IDS-572: ASSIGNMENT-2

Lending Club: Part A

FEBRUARY 26, 2023

RHEA DSOUZA
NIRMAL UMARIA
NISHIMA PURI

PART A

Question 1:

(a) Your team's goal is to help clients determine whether they should invest in p2p loans and which loans to invest in. What is the objective, and how will you evaluate 'better' vs 'worse' decisions? What is the goal of using predictive models for this? What will be the potential target variable(s)?

The goal of assisting clients to choose which loans to invest in and whether they should invest in P2P loans is to maximize returns while lowering risks. **We'll measure "better" choices as ones that maximize returns on investment while lowering the danger of default.** P2P loans offer a variety of investment choices for investors based on their level of risk tolerance. Investors who are willing to take on a lot of risk can engage in lower rated loans, which have higher returns but also higher default risks. Investors who are ready to assume more risk in exchange for possibly higher returns may find these loans to be more suitable.

Conversely, those who are more careful with their money can opt to engage in high-rated loans that have a lower chance of defaulting. Investors who value capital preservation and are prepared to accept lower returns in exchange for a lower default risk may find these loans to be more appropriate.

P2P loans provide a versatile financial choice that can be customized to the investor's particular risk tolerance. Depending on their risk appetite and financial objectives, investors can select from a variety of loan grades to create a portfolio that is unique to them. A choice that is considered to be **"better"** is one that produces the anticipated return or a return that is higher than that of other possible investment sources.

Predictive models are used in this situation to find the debts with the highest likelihood of producing profitable returns with the lowest possible default risk. By taking into consideration elements like credit history, debt-to-income ratio, job status, and loan purpose, the predictive model will assist us in assessing the borrower's creditworthiness and the risk level associated with each loan.

The probability of default or the anticipated rate of return for each loan could be the **possible target variables** for our predictive models. These goal variables allow us to spot loans that are likely to default or have poor expected returns and steer clear of them. Instead, we can prioritize investing in loans with low default risk or high expected returns by using these goal variables to find those loans.

(b) Take a look at the data attributes. How would you categorize these attributes, in broad terms, considering what they pertain to?
Before doing any analyses, what do you think may be some of the important attributes to consider for your decision task?
The data attributes that pertain to P2P lending can be broadly categorized into three categories:

Borrower attributes:
These characteristics include the borrower's age, gender, income, job situation, and credit history and are related to their demographic data.

Loan attributes:
These characteristics, such as loan amount, interest rate, loan purpose, loan duration, and loan grade, are related to the features of the loan.

Performance attributes:
These characteristics, which include the loan status, whether it was completely paid or in default, and the time to default, are related to how well the loan performed.

The loan grade, interest rate, loan amount, loan duration, borrower credit history, and loan purpose are some of the crucial factors to take into account when considering these attributes for a decision-making task. These characteristics can be used to assess each loan's risk level and possible return. To monitor the performance of the loan and take required action if necessary, it may also be essential to take into account the loan status and time to default. Finally, demographic data on the borrower can be used to gauge the borrower's creditworthiness and determine their capacity to repay the debt.

Question 2-

```
library(tidyverse)
library(lubridate)
library(data.table)
library(broom)
options(dplyr.summarise.inform = FALSE)

Lcdf <- read_csv('C:/Users/rhea/OneDrive/Desktop/UIC SEM 2/IDS 572 Data Mining/Assignment 2/lcDataSample.csv')
Rows: 110000 Columns: 145
── Column specification ──────────────────────────────────────────────────────────────────────────────────────────────
Delimiter: ","
chr  (22): term, grade, sub_grade, emp_title, emp_length, home_ownership, verification_status,
           loan_status, pymnt_plan, purpose, title, zip_code, addr_state, ear...
dbl  (84): loan_amnt, funded_amnt, funded_amnt_inv, int_rate, installment, annual_inc, dti, delinq_2yrs,
           inq_last_6mths, mths_since_last_delinq, mths_since_last_...
lgl  (38): id, member_id, url, desc, annual_inc_joint, dti_joint, verification_status_joint, revol_bal_joint,
           sec_app_earliest_cr_line, sec_app_inq_last_6mths, s...
dttm  (1): issue_d
```

#How many loans are fully-paid and charged-off?

```
Lcdf %>% group_by(loan_status) %>% tally()

# A tibble: 2 × 2
  loan_status     n
  <chr>           <int>
1 Charged Off    15377
2 Fully Paid     94567


#Are there values for loan_status other than "Fully Paid' and "Charged Off"?  If so, remove them:
Lcdf <- Lcdf %>% filter(loan_status == "Fully Paid" | loan_status == "Charged Off")

#How does loan status vary by loan grade
```

The output shows that the number of Charged Off loans is generally higher in lower grades (A being the lowest and G being the highest), while the number of Fully Paid loans is generally higher in higher grades. This suggests that higher-grade loans are generally more likely to be fully paid, while lower-grade loans are more likely to default or be charged off.

```
Lcdf %>% group_by(loan_status, grade) %>% tally()

   loan_status    grade     n
   <chr>          <chr>   <int>
 1 Charged Off    A        1369
 2 Charged Off    B        4264
 3 Charged Off    C        5206
 4 Charged Off    D        3165
 5 Charged Off    E        1090
 6 Charged Off    F         252
 7 Charged Off    G          31
 8 Fully Paid     A       23485
 9 Fully Paid     B       33601
10 Fully Paid     C       23939
11 Fully Paid     D       10290
12 Fully Paid     E        2700
13 Fully Paid     F         501
14 Fully Paid     G          51

Lcdf %>% group_by(sub_grade,loan_status) %>% tally()
# A tibble: 69 × 3
# Groups:   sub_grade [35]
   sub_grade loan_status     n
   <chr>     <chr>           <int>
 1 A1        Charged Off     104
```

```
 2 A1       Fully Paid       3934
 3 A2       Charged Off      161
 4 A2       Fully Paid       3745
 5 A3       Charged Off      193
 6 A3       Fully Paid       3852
 7 A4       Charged Off      385
 8 A4       Fully Paid       5390
 9 A5       Charged Off      526
10 A5        Fully Paid      6564
```

The output shows the count of Charged Off and Fully Paid loans for each sub_grade, which provides more detailed information than the previous code which grouped only by grade. It suggests that sub_grades within each grade have different default rates and performance levels. For example, sub_grades A1, A2, A3, A4 and A5 within grade A have different numbers of Charged Off and Fully Paid loans, with A1 having the lowest number of defaults and A5 having the highest. This can provide insights for risk assessment and loan pricing.

A(ii)How many loans are there in each grade? And do loan amounts vary by grade?

```
#How does number of loans, loan amount, interest rate vary by grade
Lcdf %>% group_by(grade) %>% tally()
# A tibble: 7 × 2
  grade     n
  <chr> <int>
1 A     24854
2 B     37865
3 C     29145
4 D     13455
5 E      3790
6 F       753
7 G        82


Lcdf %>% group_by(grade) %>% summarise(sum(loan_amnt))
# A tibble: 7 × 2
  grade `sum(loan_amnt)`
  <chr>           <dbl>
1 A           356633075
2 B           473544750
3 C           351136175
4 D           160060000
5 E            45192200
6 F             7104350
7 G              947125
```

Does interest rate for loans vary with grade, subgrade?

```
Lcdf %>% group_by(grade) %>% summarise(mean(int_rate))
# A tibble: 7 × 2
  grade `mean(int_rate)`
  <chr>           <dbl>
1 A                7.21
2 B               10.9
3 C               13.9
4 D               17.3
5 E               20.0
6 F               23.9
7 G               26.5
```

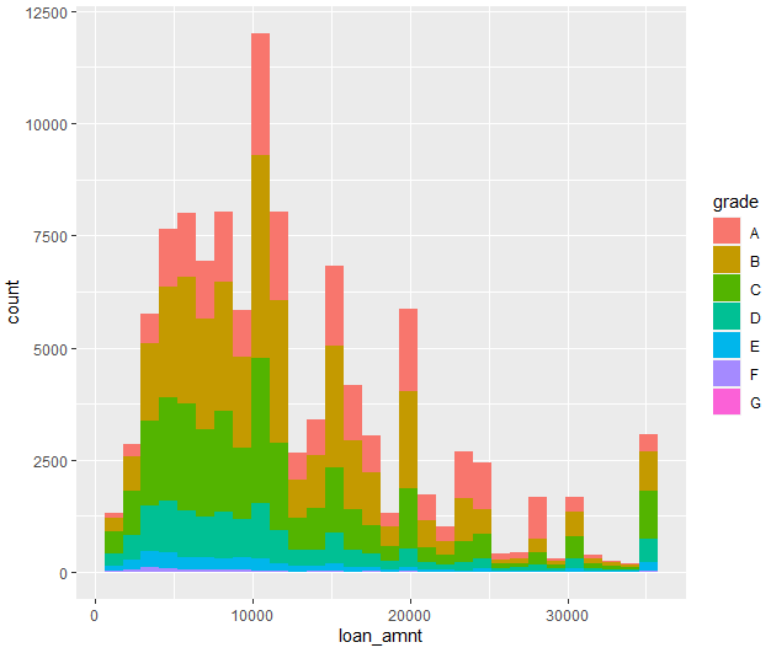
The output shows that the mean int_rate increases as the grade of the loan decreases. Grade A loans have the lowest mean int_rate of 7.21%, while grade G loans have the highest mean int_rate of 26.5%. This indicates that investors demand higher returns for investing in lower-grade loans, which are perceived as riskier.

```
Lcdf %>% group_by(sub_grade) %>% summarise(mean(int_rate))
# A tibble: 35 × 2
   sub_grade `mean(int_rate)`
   <chr>              <dbl>
 1 A1                  5.70
 2 A2                  6.43
 3 A3                  7.14
 4 A4                  7.52
 5 A5                  8.28
 6 B1                  8.96
 7 B2                 10.0
 8 B3                 11.0
 9 B4                 11.8
10 B5                  12.4
```

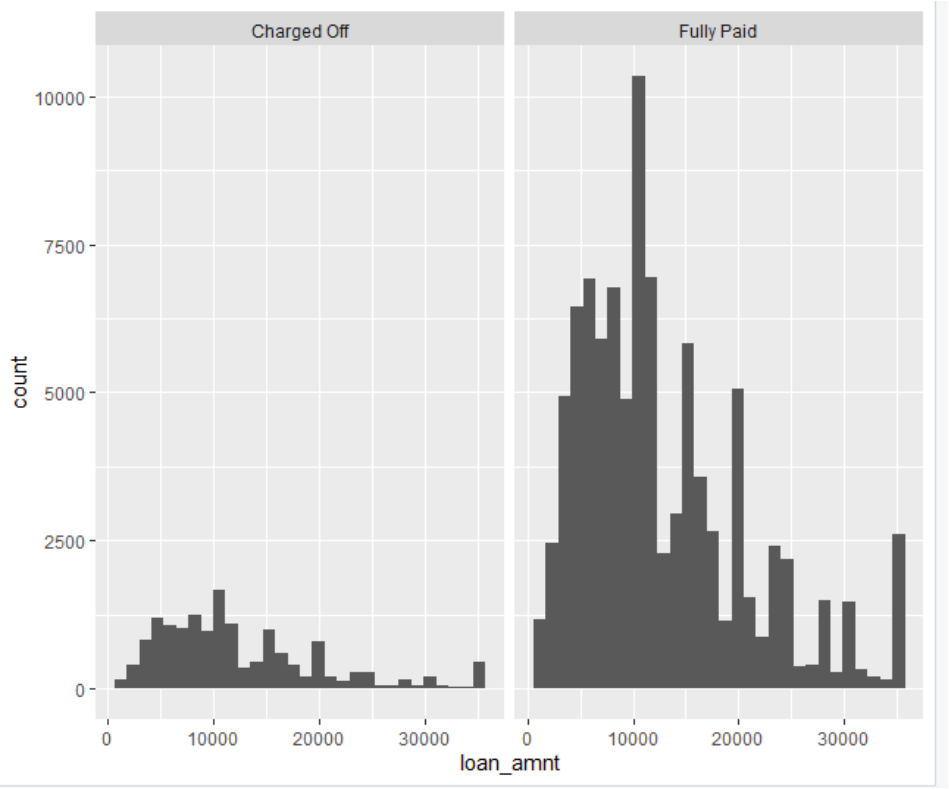
The output shows that the mean int_rate generally increases as the sub_grade of the loan decreases. Sub_grades A1 to A5 have the lowest mean int_rates, while sub_grades G2 to G5 have the highest mean int_rates. This suggests that investors demand higher returns for investing in loans with lower sub_grades, which are perceived as riskier.

The output of this code provides more detailed information than the code that grouped only by grade. It highlights the fact that sub_grades within each grade can have different interest rates. For example, within grade A, the mean int_rate increases from 5.70% for sub_grade A1 to 8.28% for sub_grade A5. This can be useful for risk assessment and loan pricing, as different sub_grades may require different interest rates to compensate for different levels of risk.

```
ggplot(LC_Data, aes( x = loan_amnt)) + geom_histogram(aes(fill=grade))
```



ggplot(LC_Data, aes(x = loan_amnt)) + geom_histogram() + facet_wrap(~loan_status)



Summary for Average and standard-deviation of Interest rate by grade and subgrade.

