, because it has a high p-value.

The screenshot shows the RStudio interface with the following components:

- Script Editor (Top Left):** Displays the code for `LoanAggregator-LinearRegression.R*`. The code includes various steps for model selection and fitting, such as removing variables based on VIF values, using step and formula functions, and finally fitting a linear regression model with specific variables.
- Environment (Top Right):** Shows the global environment with objects like `fit`, `ld`, `ld_test`, `ld_train`, and `s`.
- Console (Bottom Left):** Shows the output of the R code, including the summary of the fitted model. The summary table provides coefficients for variables like `Open.CREDIT.Lines`, `Revolving.CREDIT.Balance`, `Inquiries.in.the.Last.6.Months`, `fico`, `HW.RENT`, `HW.OWN`, `LP.cc`, `LP.dc`, and `LL.36`. It also shows the standard error, R-squared, and F-statistic.
- Packages (Bottom Right):** Shows the system library with many packages listed, including `assertthat`, `BH`, `bindr`, `bindrcpp`, `boot`, `car`, `class`, `cluster`, `codetools`, `compiler`, `datasets`, `dplyr`, `foreign`, `glue`, `graphics`, and `grDevices`.

```

107 vif(fit)
108
109 ## -----Removing HW_MORT and checking the VIF values again -----
110 fit=lm(Interest.Rate~. -ID - HW_MORT,data=ld_train)
111 vif(fit)
112
113 ## ----- Checking summary for p values of variables -----
114 summary(fit)
115
116
117 ## ----- Using step function to remove variables with high AIC values -----
118 fit = step(fit)
119
120 ## ----- Checking summary for p values of variables -----
121 summary(fit)
122
123 ## ----- Using formula function to get all the variables of the model -----
124 formula(fit)
125
126 ## ----- Running linear regression agian on filtered variables (removed Monthly Income) -----
127 fit = lm(Interest.Rate ~ Amount.Requested + Open.CREDIT.Lines +
128           Revolving.CREDIT.Balance + Inquiries.in.the.Last.6.Months +
129           fico + HW.RENT + HW.OWN + LP.cc + LP.dc + LL.36, data = ld_train)
130
131
132
133
134
135
136 < Running linear regression agian on filtered variables (removed Monthly Income) >
129: fit = lm(Interest.Rate ~ Amount.Requested + Open.CREDIT.Lines +
128:           Revolving.CREDIT.Balance + Inquiries.in.the.Last.6.Months +
127:           fico + HW.RENT + HW.OWN + LP.cc + LP.dc + LL.36, data = ld_train)

> formula(fit)
Interest.Rate ~ Amount.Requested + Monthly.Income + Open.CREDIT.Lines +
  Revolving.CREDIT.Balance + Inquiries.in.the.Last.6.Months +
  fico + HW.RENT + HW.OWN + LP.cc + LP.dc + LL.36
> fit = lm(Interest.Rate ~ Amount.Requested + Open.CREDIT.Lines +
+   Revolving.CREDIT.Balance + Inquiries.in.the.Last.6.Months +
+   fico + HW.RENT + HW.OWN + LP.cc + LP.dc + LL.36, data = ld_train)

```

Upon checking the summary again, we can see a column called estimate, which shows the kind of correlation that variable has with the dependent variable i.e. Interest rate. Amount Requested is positively correlated with Interest Rate: higher the amount requested, higher will be the rate. These are basically the coefficients (beta). So, can we say that 1 coefficient is more valuable than the other based on its value? The answer is NO, and that is because we must take into account the size of the values in the variables as well. Amount Requested has values in thousands whereas Open Credit Lines is at the most in double-digits. We need to put this into perspective and scale up or down the importance of each variable and standardize before comparing coefficients.

The screenshot shows the RStudio interface with the following components:

- Script Editor:** Displays the code for `LoanAggregator-LinearRegression.R`. The code includes steps for removing variables with high AIC values, checking p-values, and extracting all variables from the model. It then runs a linear regression model on filtered variables and checks the summary of the estimates.
- Environment Browser:** Shows the global environment with objects like `fit`, `ld`, `ld_test`, `ld_train`, and `s`.
- Console Output:** Shows the results of the `summary(fit)` command, which includes:

	Min	1Q	Median	3Q	Max
Amount.Requested	-6.5381	-1.3666	-0.1691	1.1370	9.9261

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	7.641e+01	1.100e+00	69.434	< 2e-16 ***
Amount.Requested	1.571e-04	7.644e-06	20.555	< 2e-16 ***
Open.CREDIT.Lines	-3.676e-02	1.204e-02	-3.052	0.00231 **
Revolving.CREDIT.Balance	-5.692e-06	2.956e-06	-1.926	0.05433 .
Inquiries.in.the.Last.6.Months	3.199e-01	4.195e-02	7.625	4.07e-14 ***
fico	-8.830e-02	1.486e-03	-59.405	< 2e-16 ***
HW.RENT	2.537e-01	1.091e-01	2.326	0.02012 *
HW.OWN	4.839e-01	1.993e-01	2.428	0.01527 *
LP_cc	-5.196e-01	1.577e-01	-3.295	0.00101 **
LP_dc	-3.722e-01	1.214e-01	-3.065	0.00221 **
LL_36	-3.182e+00	1.335e-01	-23.842	< 2e-16 ***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.048 on 1666 degrees of freedom
Multiple R-squared: 0.7653, Adjusted R-squared: 0.7639
F-statistic: 543.3 on 10 and 1666 DF, p-value: < 2.2e-16

If we use `plot()` function to check the fitted values and their residuals, we see a non-linear plot. This shows us that one (or more) of the variables are not linearly correlated with Interest Rate. (Check “Residuals VS Fitted plot.png”)

The screenshot shows the RStudio interface with the following components:

- Top Bar:** RStudio, File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Left Panel:** Project Explorer, Source Editor (containing R code for stepwise regression), and Console (output of the R code).
- Right Panel:** Environment View showing variables (fit, Id, id_test, id_train, s) and a Residuals vs Fitted plot.
- Bottom Panel:** A scatter plot titled "Residuals vs Fitted" showing residuals versus fitted values.

R Code in Source Editor:

```
## ----- using step function to remove variables with high AIC values -----
118 fit = step(fit)
119
120 ## Checking summary for p values of variables
121 summary(fit)
122
123 ## ----- using formula function to get all the variables of the model -----
124 formula(fit)
125
126 ## ----- running linear regression again on filtered variables (removed Monthly income) -----
127 fit = lm(Interest.Rate ~ Amount.Requested + Open.CREDIT.Lines +
128 Revolving.CREDIT.Balance + Inquiries.in.the.Last.6.Months +
129 fico + Hh.Rent + Hh.Own + Lp.Lcc + Lp.Dcc + LL_36, data = id_train)
130
131 ## ----- checking summary for Estimates of variables -----
132 summary(fit)
133
134 ## ----- Checking plot between the Fitted values and Residuals -----
135 plot(fit, which = 1)
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
```

Console Output:

```
## Checking plot between the Fitted values and Residuals: 1
[1] "Residuals vs Fitted"
[1] "Fitted values"      [1] "Residuals"        [1] "Residuals vs Fitted"
```

Environment View:

Name	Type	Length	Size	Value
fit	lm	12	649.3 KB	Large lm (12 elements, 649.3 kB)
Id	data.frame	17	301.9 KB	2396 obs. of 17 variables
id_test	data.frame	17	95.1 KB	719 obs. of 17 variables
id_train	data.frame	17	218.6 KB	1677 obs. of 17 variables
s	integer	1677	6.6 kB	int [1:1677] 443 1683 1373 403 2258 ...

Residuals vs Fitted Plot:

A scatter plot titled "Residuals vs Fitted" showing the relationship between residuals and fitted values. The x-axis is labeled "Fitted values" and ranges from approximately 0 to 25. The y-axis is labeled "Residuals" and ranges from -10 to 10. The data points show a clear negative trend, indicating a non-linear relationship. A red line represents a linear regression fit through the data.

Upon going through the Smooth Line Scatter plot of FICO score VS Interest Rate, we see that, barring the extremes, their relationship is not Linear. Their relationship appears to be almost square like. (Check “Fico VS Interest Rate.png”)

One way out of this would be to convert fico to squared value and then fit the training data on it.

The screenshot displays the RStudio interface with several panes:

- Code pane:** Shows R code for linear regression analysis, including steps to remove variables with high AIC values, fit the model, and check for multicollinearity using the VIF function.
- Environment pane:** Lists global variables: `fit` (lm), `ld_test` (data.frame), `ld_train` (data.frame), and `s` (integer).
- Plots pane:** Displays a scatter plot of `fico` (X-axis, 650-800) versus `Interest.Rate` (Y-axis, 5-15). The plot shows a non-linear relationship with a blue line and a grey shaded area representing confidence intervals.
- Console pane:** Shows the results of running the R code, including package loading and the creation of the `ld_train` and `ld_test` data frames.