LC_Data%>% group_by(grade) %>% summarise(numLoans=n(), avgInterest = mean(int_rate), std_dev_Interest = sd(int_rate))

grade	numLoans	avgInterest	std_dev_Interest
<chr>	<int>	<dbl>	<dbl>
1 A	24854	7.21	0.973
2 B	37865	10.9	1.48
3 C	29145	13.9	1.23
4 D	13455	17.3	1.22
5 E	3790	20.0	1.40
6 F	753	23.9	0.955
7 G	82	26.5	0.958

The output shows that the average interest rate generally increases as the grade of the loan decreases, with grade G having the highest average interest rate of 26.5%. The output also shows that the standard deviation of interest rates is highest for grade B and lowest for grade G. This suggests that there is more variability in interest rates for grade B loans, while grade G loans have relatively consistent interest rates.

LC_Data%>% group_by(sub_grade) %>% summarise(numLoans=n(), avgInterest = mean(int_rate), std_dev_Interest = sd(int_rate))

sub_grade	numLoans	avgInterest	std_dev_Interest
<chr>	<int>	<dbl>	<dbl>
1 A1	4038	5.70	0.348
2 A2	3906	6.43	0.167
3 A3	4045	7.14	0.341
4 A4	5775	7.52	0.360
5 A5	7090	8.28	0.438
6 B1	6860	8.96	0.757
7 B2	7698	10.0	0.832
8 B3	8283	11.0	0.923
9 B4	7760	11.8	0.891
10 B5	7264	12.4	0.942

LC_Data%>% group_by(grade,sub_grade) %>% summarise(mean_intRate = mean(int_rate))

grade	sub_grade	mean_intRate
<chr>	<chr>	<dbl>
1 A	A1	5.70
2 A	A2	6.43
3 A	A3	7.14
4 A	A4	7.52
5 A	A5	8.28
6 B	B1	8.96
7 B	B2	10.0
8 B	B3	11.0
9 B	B4	11.8

10 B B5 12.4

Minimum interest rates for each grades and subgrades

```
LC_Data%>% group_by(grade) %>% summarize(min(int_rate))
grade  `min(int_rate)`
  <chr>          <dbl>
1 A               5.32
2 B               6
3 C               6
4 D               6
5 E               6
6 F              20.9
7 G              24.9

LC_Data%>% group_by(sub_grade) %>% summarize(min(int_rate))

sub_grade `min(int_rate)`
   <chr>              <dbl>
 1 A1                 5.32
 2 A2                 6.24
 3 A3                 6.68
 4 A4                 6.92
 5 A5                 6
 6 B1                 8.18
 7 B2                 6
 8 B3                 6
 9 B4                 6
10 B5                  6
```

Maximum interest rates for each grades and subgrades

```
LC_Data%>% group_by(grade) %>% summarize(max(int_rate))
grade `max(int_rate)`
  <chr>          <dbl>
1 A               9.25
2 B              14.1
3 C              17.3
4 D              20.3
5 E              23.4
6 F              26.0
7 G              29.0

LC_Data%>% group_by(sub_grade) %>% summarize(max(int_rate))
sub_grade `max(int_rate)`
   <chr>              <dbl>
 1 A1                 6.03
 2 A2                 6.97
 3 A3                 7.62
 4 A4                 8.6
```

```
 5 A5              9.25
 6 B1             10.2
 7 B2             11.1
 8 B3             12.1
 9 B4             13.1
10 B5              14.1
```

```
LC_Data%>%  group_by(grade)  %>%  summarise(nLoans=n(),  defaults=sum(loan_status=="Charged  Off"),
defaultRate=defaults/nLoans,  avgInterest=  mean(int_rate),stdInterest=sd(int_rate),  avgLoanAMt=mean(loan_amnt),
avgPmnt=mean(total_pymnt))
```

```
# A tibble: 7 × 8
  grade nLoans defaults defaultRate avgInterest stdInterest avgLoanAMt avgPmnt
  <chr>  <int>    <int>       <dbl>       <dbl>       <dbl>     <dbl>   <dbl>
1 A      24854     1369      0.0551        7.21       0.973     14349.  15388.
2 B      37865     4264      0.113        10.9        1.48      12506.  13634.
3 C      29145     5206      0.179        13.9        1.23      12048.  13094.
4 D      13455     3165      0.235        17.3        1.22      11896.  12831.
5 E       3790     1090      0.288        20.0        1.40      11924.  12624.
6 F        753      252      0.335        23.9        0.955      9435.   9953.
7 G         82       31      0.378        26.5        0.958     11550.  11836.
```

Question 2-a-(iii)----------

For the fully paid loans, it is noticeable that the time-to-full-payoff is maximum in the case of 3 years for approximately 22000 Fully Paid loans issued. The range goes till 4 years in the case of Fully Paid loans. We also notice that the actual loan term keeps on decreasing as the loan grade goes from C through G, while it increases as we go from A to C. It is also noticeable that approximately 75% of the population repay their loan in the term of 3 years while some of them reaching up to 4 years in the case of grade C

#Data For loans fully paid - time-to-payoff

```
head(LC_Data[, c("last_pymnt_d", "issue_d")])
  last_pymnt_d        issue_d
  <dttm>              <dttm>
1 2018-03-01 00:00:00 2015-03-01 00:00:00
2 2015-03-01 00:00:00 2014-05-01 00:00:00
3 2018-09-01 00:00:00 2015-09-01 00:00:00
4 2015-06-01 00:00:00 2014-05-01 00:00:00
5 2017-06-01 00:00:00 2015-05-01 00:00:00
6 2018-10-01 00:00:00 2015-11-01 00:00:00
```

```
LC_Data$last_pymnt_d<-paste(LC_Data$last_pymnt_d, "-01", sep = "")
```

 # Then convert this character to a date type variable

```
LC_Data$last_pymnt_d<-parse_date_time(LC_Data$last_pymnt_d,  "myd")
```

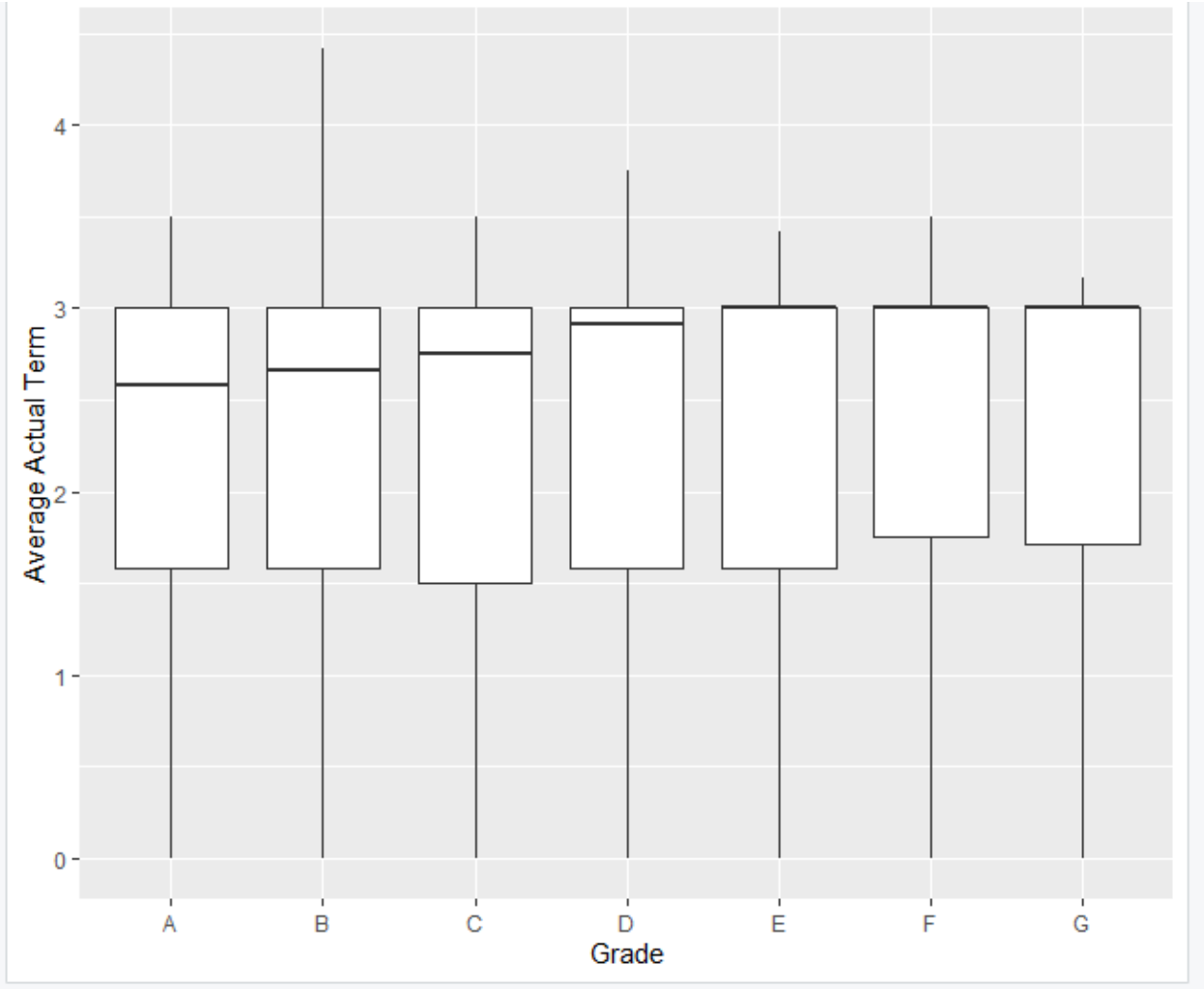
```
head(LC_Data[, c("last_pymnt_d", "issue_d")])
```

```
last_pymnt_d issue_d
  <dttm>      <dttm>
1 NA          2015-03-01 00:00:00
2 NA          2014-05-01 00:00:00
3 NA          2015-09-01 00:00:00
4 NA          2014-05-01 00:00:00
5 NA          2015-05-01 00:00:00
6 NA          2015-11-01 00:00:00
```

```
LC_Data$actualTerm <- ifelse(LC_Data$loan_status=="Fully Paid",as.duration(LC_Data$issue_d%--%
LC_Data$last_pymnt_d)/dyears(1), 3)
```

```
head(LC_Data$actualTerm)
[1] NA NA NA NA NA NA
```

```
ggplot(LC_Data, aes(x=actualTerm, y=grade)) + geom_boxplot()+coord_flip()+labs(y="Grade", x = "Average Actual Term")
```




```
dim(LC_Data)
```

[1] 110000 146

2 a- (iv)Calculate the annual return for a loan. Show how you calculate the percentage annual return.

lcdf %>% select(loan_status, int_rate, funded_amnt, total_pymnt) %>% head()

lcdf$annRet <- ((lcdf$total_pymnt -lcdf$funded_amnt)/lcdf$funded_amnt)*(12/36)*100

```
# A tibble: 6 x 4
  loan_status int_rate funded_amnt total_pymnt
  <chr>          <dbl>       <dbl>       <dbl>
1 Fully Paid     23.0         4400       6120.
2 Fully Paid     22.0         5850       6377.
3 Fully Paid      6.24        5000       5496.
4 Fully Paid     15.0         1600       1840.
5 Fully Paid      9.17       16000      18128.
6 Fully Paid      8.18        3000       3394.
```

The Data above shows the idea of how much is the variation of interest rate of certain fully paid loans and it also shows the amount of money funded which is the principal amount and then the total payment shows the amount with the interest added to get an overview of the total payments.

```
# A tibble: 7 x 11
  grade nLoans defaults avgInterest stdInterest avgLoanAMt avgPmnt avgRet stdRet minRet maxRet
  <chr>  <int>    <int>       <dbl>       <dbl>      <dbl>   <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
1 A      24854     1369        7.21       0.973     14349.  15388.   2.35   4.16  -32.3   5.80
2 B      37865     4264       10.9        1.48      12506.  13634.   2.95   6.21  -33.3  13.8
3 C      29145     5206       13.9        1.23      12048.  13094.   2.85   8.15  -33.3  10.5
4 D      13455     3165       17.3        1.22      11896.  12831.   2.80  10.0   -33.3  11.9
5 E       3790     1090       20.0        1.40      11924.  12624.   2.53  11.4   -33.3  14.0
6 F        753      252       23.9        0.955      9435.   9953.   2.95  12.9   -32.0  18.0
7 G         82       31       26.5        0.958     11550.  11836.   1.36  14.9   -28.6  17.0
```

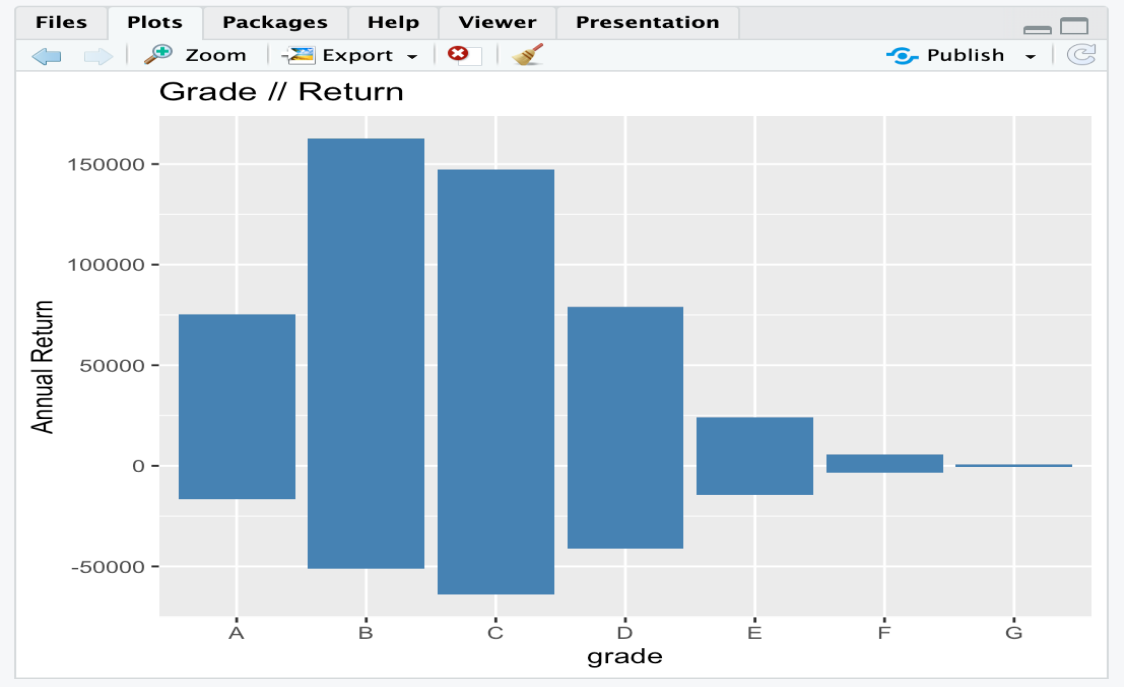
The above data shows that the max return percentage grade wise and the data is negative is for the charged off loans which is basically loss.

Is there any return from loans which are 'charged off'? Explain. How does return from charged -off loans vary by loan grade?

lcdf %>% filter(loan_status == "Charged Off") %>% group_by(grade) %>% summarise(nLoans=n(), avgInterest= mean(int_rate), avgLoanAmt=mean(loan_amnt), avgPmnt=mean(total_pymnt), avgRet=mean(annRet), minRet=min(annRet), maxRet=max(annRet))

```
# A tibble: 7 x 8
  grade nLoans avgInterest avgLoanAmt avgPmnt avgRet minRet maxRet
  <chr>  <int>       <dbl>      <dbl>   <dbl>  <dbl>  <dbl>  <dbl>
1 A       1369        7.49     13747.   8781.  -12.1  -32.3   5.80
2 B       4264       11.0      12195.   7939.  -11.7  -33.3  13.8
3 C       5206       14.0      12085.   7792.  -11.9  -33.3   9.54
4 D       3165       17.2      12383.   7724.  -12.6  -33.3  11.3
5 E       1090       20.0      12722.   7858.  -12.6  -33.3  11.7
6 F        252       23.9      10032.   5931.  -12.2  -32.0  14.4
7 G         31       26.4      12469.   7056.  -15.5  -28.6   4.86
```

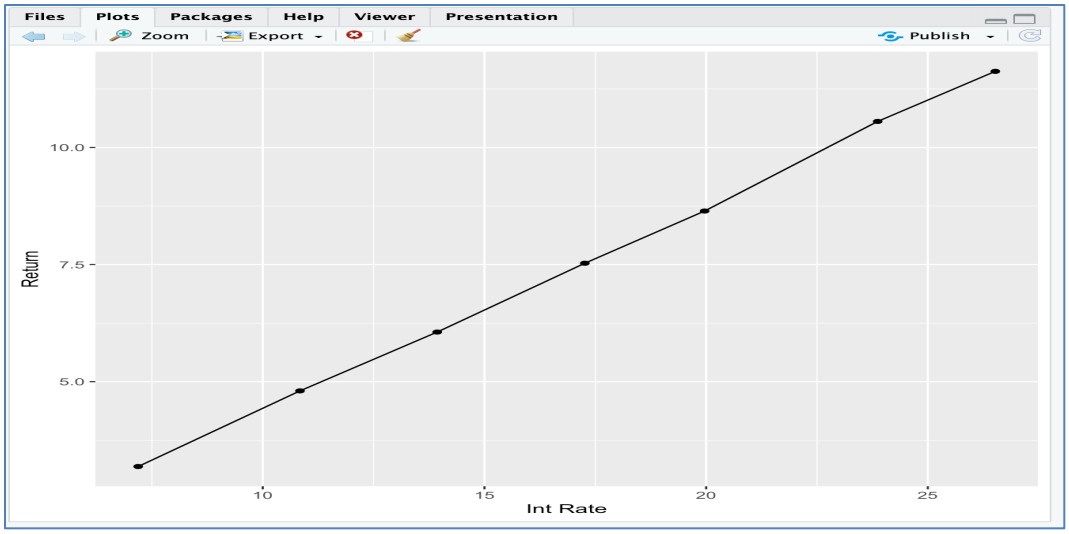
Graph to visualize the data above:



The data above and the graph shows the min,max and avg. return from the charged off loans which shows over all the grades from A to G. It is seen that the average return from the loans which are charged off is in negative which shows that they all are losses however some loans have return which is positive and which is a profit for the investor. The highest positive return is from grade B and lowest positive return is from grade G. The graph helps us understand the data and the analysis better and hence can be used to do better investment planning.

Compare the average return values with the average interest-rate on loans – do you notice any differences, and how do you explain this?

returns_garde<-lcdf %>% group_by(grade) %>% summarise(nLoans=n(), defaults=sum(loan_status=="Charged Off"), avgInterest= mean(int_rate), stdInterest=sd(int_rate), avgLoanAMt=mean(loan_amnt), avgPmnt=mean(total_pymnt), avgRet=mean(annRet), stdRet=sd(annRet), minRet=min(annRet), maxRet=max(annRet))

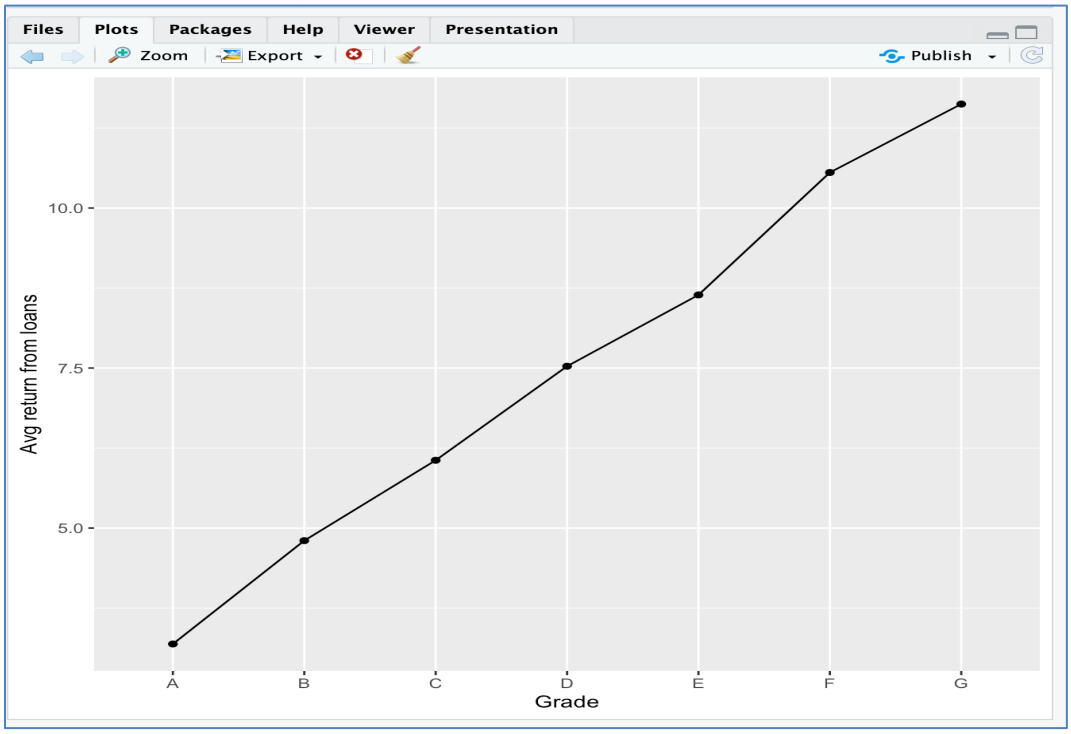


The line graph above shows that the average return is directly proportional to the interest rates of the loan and hence the higher the interest rate the higher is the avg return from the particular loan.

How do returns vary by grade, and by sub-grade.

Grade:

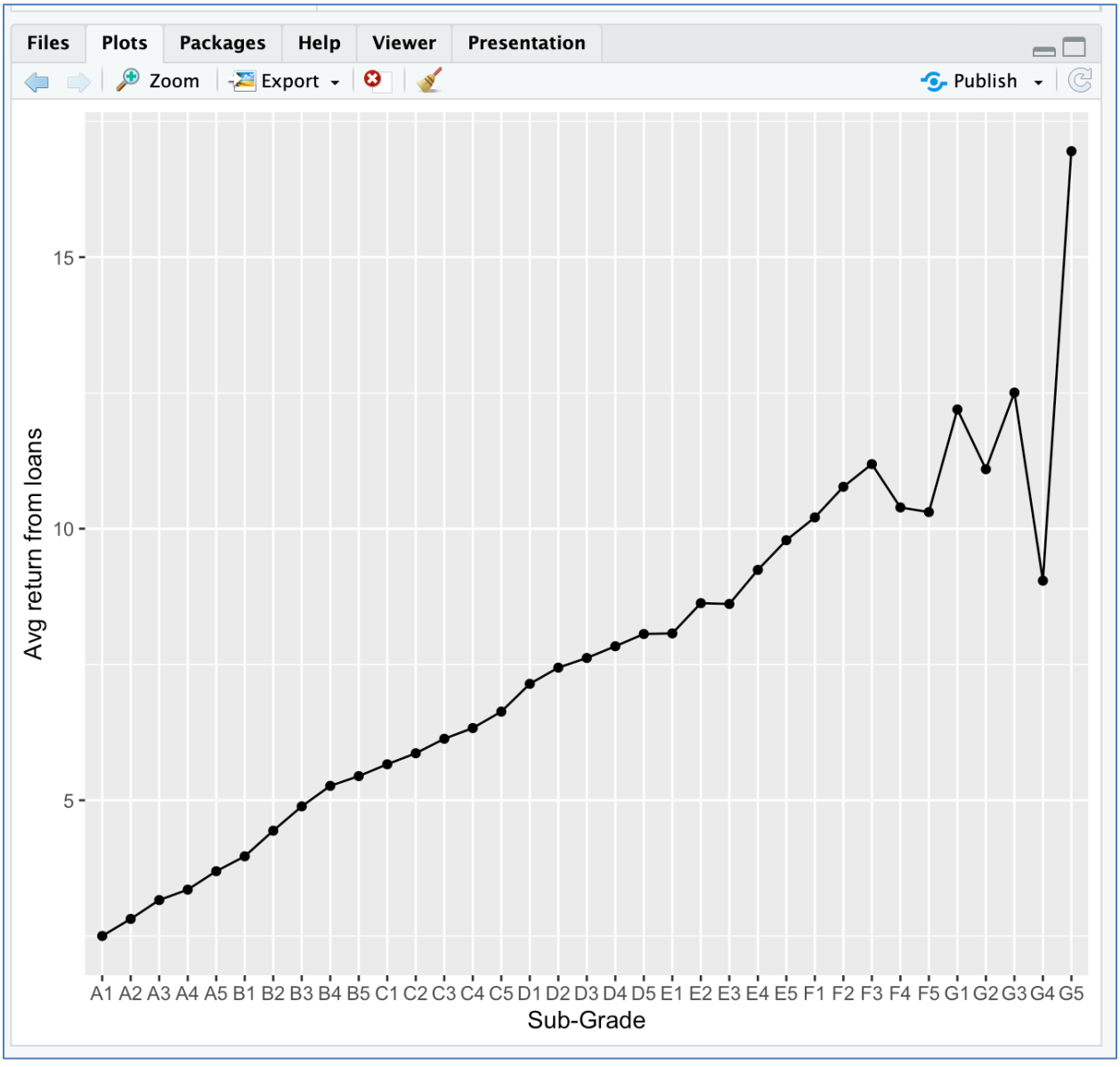
returns_grade<-lcdf %>% filter(loan_status == "Fully Paid") %>% group_by(grade) %>% summarise(nLoans=n(), avgInterest= mean(int_rate), avgLoanAmt=mean(loan_amnt), avgPmnt=mean(total_pymnt), avgRet=mean(annRet), minRet=min(annRet), maxRet=max(annRet))



The line graph shows that the avg return is higher with a lower grade (G) and least with the highest grade (A) the reason behind this is that the loans with Grade A have clients with less risk and better credit history and hence the interest rate of those loans is less which makes the avg return less and vice versa is for the loans with Grade G as they have more risk and bad credit history which makes the interest rate to be higher causing higher avg return.