The next plot checks for the Normality of the model. We can see that most of the values fall on the reference line, except for some extreme values. (Check “Normality Plot.png”)

The figure shows a screenshot of the RStudio interface. On the left, the code editor displays R code for performing a linear regression analysis. The code includes steps for fitting the model, summarizing it, and creating plots of fitted values vs residuals and FICO vs Interest Rate. It also includes a check for the normality of variables. The right side of the interface shows the 'Environment' pane listing objects like `fit` (an lm object), `Id` (a data frame), and `s` (an integer). Below it is the 'Plots' pane, which features a 'Normal Q-Q' plot. This plot has 'Standardized residuals' on the y-axis and 'Theoretical Quantiles' on the x-axis. A solid black line represents the data points, and a dashed diagonal line represents the expected normal distribution. The data points show a strong positive linear correlation, indicating non-normality.

Upon plotting the density function for comparison with a Standard Normal Deviation plot, we see that there is just a slight deviation in our data, which is no cause for concern. (Check “Normality Comparison.png”)

The figure shows a screenshot of the RStudio IDE. The top menu bar includes File, Edit, Code View, Plots, Session, Build, Debug, Profile, Tools, Help, and Go to Reference. The left sidebar shows a project named 'MyPredictiveModel' with files like 'loanAggregationLinearRegression.R'. The main workspace contains R code for data loading, model fitting, and diagnostic plots. The code includes:

```
128 Revolving.CREDIT.Balance ~ Inquiries.in.The.Last.6.Months +
129 fico + HW.RENT + HW.OWN + LP_cc + LP_dc + LL_3b, data = ld_train)
130
131 ## ----- Checking summary for Estimates of variables -----
132 summary(fit)
133
134 ## ----- checking plot between the Fitted values and Residuals -----
135 plot(fit, which = 1)
136
137 ## ----- checking plot between the FICO and Interest Rate -----
138 library(ggplot2)
139 ggplot(ld_train, aes(x = fico, y = interest.Rate)) + geom_smooth()
140
141 ## ----- checking plot for Normality of the Variables -----
142 plot(fit, which = 2)
143
144 ## ----- Checking density plots in comparison with standard normal deviation -----
145 df$dfires$fit$residuals
146 ggplot(df$dfires$residuals) + geom_density(color="red")+
147 stat_function(fun=dnorm, args=list(mean=mean(df$dfires), sd=sd(df$dfires)),color="green")
148
149
150
151
152
153
154
155
156
157 <
```

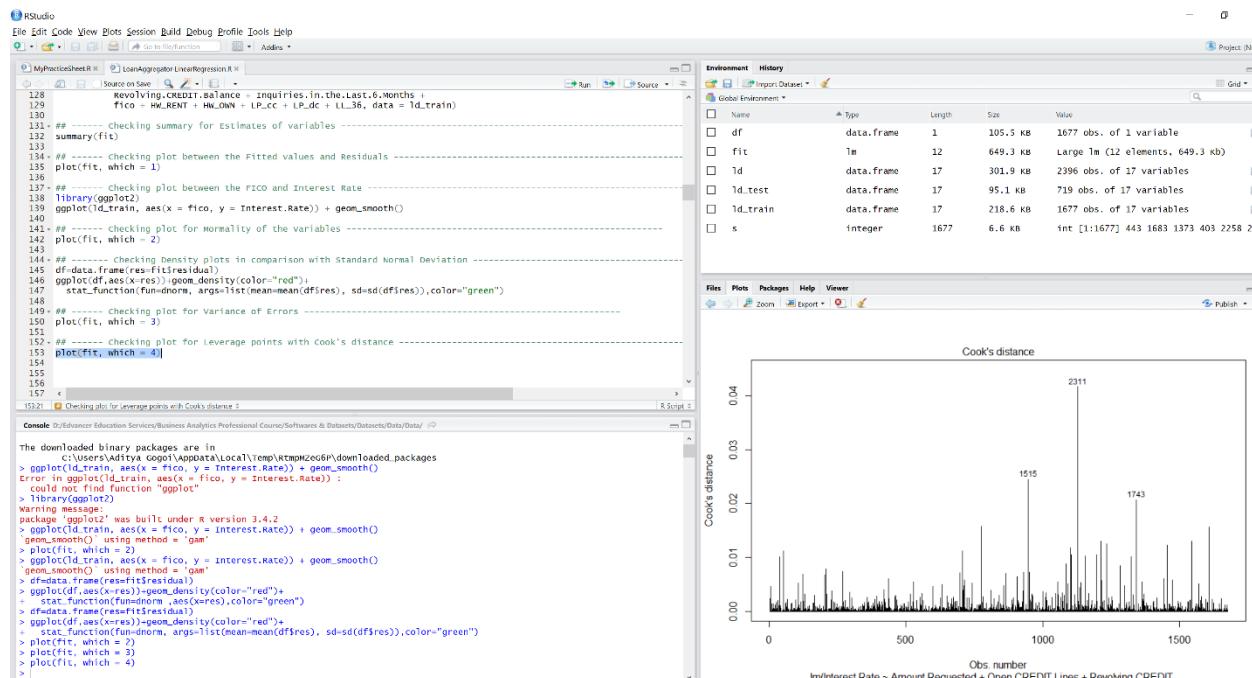
The bottom left pane shows the R Console output, which includes messages about package installations and density plot comparisons. The bottom right pane displays a density plot titled 'Checking Density plots in comparison with Standard Normal Deviations'. It features two bell-shaped curves: a red curve representing the empirical distribution of residuals and a green curve representing the standard normal distribution.