Sub Grade:

returns_sub_grade<-lcdf %>% filter(loan_status == "Fully Paid") %>% group_by(sub_grade) %>% summarise(nLoans=n(), avgInterest= mean(int_rate), avgLoanAmt=mean(loan_amnt), avgPmnt=mean(total_pymnt), avgRet=mean(annRet), minRet=min(annRet), maxRet=max(annRet))



Similar trend is seen in subgrade as seen in the grade for the avg return; the sub grade line graph helps us analyze the returns in a better way.

If you decide to invest in loans based on this data exploration, which loans would you prefer to invest in?

Based on the willingness to take risk while investing in the money will be considered, for smaller amounts we would be willing to take higher risk and hence we would invest in loans with Grades G-E and its sub grades and with increase in the amount of investment we would reduce the risk and hence go with Grades A - D as the risk is lower and the return is also lower but the amount is high so the collective return in magnitude would be high only hence we would vary the grades of the loans we would Invest in based on the amount we invest.

2 a-(v) What are people borrowing money for (loan purpose)?

lcdf %>% group_by(purpose) %>% tally()

purpose_loan_status<-
lcdf%>%group_by(purpose,loan_status)%>%summarise(n=n(),mean_loan=mean(loan_amnt))%>%mutate(freq=n/sum(n)*100)

	purpose	n
	<fct>	<int>
1	car	1084
2	credit_card	27091
3	debt_consolidation	63291
4	home_improvement	6192
5	house	432
6	major_purchase	2111
7	medical	1170
8	moving	755
9	other	5948
10	small_business	1117
11	vacation	753

The data above shows the various purposes of the loans taken and also shows the number of such loans taken for each.

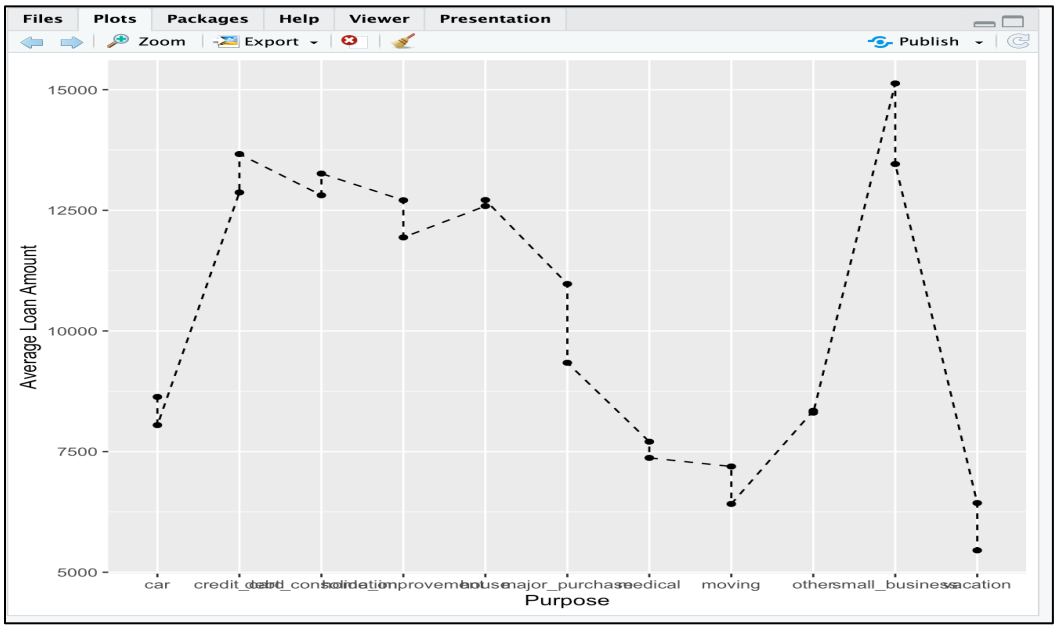
```
# A tibble: 22 x 5
# Groups:   purpose [11]
   purpose            loan_status     n mean_loan  freq
   <fct>              <chr>       <int>     <dbl> <dbl>
 1 car                Charged Off   125     8634.  11.5
 2 car                Fully Paid    959     8049.  88.5
 3 credit_card        Charged Off  3169    12870.  11.7
 4 credit_card        Fully Paid  23922    13666.  88.3
 5 debt_consolidation Charged Off  9254    12812.  14.6
 6 debt_consolidation Fully Paid  54037    13261.  85.4
 7 home_improvement   Charged Off   780    12709.  12.6
 8 home_improvement   Fully Paid   5412    11939.  87.4
 9 house              Charged Off    70    12588.  16.2
10 house              Fully Paid    362    12714.  83.8
# … with 12 more rows
```

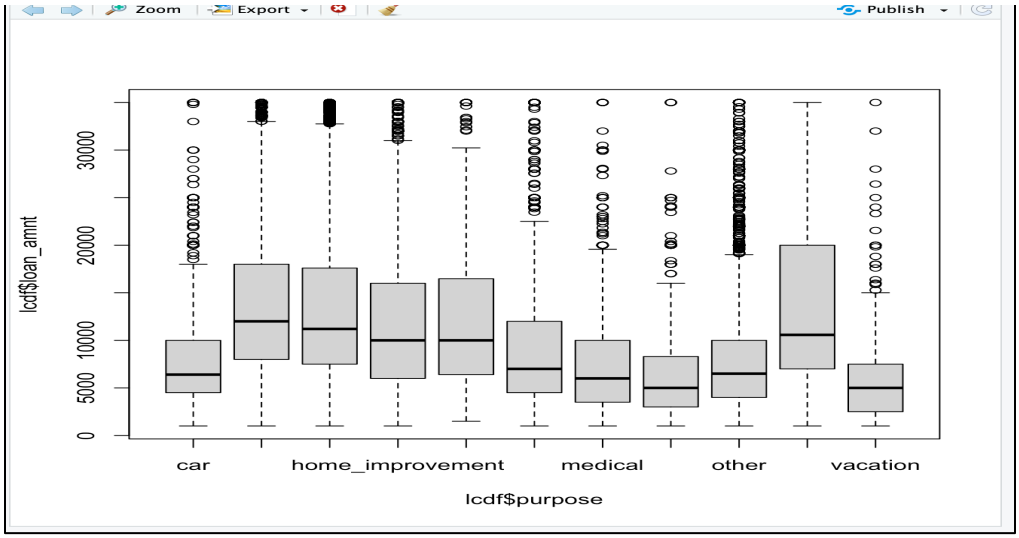
The above data shows the number of loans which are charged off for various purposes with house loans having the least percent of defaulters and credit card loans having the highest defaulters

Examine how many loans, average amounts, etc. by purpose? Do loan amounts vary by purpose?

Line:

ggplot(data=purpose_loan_status, aes(x=purpose, y=mean_loan, group=1)) + geom_line(linetype = "dashed")+geom_point()+labs(y="Average Loan Amount", x = "Purpose")





The above plot and graph shows that even though the highest number of loans taken are in the form of credit cards for debt consolidation, the loan amount is not highest for this and hence we get to see how the data when viewed in terms of magnitude of amount we see that “others” has highest. Based on the target of the investment they can invest in loans.

Does loan-grade assigned by Lending Club vary by purpose?

R 4.1.1 · ~/Desktop/DM A2/

	A	B	C	D	E	F	G
car	311	333	278	117	33	11	1
credit_card	8852	10795	5464	1686	267	24	3
debt_consolidation	12643	21993	18035	8130	2121	341	28
home_improvement	1602	2033	1584	699	226	43	5
house	51	99	114	92	49	23	4
major_purchase	571	622	555	272	77	11	3
medical	105	292	391	239	104	34	5
moving	39	104	249	218	101	42	2
other	522	1279	1907	1488	585	151	16
small_business	92	166	301	324	164	57	13
vacation	66	149	267	190	63	16	2

The data above shows the various numbers of loans and the respective grades of the loans.

Console | Terminal | Background Jobs

R 4.1.1 · ~/Desktop/DM A2/

	car	credit_card	debt_consolidation	home_improvement	house	major_purchase	medical
ANY	0	0	0	0	0	0	0
MORTGAGE	460	12239	29512	4684	121	868	497
NONE	0	7	2	1	0	0	0
OTHER	0	1	2	1	0	0	0
OWN	124	2800	6055	968	54	246	130
RENT	500	12044	27720	538	257	997	543

	moving	other	small_business	vacation
ANY	0	1	0	0
MORTGAGE	128	2273	499	263
NONE	0	0	0	0
OTHER	0	0	0	0
OWN	38	630	121	81
RENT	589	3044	497	409

The above data is a little detailed snippet of people with home improvement loans and checking if they own a house or they have rented an apartment. The clients and consider this information to estimate the risk of the loans as the risk of the loan reduces with them having a house.

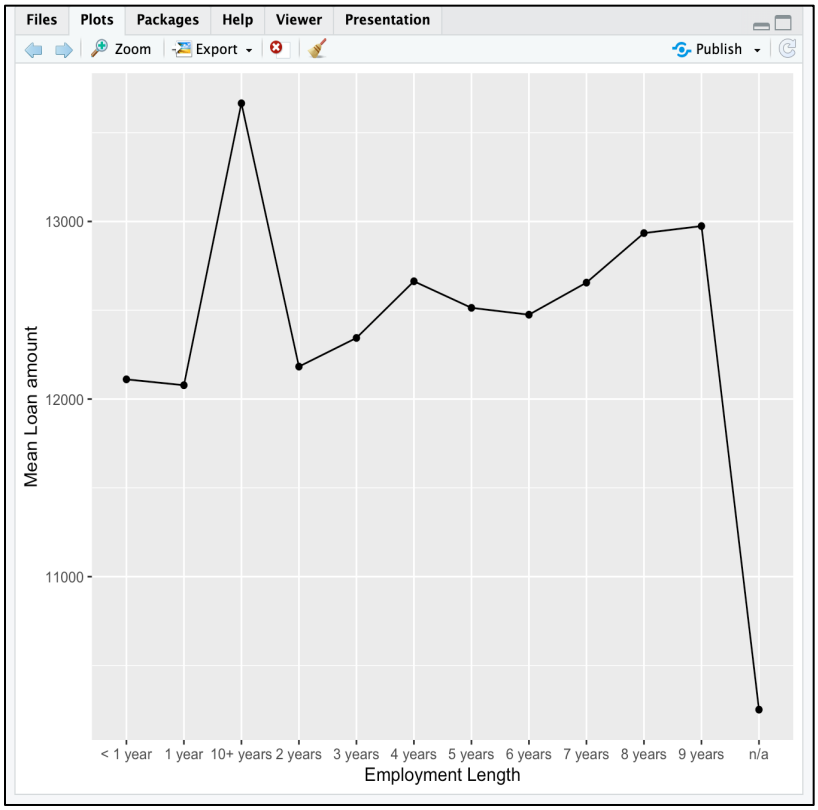
vi) Consider some borrower characteristics like employment-length, annual-income, fico-scores (low, high). How do these relate to loan attributes like, for example, loan_amout, loan_status, grade, purpose, actual return, etc.

Employment period versus Mean Loan amount:

emply_loanamt<-lcdf%>%group_by(emp_length)%>%summarise(n=n(),mean_loan=mean(loan_amnt))

```
# A tibble: 12 x 3
   emp_length      n mean_loan
   <chr>       <int>     <dbl>
 1 < 1 year     8784    12111.
 2 1 year       7336    12078.
 3 10+ years   34350    13665.
 4 2 years      9798    12183.
 5 3 years      8887    12344.
 6 4 years      6503    12663.
 7 5 years      6973    12514.
 8 6 years      5449    12475.
 9 7 years      5575    12656.
10 8 years      5456    12934.
11 9 years      4187    12974.
12 n/a          6646    10251.
```

ggplot(data=emply_loanamt, aes(x=emp_length, y=mean_loan, group=1)) +geom_line()+geom_point() + labs(y="Mean Loan amount", x = "Employment Length")



Employment length versus grade:

emply_grade<-lcdf%>%group_by(emp_length,grade)%>%summarise(n=n(),mean_loan=mean(loan_amnt))

```
# A tibble: 84 x 4
# Groups:   emp_length [12]
   emp_length grade     n mean_loan
   <chr>      <chr> <int>     <dbl>
 1 < 1 year   A      1984    14170.
 2 < 1 year   B      2943    12142.
 3 < 1 year   C      2334    11248.
 4 < 1 year   D      1109    10910.
 5 < 1 year   E       337    10700.
 6 < 1 year   F        70     7559.
 7 < 1 year   G         7     6900
 8 1 year     A      1555    13962.
 9 1 year     B      2445    12257.
10 1 year     C      2073    11042.
# … with 74 more rows
```

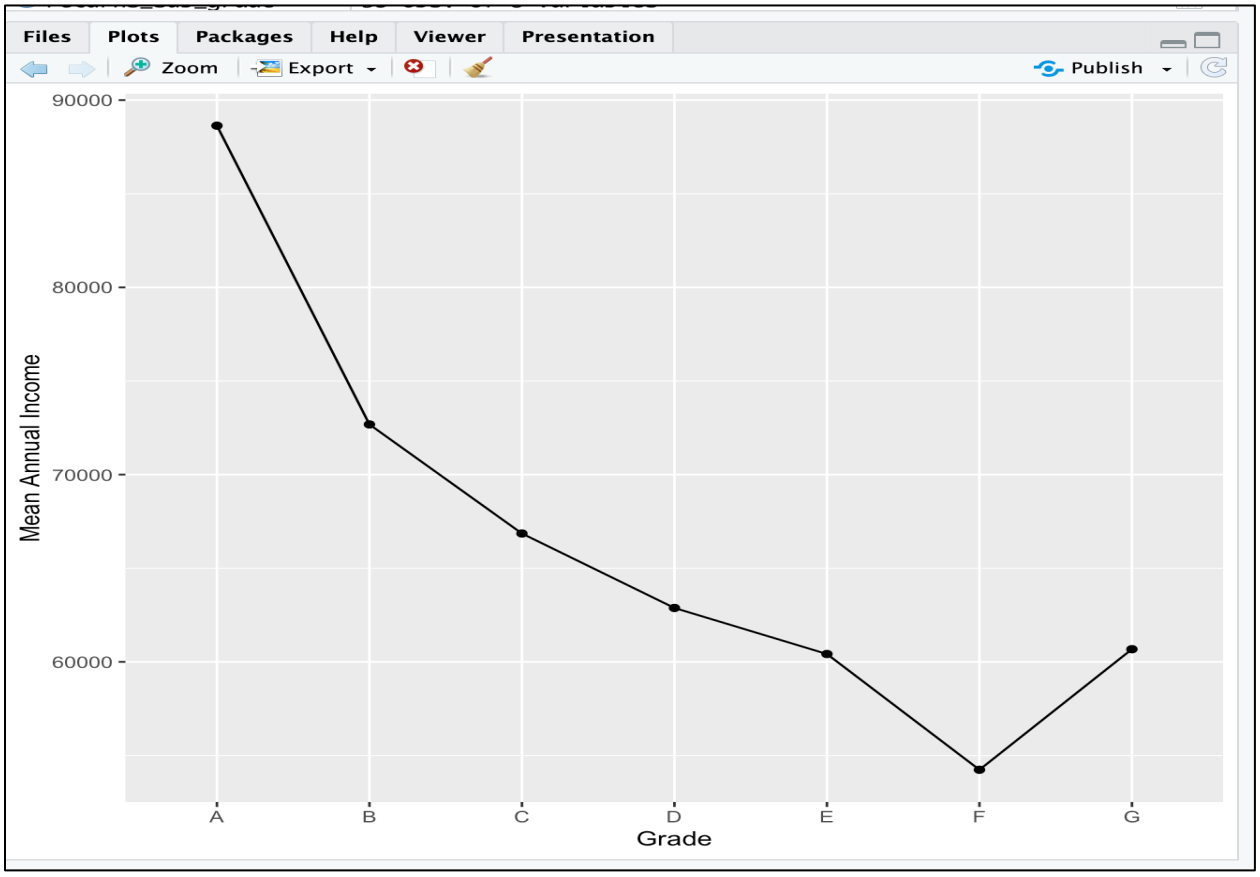
Employment length versus purpose

emply_purpose<-
lcdf%>%group_by(emp_length,purpose)%>%summarise(n=n(),mean_loan=mean(loan_amnt))

```
Console   Terminal ×   Background Jobs ×
R 4.1.1 · ~/Desktop/DM A2/
> print(emply_purpose)
# A tibble: 132 x 4
# Groups:   emp_length [12]
   emp_length purpose               n mean_loan
   <chr>      <fct>             <int>     <dbl>
 1 < 1 year   car                  90     8005.
 2 < 1 year   credit_card        2415    12862.
 3 < 1 year   debt_consolidation 4976    12502.
 4 < 1 year   home_improvement    330    12922.
 5 < 1 year   house                44    11629.
 6 < 1 year   major_purchase      162     9592.
 7 < 1 year   medical              79     7220.
 8 < 1 year   moving               122     6598.
 9 < 1 year   other                443     7591.
10 < 1 year   small_business       78    13894.
```

Annual income versus Grade:

annInc_grade<-lcdf%>%group_by(grade)%>%summarise(n=n(),mean_anninc=mean(annual_inc))



Defaults vary by emp_length

table(lcdf$loan_status, lcdf$emp_length)

```
             < 1 year 1 year 10+ years 2 years 3 years 4 years 5 years 6 years 7 years 8 years
  Charged Off     1269   1097      4380    1327    1265     895     983     757     737     724
  Fully Paid      7515   6239     29970    8471    7622    5608    5990    4692    4838    4732

             9 years   n/a
  Charged Off     598  1345
  Fully Paid     3589  5301
> 
```

(vii) Generate some (at least 3) new derived attributes which you think may be useful for predicting default., and explain what these are. For these, do an analysis as in the questions above (as reasonable based on the derived variables).

Derived attribute 1: proportion of satisfactory bankcard accounts

```
lcdf$propSatisBankcardAccts <- ifelse(lcdf$num_bc_tl>0, lcdf$num_bc_sats/lcdf$num_bc_tl, 0)
```

Derived attribute 2 - the length of borrower's history with LC

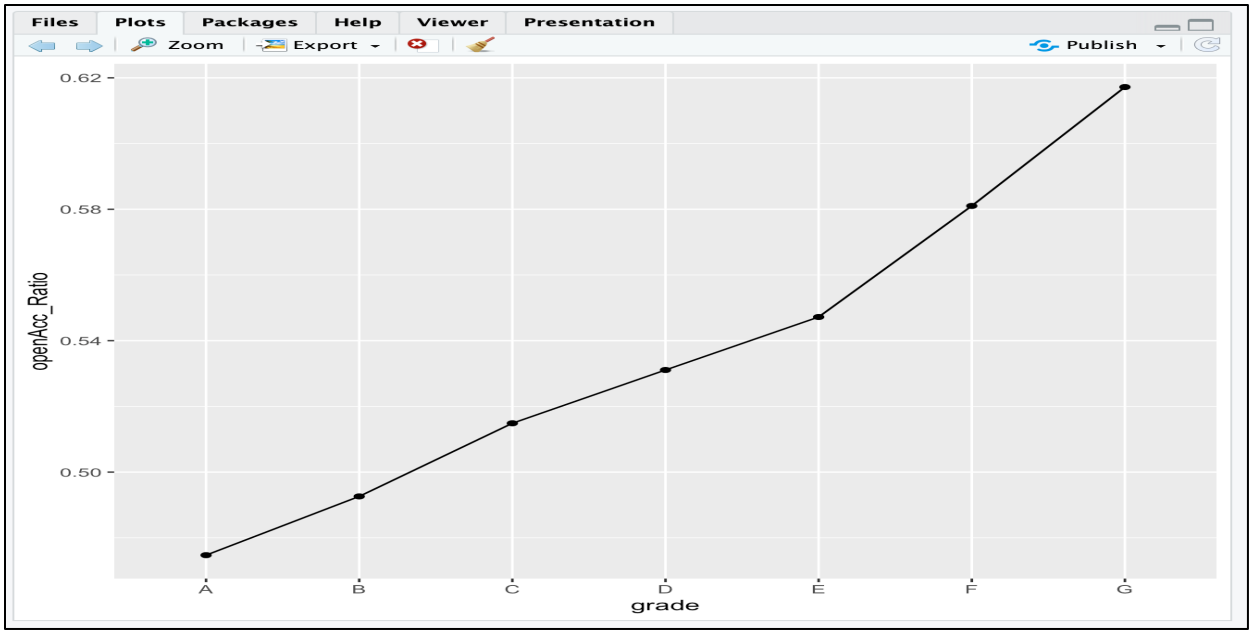
```
lcdf$earliest_cr_line<-paste(lcdf$earliest_cr_line, "-01", sep = "")

lcdf$earliest_cr_line<-parse_date_time(lcdf$earliest_cr_line, "myd")

lcdf$borrHistory <- as.duration(lcdf$earliest_cr_line %--% lcdf$issue_d) /dyears(1)
```

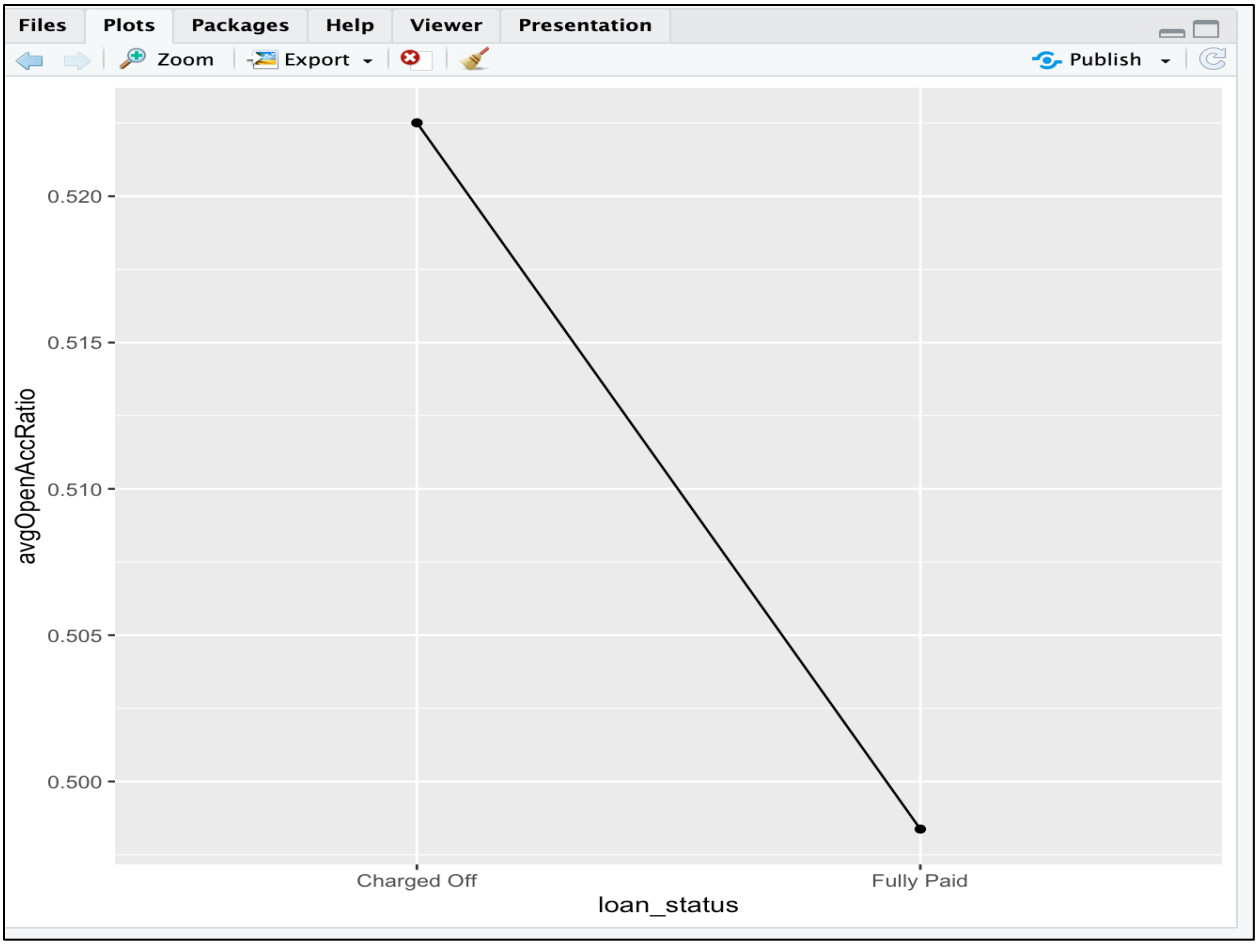
Derived Attribute 3 : ratio of openAccounts to totalAccounts

```
lcdf$openAccRatio <- ifelse(lcdf$total_acc>0, lcdf$open_acc/lcdf$total_acc, 0)

head(lcdf$openAccRatio)

openAcc_Grade <- lcdf %>% group_by(grade) %>% summarise(openAcc_Ratio=mean(openAccRatio))
```

	grade <chr>	openAcc_Ratio <dbl>
1	A	0.475
2	B	0.493
3	C	0.515
4	D	0.531
5	E	0.547
6	F	0.581
7	G	0.617

The data shows that loans with better grade are more likely to be fully paid soon and hence the open account ratio is less for them and vice versa same can be observed in the graphs below.