Next, we will check for variance in errors. For all purposes, the plot shows a constant variance in the errors. We see that the scatter is consistent and the thread line is almost linear. Although ideally, it should have been completely flat. (Check “Variance in Errors.png”)

The screenshot shows an RStudio environment with the following details:

- Top Bar:** File, Edit, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Left Panel:** Script pane containing R code for linear regression. The code includes loading data from 'inquiries.in the.last.6.months', fitting a model, summarizing estimates, plotting residuals, checking normality, and creating density plots for residuals.
- Middle Panel:** Console output showing the unpacking of the 'mosaic' package and the successful execution of the R code.
- Right Panel:** Environment pane showing variables: df (data.frame), fit (lm), id (data.frame), id_test (data.frame), id_train (data.frame), and s (integer).
- Bottom Panel:** Plots pane displaying a Scale-Location plot titled 'Scale-Location'. The y-axis is labeled '(Standardized residual)^2' and ranges from 0.0 to 2.0. The x-axis is labeled 'Fitted values' and ranges from 0 to 20. The plot shows a clear upward trend, indicating heteroscedasticity.

We will check for outliers. Certain outliers cause the regression line to shift just slightly from the original incline, because they are already in the same line as the other data points. But some outliers really affect the regression line by a large degree, and these outliers are called 'leverage points'. We need to find out the leverage points and discard them from our dataset. We will use Cook's Distance to see how much does a regression line's inclination change by the absence of a point. If it varies a lot, then that point must be discarded. For our dataset, we see that none of the points have a Cook's distance greater than 1. (Check "Cook's Distance.png")



Now that we are sure that the data and model is optimal, we will predict on the test data, get its difference from original values, square it, get its mean and then get its square root. So basically, the Root Mean Square Error (RMSE). The predict() function needs 2 parameters - the model and the data on which I am predicting. The RMSE for this model came at 2.01842.

The screenshot shows the RStudio interface with the following details:

- Code Editor:** Displays the script `LoanAggregator-LinearRegression.R` containing R code for model validation. The last few lines of the script calculate the RMSE:

```

156 mean((ld_test$Interest.Rate-predict(fit,newdata=ld_test))**2) %>%
157 sqrt()

```

- Console:** Shows the execution of the script and the output of the RMSE calculation:

```

> ggpplot(ld_train, aes(x = fico, y = Interest.Rate)) + geom_smooth()
Error in ggpplot(ld_train, aes(x = fico, y = Interest.Rate)) :
  could not find function "ggplot"
> library(ggplot2)
Warning message:
package 'ggplot2' was built under R version 3.4.2
> ggpplot(ld_train, aes(x = fico, y = Interest.Rate)) + geom_smooth()
`geom_smooth()` using method = 'gam'
> plot(fit, which = 2)
> ggpplot(ld_train, aes(x = fico, y = Interest.Rate)) + geom_smooth()
`geom_smooth()` using method = 'gam'
> df=data.frame(res=fit$residual)
> ggpplot(df,aes(x=res))+geom_density(color="red")+
+ stat_function(fun=dnorm ,aes(x=res),color="green")
> df=data.frame(res=fit$residual)
> ggpplot(df,aes(x=res))+geom_density(color="red")+
+ stat_function(fun=dnorm ,args=list(mean=mean(df$res), sd=sd(df$res)),color="green")
> plot(fit, which = 2)
> plot(fit, which = 3)
> plot(fit, which = 4)
> mean((ld_test$Interest.Rate-predict(fit,newdata=ld_test))**2) %>%
+ sqrt()
[1] 2.01842
>

```

- Environment:** Shows the global environment variables: `df`, `fit`, `ld`, `ld_test`, `ld_train`, and `s`.
- Plots:** A histogram titled "Cook's distance" showing the distribution of Cook's distances for the data points. The x-axis ranges from 0 to 1, and the y-axis ranges from 0.00 to 0.04. The distribution is highly right-skewed with a few large peaks.

Conclusion

The regression model we created has a decent RMSE of ~ 2. But this can be improved by including some variables (like state, which might affect the loan interest rate) and making improvements on them.