Files
Plots
Packages
Help
Viewer
Presentation
Zoom
Export
Publish
0.62
0.58
0.54
0.50
openAcc_Ratio
A
B
C
D
E
F
G
grade

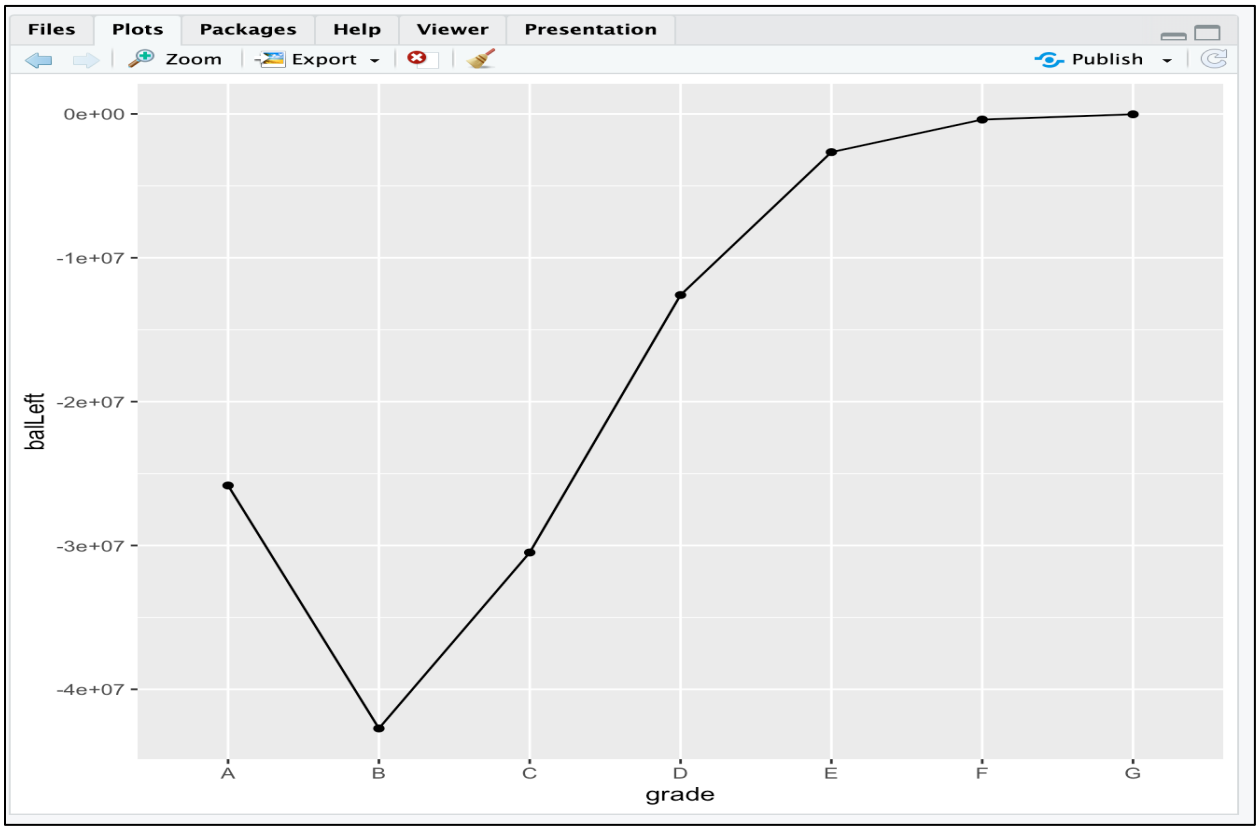

Files
Plots
Packages
Help
Viewer
Presentation
Zoom
Export
Publish
0.520
0.515
0.510
0.505
0.500
avgOpenAccRatio
Charged Off
Fully Paid
loan_status

Derived Attribute 4: Balance to pay - "Outstanding Amount"

lcdf$balance_to_pay <- lcdf$funded_amnt - lcdf$total_pymnt

For Grade:

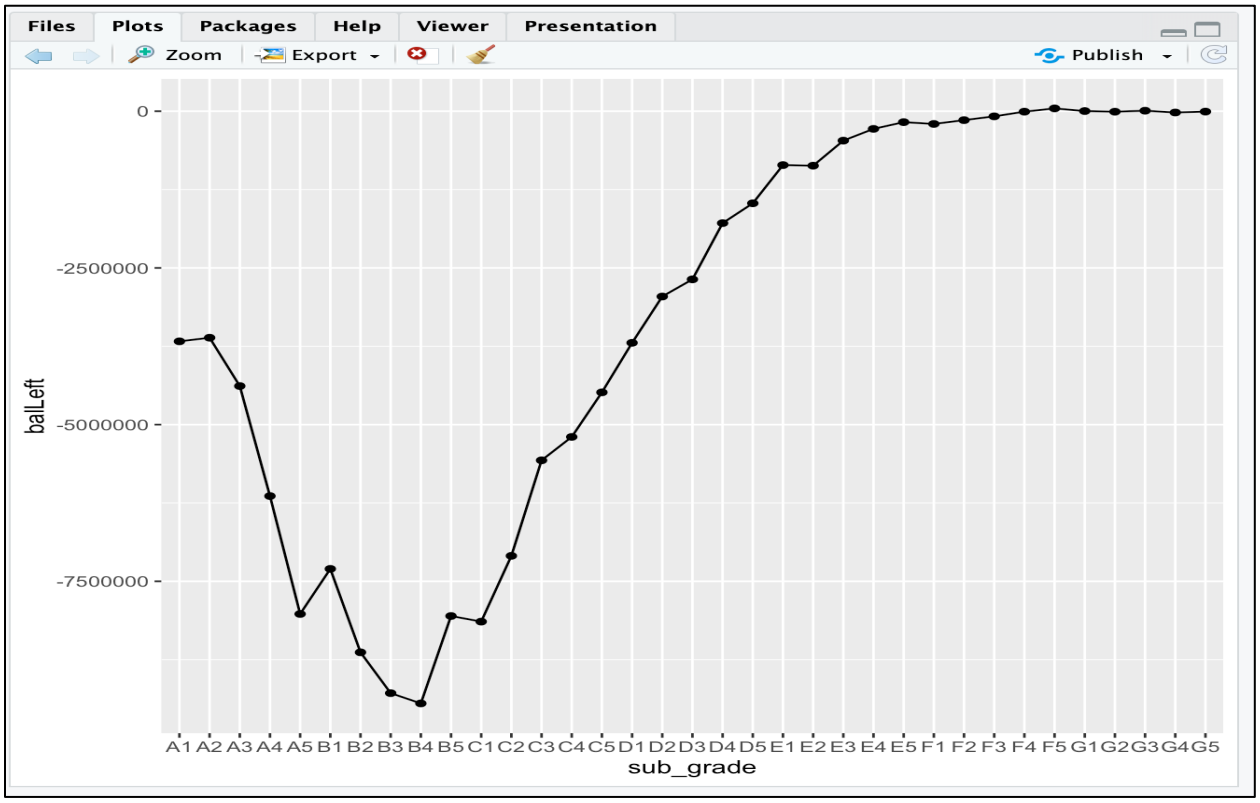
```
# A tibble: 7 x 2
  grade    balLeft
  <chr>      <dbl>
1 A      -25830381.
2 B      -42719725.
3 C      -30487722.
4 D      -12587631.
5 E       -2654009.
6 F        -390029.
7 G         -23392.
```



The outstanding amount for the loans with higher grades like G and F is higher than the loans with better grades like A,B,C and hence this shows that the loans are paid off properly if the grade is better.

For Sub Grade:

```
Console  Terminal  Background Jobs
R 4.1.1 · ~/Desktop/DM A2/
# A tibble: 35 x 2
   sub_grade   balLeft
   <chr>         <dbl>
 1 A1         -3671412.
 2 A2         -3613719.
 3 A3         -4384561.
 4 A4         -6139579.
 5 A5         -8021110.
 6 B1         -7301500.
 7 B2         -8632932.
 8 B3         -9284107.
 9 B4         -9447890.
10 B5         -8053297.
# … with 25 more rows
```



The same trend as grade is observed in sub grade the above data and graph shows a detailed view of each grade and subgrade.

2(b)Are there missing values in the data ? What is the proportion of missing values in different variables? Explain how you will handle missing values for different variables. You should consider what he variable is about, and what missing values may arise from – for example, a variable months Since Last Delinquency may have no value for someone who has not yet had a delinquency; what is a sensible value to replace the

missing values in this case? Are there some variables you will exclude from your model due to missing values?
Yes, there are a lot of variables that have complete missing values or values set as NA. These are the records that can be hidden as a reason to protect the privacy of an individual (Eg: id, member_id,etc.). These columns that are a threat to the privacy of an individual should be deleted as they are of no value to the analyst. They are just present to show that there exists some details about the person which is vulnerable and can't be displayed. Hence, it is pointless to keep those variables in the data set
or Partial Missing Values

There are also some variables that have some of their values missing. Eg: mths_since_last_major_derog, open account information, etc. Now these values are set as NA in many cases where the data should be numerical. To tackle this problem, we can replace these NA or NULL values with 0 so as to avoid any discrepancy and make sure that the data set is useful. The other solution to tackle this problem is to consider a sample of similar records for whose values are not 0 or NULL. But since this is a time- consuming process it's better to stick to the first method. If not, there can be errors that may arise in the process of deriving variables and that will make the results vulnerable. In conclusion, it completely depends upon the type of variable that the data frame contains. Every variable has its own significance. There can be some variable like id, address, name, phone number, etc that can be a threat to someone. Variables like these should be completely avoided. On the other hand, there are some variables without which the analysis can not be done. These variables s

#Drop col's with all empty values into new data frame -lcdf

```
lcdf <- LC_Data %>% select_if(function(x){!all(is.na(x))})
dim(lcdf)
```

```
[1] 110000    115
```

#columns where there are missing values

```
colMeans(is.na(lcdf))[colMeans(is.na(lcdf))>0]
```

emp_title	title	dti	mths_since_last_delinq
6.762727e-02	1.818182e-04	9.090909e-06	5.056182e-01
mths_since_last_record	revol_util	last_pymnt_d	next_pymnt_d
8.350364e-01	4.181818e-04	6.363636e-04	9.994909e-01
last_credit_pull_d	mths_since_last_major_derog	tot_coll_amt	tot_cur_bal
5.454545e-05	7.333455e-01	3.721818e-02	3.721818e-02
open_acc_6m	open_act_il	open_il_12m	open_il_24m
9.743727e-01	9.743727e-01	9.743727e-01	9.743727e-01
mths_since_rcnt_il	total_bal_il	il_util	open_rv_12m
9.751636e-01	9.743727e-01	9.779818e-01	9.743727e-01
open_rv_24m	max_bal_bc	all_util	total_rev_hi_lim
9.743727e-01	9.743727e-01	9.743727e-01	3.721818e-02
inq_fi	total_cu_tl	inq_last_12m	acc_open_past_24mths
9.743727e-01	9.743727e-01	9.743727e-01	9.872727e-03
avg_cur_bal	bc_open_to_buy	bc_util	mo_sin_old_il_acct
3.724545e-02	1.951818e-02	2.014545e-02	7.353636e-02
mo_sin_old_rev_tl_op	mo_sin_rcnt_rev_tl_op	mo_sin_rcnt_tl	mort_acc
3.721818e-02	3.721818e-02	3.721818e-02	9.872727e-03

```
lcdf <- lcdf %>% select(-names(lcdf)[colMeans(is.na(lcdf))>0.6])
dim(lcdf)
[1] 110000     94
```

#Check where the missing values are present

```
names(colMeans(is.na(lcdf))[colMeans(is.na(lcdf))>0])
[1] "emp_title"                "title"                    "dti"                      "mths_since_last_delinq"
 [5] "revol_util"              "last_pymnt_d"             "last_credit_pull_d"       "tot_coll_amt"
 [9] "tot_cur_bal"             "total_rev_hi_lim"         "acc_open_past_24mths"     "avg_cur_bal"
[13] "bc_open_to_buy"          "bc_util"                  "mo_sin_old_il_acct"       "mo_sin_old_rev_tl_op"
[17] "mo_sin_rcnt_rev_tl_op"   "mo_sin_rcnt_tl"           "mort_acc"                 "mths_since_recent_bc"
[21] "mths_since_recent_inq"   "num_accts_ever_120_pd"   "num_actv_bc_tl"           "num_actv_rev_tl"
[25] "num_bc_sats"             "num_bc_tl"                "num_il_tl"                "num_op_rev_tl"
[29] "num_rev_accts"           "num_rev_tl_bal_gt_0"      "num_sats"                 "num_tl_120dpd_2m"
[33] "num_tl_30dpd"            "num_tl_90g_dpd_24m"       "num_tl_op_past_12m"       "pct_tl_nvr_dlq"
[37] "percent_bc_gt_75"        "tot_hi_cred_lim"          "total_bal_ex_mort"        "total_bc_limit"
[41] "total_il_high_credit_limit" "satisBankcardAccts_prop"

lcdf<- lcdf %>% replace_na(list(bc_open_to_buy=median(lcdf$bc_open_to_buy, na.rm=TRUE), num_tl_120dpd_2m =
median(lcdf$num_tl_120dpd_2m, na.rm=TRUE),percent_bc_gt_75 = median(lcdf$percent_bc_gt_75, na.rm=TRUE),
bc_util=median(lcdf$bc_util, na.rm=TRUE) ))
names(colMeans(is.na(lcdf))[colMeans(is.na(lcdf))>0])

[1] "emp_title"                "title"                    "dti"                      "mths_since_last_delinq"
 [5] "revol_util"              "last_pymnt_d"             "last_credit_pull_d"       "tot_coll_amt"
 [9] "tot_cur_bal"             "total_rev_hi_lim"         "acc_open_past_24mths"     "avg_cur_bal"
[13] "mo_sin_old_il_acct"      "mo_sin_old_rev_tl_op"     "mo_sin_rcnt_rev_tl_op"    "mo_sin_rcnt_tl"
[17] "mort_acc"                "mths_since_recent_bc"     "mths_since_recent_inq"    "num_accts_ever_120_pd"
[21] "num_actv_bc_tl"          "num_actv_rev_tl"          "num_bc_sats"              "num_bc_tl"
[25] "num_il_tl"               "num_op_rev_tl"            "num_rev_accts"            "num_rev_tl_bal_gt_0"
[29] "num_sats"                "num_tl_30dpd"             "num_tl_90g_dpd_24m"       "num_tl_op_past_12m"
[33] "pct_tl_nvr_dlq"          "tot_hi_cred_lim"          "total_bal_ex_mort"        "total_bc_limit"
[37] "total_il_high_credit_limit" "satisBankcardAccts_prop"

dim(lcdf)
[1] 110000     94
```

QUESTION 2(c) Consider the potential for data leakage. You do not want to include variables in your model which may not be available when applying the model; that is, some data may not be available for new loans before they are funded. Leakage may also arise from variables in the data which may have been updated during the loan period (ie., after the loan is funded). Identify and explain which variables you will exclude in developing predictive models.

The above-mentioned number of variables represent data not available at the time the loan was issued. In all the models constructed thus far, we have carefully removed these attributes. we reintroduce them. Predictively, utilizing these additional attributes improves the model's performance. Using those attributes, we obtain a false positive rate of over 0.99. The total payments made on the loan will insignificantly be associated with all four of the main attributes (Funded Amount, loan default, loan early repayment, and time of aforementioned events). Indeed,if a loan defaults or is paid back early, total payments on the loan are likely to be lower. Thus, utilizing that variable for prediction will most likely result in a good model performance. However, when referring and investigating future loans, the total amount would eventually be repaid on that loan. and thus, a model trained using that variable could not be used to make future predictions

 # new data after considering for leakage

```
new_data <- LC_Data%>% select(-c(funded_amnt_inv, term, emp_title, pymnt_plan, title, zip_code, addr_state,
out_prncp, out_prncp_inv, total_pymnt_inv, total_rec_prncp, total_rec_int,total_rec_late_fee,recoveries,
```

collection_recovery_fee, last_credit_pull_d,policy_code, disbursement_method, debt_settlement_flag, hardship_flag, application_type))

#removing additional variables which are not present in the

```
new_data <- new_data %>% select(-c(last_pymnt_d, last_pymnt_amnt))
dim(new_data)

[1] 110000    130

lcdf$return = lcdf$total_pymnt-lcdf$funded_amnt
lcdf$returnperyear = (lcdf$return/lcdf$funded_amnt)/3*100

names(colMeans(is.na(new_data))[colMeans(is.na(new_data))>0])
new_data <- new_data %>% select(-c(return,returnperyear))
[1] "id"                                "member_id"                         "url"
 [4] "desc"                             "dti"                               "earliest_cr_line"
 [7] "mths_since_last_delinq"           "mths_since_last_record"            "revol_util"
[10] "next_pymnt_d"                     "mths_since_last_major_derog"       "annual_inc_joint"
[13] "dti_joint"                        "verification_status_joint"         "tot_coll_amt"
[16] "tot_cur_bal"                      "open_acc_6m"                       "open_act_il"
[19] "open_il_12m"                      "open_il_24m"                       "mths_since_rcnt_il"
[22] "total_bal_il"                     "il_util"                           "open_rv_12m"
[25] "open_rv_24m"                      "max_bal_bc"                        "all_util"
[28] "total_rev_hi_lim"                 "inq_fi"                            "total_cu_tl"
[31] "inq_last_12m"                     "acc_open_past_24mths"              "avg_cur_bal"
[34] "bc_open_to_buy"                   "bc_util"                           "mo_sin_old_il_acct"
[37] "mo_sin_old_rev_tl_op"             "mo_sin_rcnt_rev_tl_op"             "mo_sin_rcnt_tl"
[40] "mort_acc"                         "mths_since_recent_bc"              "mths_since_recent_bc_dlq"
[43] "mths_since_recent_inq"            "mths_since_recent_revol_delinq"    "num_accts_ever_120_pd"
[46] "num_actv_bc_tl"                   "num_actv_rev_tl"                   "num_bc_sats"
[49] "num_bc_tl"                        "num_il_tl"                         "num_op_rev_tl"
[52] "num_rev_accts"                    "num_rev_tl_bal_gt_0"               "num_sats"
[55] "num_tl_120dpd_2m"                 "num_tl_30dpd"                      "num_tl_90g_dpd_24m"
[58] "num_tl_op_past_12m"               "pct_tl_nvr_dlq"                    "percent_bc_gt_75"
[61] "tot_hi_cred_lim"                  "total_bal_ex_mort"                 "total_bc_limit"
[64] "total_il_high_credit_limit"       "revol_bal_joint"                   "sec_app_earliest_cr_line"
[67] "sec_app_inq_last_6mths"           "sec_app_mort_acc"                  "sec_app_open_acc"
[70] "sec_app_revol_util"               "sec_app_open_act_il"               "sec_app_num_rev_accts"
[73]  "sec_app_chargeoff_within_12_mths"                                    "sec_app_collections_12_mths_ex_med"
"sec_app_mths_since_last_major_derog"
[76] "hardship_type"                    "hardship_reason"                   "hardship_status"
[79] "deferral_term"                    "hardship_amount"                   "hardship_start_date"
[82] "hardship_end_date"                "payment_plan_start_date"           "hardship_length"
[85] "hardship_dpd"                     "hardship_loan_status"              "orig_projected_additional_accrued_interest"
[88] "hardship_payoff_balance_amount"   "hardship_last_payment_amount"      "debt_settlement_flag_date"
[91] "settlement_status"                "settlement_date"                   "settlement_amount"
[94] "settlement_percentage"            "settlement_term"                   "satisBankcardAccts_prop"
[97] "borrHistory"
```

Question 3: Do a univariate analysis to determine which variables (from amongst those you decide to consider for the next stage prediction task) will be individually useful for predicting the dependent variable (loan_status). For this, you need a measure of relationship between the dependent variable and each of the potential predictor variables. Given loan-status as a binary dependent variable, which measure will you use? From your analyses using this measure, which variables do you think will be useful for predicting loan_status? (Note – if certain variables on their own are highly predictive of the outcome, it is good to ask if this variable has a leakage issue).

We can compute the measure of relationship between each possible predictor variable and loan status in order to identify which variables are helpful for predicting the dependent variable (loan status) using univariate analysis. Since loan status is a binary dependent variable, we can quantify the association between the two variables using the point-biserial correlation coefficient or the phi coefficient.

We would need to create a binary classification model for each variable in the dataset in order to compute AUC for all of the variables in the dataset.

names <chr>	x <dbl>
loan_status	1.0000000
actualReturn	0.9870803
annRet	0.9679782
total_pymnt	0.7520816
actualTerm	0.6785136
sub_grade	0.6626403
int_rate	0.6557347
grade	0.6517691
inq_last_12m	0.5992411
open_rv_24m	0.5844758

1-10 of 86 rows Previous 1 2 3 4 5 6

```
> aucAll[aucAll>0.5]
                  loan_amnt                   funded_amnt
                  0.5150593                     0.5150668
                   int_rate                    installment
                  0.6557347                     0.5012055
                      grade                     sub_grade
                  0.6517691                     0.6626403
                 emp_length                home_ownership
                  0.5285924                     0.5520824
                 annual_inc                   loan_status
                  0.5751039                     1.0000000
                        dti                inq_last_6mths
                  0.5619944                     0.5493768
      mths_since_last_delinq       mths_since_last_record
                  0.5064687                     0.5100464
                   open_acc                     revol_bal
                  0.5047497                     0.5375396
                 revol_util                     total_acc
                  0.5310350                     0.5227152
         initial_list_status                   total_pymnt
                  0.5167777                     0.7520816
```

```
       initial_list_status                    total_pymnt
                 0.5167777                      0.7520816
              tot_cur_bal                     open_il_12m
                 0.5646110                      0.5667147
             open_il_24m                mths_since_rcnt_il
                 0.5784055                      0.5811666
           total_bal_il                         il_util
                 0.5262923                      0.5294104
             open_rv_24m                     max_bal_bc
                 0.5844758                      0.5610253
                all_util                total_rev_hi_lim
                 0.5516441                      0.5656761
                  inq_fi                    inq_last_12m
                 0.5585263                      0.5992411
     acc_open_past_24mths                    avg_cur_bal
                 0.5791133                      0.5717757
          bc_open_to_buy                       bc_util
                 0.5716613                      0.5435288
      mo_sin_old_il_acct                mo_sin_old_rev_tl_op
                 0.5358838                      0.5530406
```

```
            mo_sin_old_il_acct                mo_sin_old_rev_tl_op
                     0.5358838                         0.5530406
          mo_sin_rcnt_rev_tl_op                      mo_sin_rcnt_tl
                     0.5529760                         0.5589976
                      mort_acc                mths_since_recent_bc
                     0.5576951                         0.5569380
      mths_since_recent_bc_dlq                mths_since_recent_inq
                     0.5004078                         0.5451220
mths_since_recent_revol_delinq                         num_bc_tl
                     0.5036378                         0.5143056
                        num_il_tl                      num_op_rev_tl
                     0.5146568                         0.5179599
                    num_rev_accts                         num_sats
                     0.5080241                         0.5052517
                num_tl_120dpd_2m                    pct_tl_nvr_dlq
                     0.5000823                         0.5139742
                 tot_hi_cred_lim                total_bal_ex_mort
                     0.5764183                         0.5235713
                  total_bc_limit          total_il_high_credit_limit
                     0.5724659                         0.5176916
                          annRet                        actualTerm
                     0.9679782                         0.6785136
```

```
                    actualReturn              propSatisBankcardAccts
                       0.9870803                           0.5225518
                      borrHistory
                       0.5428471
>
```

Leakage occurs when a variable that should not be available at the time of prediction is included as a predictor in the model. This can result in overly optimistic predictions and poor generalization to new data. In this case, **variables such as "total_pymnt" and "actualReturn"** are likely to create leakage as **they are highly correlated with the target variable "loan_status" and are likely to be known at the time of prediction.**

Including variables such as "total_pymnt" and "actualReturn" as predictors can result in a model that appears to perform well on the training data but fails to generalize to new data. **Therefore, it is important to avoid using such variables as predictors to prevent leakage and ensure the accuracy and generalizability of the model.**

4.We will next develop predictive models for loan_status. 4. (a) Split the data into training and validation sets. What proportions do you consider, why? (b) How will you evaluate performance – which measure do you consider, and why? For evaluation of models, you should include confusion matrix related measures, as well as ROC analyses and lifts. Clearly explain which performance measures you focus on, and why.

The reason for splitting the data into training and validation sets is to evaluate the performance of the model on data that it has not seen before, and to prevent overfitting. The training set is used to fit the model, while the validation set is used to assess the model's performance on new data.

#DT models using rpart
new_dt=new_data
glimpse(new_data)

```
> glimpse(new_data)
Rows: 110,000
Columns: 128
$ id                    <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,…
$ member_id             <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,…
$ loan_amnt             <dbl> 4400, 5850, 5000, 1600, 16000, 3000, 9000, 12000, 7000, 18000, 25000, 10000, 13000, 8000, 6500,…
$ funded_amnt           <dbl> 4400, 5850, 5000, 1600, 16000, 3000, 9000, 12000, 7000, 18000, 25000, 10000, 13000, 8000, 6500,…
$ int_rate              <dbl> 22.99, 21.99, 6.24, 14.99, 9.17, 8.18, 10.99, 8.90, 7.26, 6.99, 12.99, 16.29, 7.26, 9.17, 13.35…
$ installment           <dbl> 170.30, 223.39, 152.66, 55.46, 510.07, 94.26, 294.61, 381.04, 216.98, 555.71, 842.23, 353.01, 4…
$ grade                 <chr> "F", "E", "A", "C", "B", "B", "B", "A", "A", "A", "B", "C", "A", "B", "C", "D", "B", "C", "A", …
$ sub_grade             <chr> "F1", "E4", "A2", "C5", "B2", "B1", "B4", "A5", "A4", "A3", "B5", "C4", "A4", "B2", "C2", "D5",…
$ emp_length            <chr> "3 years", "< 1 year", "6 years", "10+ years", "10+ years", "< 1 year", "3 years", "10+ years",…
$ home_ownership        <chr> "RENT", "MORTGAGE", "MORTGAGE", "MORTGAGE", "MORTGAGE", "MORTGAGE", "RENT", "RENT", "MORTGAGE",…
$ annual_inc            <dbl> 22000.00, 19500.00, 48824.00, 50000.00, 114000.00, 53000.00, 25000.00, 92000.00, 60000.00, 7202…
$ verification_status   <chr> "Source Verified", "Verified", "Source Verified", "Source Verified", "Not Verified", "Not Verif…
$ issue_d               <dttm> 2015-03-01, 2014-05-01, 2015-09-01, 2014-05-01, 2015-05-01, 2015-11-01, 2015-02-01, 2013-11-01…
$ loan_status           <chr> "Fully Paid", "Fully Paid", "Fully Paid", "Fully Paid", "Fully Paid", "Fully Paid", "Fully Paid…
$ url                   <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,…
$ desc                  <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,…
```

varsOmit <- c('actualTerm', 'actualReturn', 'annRet', 'total_pymnt') #are there others?
library(rpart)
#Check of the target, loan_status, is a factor variable -- if not, convert to a factor variable
#lcdf$loan_status <- factor(lcdf$loan_status, levels=c("Fully Paid", "Charged Off"))
new_data2=new_data
new_data1 <- new_data%>%select(-c(annRet,total_pymnt, balance_to_pay))
new_data1 <- new_data1%>%select(-c(grade))
new_data1 <- new_data1%>%select(-c(actualTerm,funded_amnt))
names(colMeans(is.na(new_data1)))[colMeans(is.na(new_data1))>0]

```
 [1] "id"                          "member_id"                   "url"
 [4] "desc"                        "dti"                         "earliest_cr_line"
 [7] "mths_since_last_delinq"      "mths_since_last_record"      "revol_util"
[10] "next_pymnt_d"                "mths_since_last_major_derog" "annual_inc_joint"
[13] "dti_joint"                   "verification_status_joint"   "tot_coll_amt"
[16] "tot_cur_bal"                 "open_acc_6m"                 "open_act_il"
[19] "open_il_12m"                 "open_il_24m"                 "mths_since_rcnt_il"
[22] "total_bal_il"                "il_util"                     "open_rv_12m"
[25] "open_rv_24m"                 "max_bal_bc"                  "all_util"
[28] "total_rev_hi_lim"            "inq_fi"                      "total_cu_tl"
[31] "inq_last_12m"                "acc_open_past_24mths"        "avg_cur_bal"
[34] "bc_open_to_buy"              "bc_util"                     "mo_sin_old_il_acct"
[37] "mo_sin_old_rev_tl_op"        "mo_sin_rcnt_rev_tl_op"       "mo_sin_rcnt_tl"
[40] "mort_acc"                    "mths_since_recent_bc"        "mths_since_recent_bc_dlq"
[43] "mths_since_recent_inq"       "mths_since_recent_revol_delinq" "num_accts_ever_120_pd"
[46] "num_actv_bc_tl"              "num_actv_rev_tl"             "num_bc_sats"
[49] "num_bc_tl"                   "num_il_tl"                   "num_op_rev_tl"
[52] "num_rev_accts"               "num_rev_tl_bal_gt_0"         "num_sats"
[55] "num_tl_120dpd_2m"            "num_tl_30dpd"                "num_tl_90g_dpd_24m"
```

#replacing some of the missing NA values in the columns by median values

new_data1<-new_data1 %>% replace_na(list(mths_since_last_delinq=median(new_data1$mths_since_last_delinq, na.rm=TRUE),revol_util = median(new_data1$revol_util, na.rm=TRUE), avg_cur_bal = median(new_data1$avg_cur_bal, na.rm=TRUE), mths_since_recent_bc = median(new_data1$mths_since_recent_bc, na.rm=TRUE),mths_since_recent_inq = median(new_data1$mths_since_recent_inq, na.rm=TRUE), num_rev_accts =

```
median(new_data1$num_rev_accts,
na.rm=TRUE),pct_tl_nvr_dlq=median(new_data1$pct_tl_nvr_dlq,na.rm=TRUE),mo_sin_old_il_acct=median(new_data1$
mo_sin_old_il_acct, na.rm=TRUE) ))

names(colMeans(is.na(new_data1)))[colMeans(is.na(new_data1))>0]
```

```
 [1] "id"                          "member_id"                   "url"
 [4] "desc"                        "dti"                         "earliest_cr_line"
 [7] "mths_since_last_record"      "next_pymnt_d"                "mths_since_last_major_derog"
[10] "annual_inc_joint"            "dti_joint"                   "verification_status_joint"
[13] "tot_coll_amt"                "tot_cur_bal"                 "open_acc_6m"
[16] "open_act_il"                 "open_il_12m"                 "open_il_24m"
[19] "mths_since_rcnt_il"          "total_bal_il"                "il_util"
[22] "open_rv_12m"                 "open_rv_24m"                 "max_bal_bc"
[25] "all_util"                    "total_rev_hi_lim"            "inq_fi"
[28] "total_cu_tl"                 "inq_last_12m"                "acc_open_past_24mths"
[31] "bc_open_to_buy"              "bc_util"                     "mo_sin_old_rev_tl_op"
[34] "mo_sin_rcnt_rev_tl_op"       "mo_sin_rcnt_tl"              "mort_acc"
[37] "mths_since_recent_bc_dlq"    "mths_since_recent_revol_delinq" "num_accts_ever_120_pd"
[40] "num_actv_bc_tl"              "num_actv_rev_tl"             "num_bc_sats"
[43] "num_bc_tl"                   "num_il_tl"                   "num_op_rev_tl"
[46] "num_rev_tl_bal_gt_0"         "num_sats"                    "num_tl_120dpd_2m"
[49] "num_tl_30dpd"                "num_tl_90g_dpd_24m"          "num_tl_op_past_12m"
[52] "percent_bc_gt_75"            "tot_hi_cred_lim"             "total_bal_ex_mort"
```

```
new_data1$loan_status <- factor(new_data1$loan_status)#, levels=c("Fully Paid", "Charged Off"))

dim(new_data1)
[1] 110000    122


library(rpart)
library(rpart.plot)
library(ranger)
```

#Splitting data into 70% training and 30% testing ratio.

```
#Splitting data into 70% training and 30%  testing ratio.


TRNPROP = 0.7  #proportion of examples in the training sample
nr<-nrow(new_data1)
trnIndex<- sample(1:nr, size = round(TRNPROP * nr), replace=FALSE)

new_data1Trn <- new_data1[trnIndex, ]
new_data1Tst <- new_data1[-trnIndex, ]

#ran a  random forest based using ranger, splitrule is gini
new_data1T1<- ranger(loan_status ~., data=new_data1Trn, classification = TRUE,
                     num.trees =200, importance='permutation', probability = TRUE)

sort(new_data1T1$variable.importance, decreasing = TRUE)
```

```
 [1] "id"                    "member_id"                 "loan_amnt"
 [4] "int_rate"              "installment"               "sub_grade"
 [7] "emp_length"            "home_ownership"            "annual_inc"
[10] "verification_status"   "issue_d"                   "loan_status"
[13] "url"                   "desc"                      "purpose"
[16] "dti"                   "delinq_2yrs"               "earliest_cr_line"
[19] "inq_last_6mths"        "mths_since_last_delinq"    "mths_since_last_record"
[22] "open_acc"              "pub_rec"                   "revol_bal"
[25] "revol_util"            "total_acc"                 "initial_list_status"
[28] "next_pymnt_d"          "collections_12_mths_ex_med" "mths_since_last_major_derog"
[31] "annual_inc_joint"      "dti_joint"                 "verification_status_joint"
[34] "acc_now_delinq"        "tot_coll_amt"              "tot_cur_bal"
[37] "open_acc_6m"           "open_act_il"               "open_il_12m"
[40] "open_il_24m"           "mths_since_rcnt_il"        "total_bal_il"
[43] "il_util"               "open_rv_12m"               "open_rv_24m"
[46] "max_bal_bc"            "all_util"                  "total_rev_hi_lim"
[49] "inq_fi"                "total_cu_tl"               "inq_last_12m"
[52] "acc_open_past_24mths"  "avg_cur_bal"               "bc_open_to_buy"
[55] "bc_util"               "chargeoff_within_12_mths"  "delinq_amnt"
```

#Do you want to use all the variables in the dataset as predictors?
In general, it is a good idea to use all significant variables as predictors in order to make sure that the model includes all significant variables that affect the outcome variable.
Overfitting, which can result in subpar performance on fresh data, can be caused by adding unnecessary or highly linked variables.

To avoid leakage, we should only include the variables that are available at the time of the loan approval decision. To avoid leakage, we should only include the variables that are available at the time of the loan approval decision. The variables which are available at the time of loan acceptance, the following factors should be taken into account as predictors based on the information provided:
Loan_amnt,int_rate,grade,sub_grade,emp_length,home_ownership,annual_inc,verification_status,purpose,dti ,delinq_2yrs,inq_last_6mths,,pub_rec,revol_util,total_acc,initial_list_status etc

#Are are some variable you want to exclude - due to leakage, or other reasons?
Variables such as total_pymnt, acc_now_delinq, tot_coll_amt, tot_cur_bal, and others that relate to post-approval or post-disbursement information should be excluded to avoid leakage.

 # What about variables like actualTerm, actualReturn, ... which you calculated?
These will be useful in performance assessment, but should not be used in building the model.

#Are there any data variables which you may not want to use in developing the model?
Based on the variables found above, the following variables can be avoided.

funded_amnt: Given how closely this variable resembles loan amnt, using both of them in the model could result in multicollinearity problems.

sub_grade: Due to its high level of precision, this variable might not be helpful in predicting the intended variable. We can substitute the more broad variable "score" instead.

acc_now_delinq: The fact that there are so few non-zero numbers for this variable may make it challenging for the model to identify patterns in the data. We might want to think about taking this element out of the model.

5. Develop a decision tree model to predict default. Train decision tree models (use either rpart or c50)
What parameters do you experiment with, and what performance do you obtain (on training and validation sets)? Clearly tabulate your results and briefly describe your findings.
[If something looks too good, it may be due to leakage – make sure you address this]
Identify the best tree model. Why do you consider it best? Describe this model – in terms of complexity (size).
Examine variable importance. How does this relate to your univariate analyses in Question
3 above?

```
lcDT1 <- rpart(loan_status ~., data=lcdfTrn %>% select(-all_of(varsOmit)),
method="class", parms = list(split = "information"), control =
rpart.control(minsplit = 30))
```

```
Classification tree:
rpart(formula = loan_status ~ ., data = lcdfTrn %>% select(-all_of(varsOmit)),
    method = "class", parms = list(split = "information"), control = rpart.control(minsplit
 = 30))

Variables actually used in tree construction:
character(0)

Root node error: 7617/54972 = 0.13856

n= 54972

  CP nsplit rel error xerror xstd
1  0      0         1      0    0
```

The output displays the results of using rpart function to train a classification tree. The formula : loan status is used to build the tree, and it indicates that the model is attempting to predict the loan status variable using all other factors in the dataset.

The output indicates that the root node error rate is 0.13856, which means that 13.9% of the observations in the training set are misclassified by the tree at the root node. From the output, we can see that the tree did not grow beyond the root node, as no variables were used in tree construction. This indicates that the model has to be improved, and we might have to change parameters like the minimum split size or use new splitting criteria to promote the growth of the tree.

The tree does not grow at all, so we set lower value of cp.

```
lcDT1 <- rpart(loan_status ~., data=lcdfTrn %>% select(-all_of(varsOmit)),
method="class", parms = list(split = "information"), control =
rpart.control(cp=0.0001, minsplit = 50))
```

The cp argument sets the complexity parameter to 0.0001, which controls the tree's growth, and the minsplit argument sets the minimum number of observations in any terminal node to 50.

```
Classification tree:
rpart(formula = loan_status ~ ., data = lcdfTrn %>% select(-all_of(varsOmit)),
    method = "class", parms = list(split = "information"), control = rpart.control(cp = 1e-04,
        minsplit = 50))

Variables actually used in tree construction:
 [1] acc_open_past_24mths           annual_inc
 [3] avg_cur_bal                    bc_open_to_buy
 [5] bc_util                        borrHistory
 [7] delinq_2yrs                    dti
 [9] emp_length                     funded_amnt
[11] grade                          home_ownership
[13] inq_last_6mths                 installment
[15] int_rate                       loan_amnt
[17] max_bal_bc                     mo_sin_old_il_acct
[19] mo_sin_old_rev_tl_op           mo_sin_rcnt_rev_tl_op
[21] mo_sin_rcnt_tl                 mort_acc
[23] mths_since_last_delinq         mths_since_last_major_derog
[25] mths_since_last_record         mths_since_recent_bc
[27] mths_since_recent_bc_dlq       mths_since_recent_inq
[29] mths_since_recent_revol_delinq num_actv_bc_tl
[31] num_actv_rev_tl                num_bc_sats
[33] num_bc_tl                      num_il_tl
[35] num_op_rev_tl                  num_rev_accts
[37] num_rev_tl_bal_gt_0            num_tl_op_past_12m
[39] open_acc                       pct_tl_nvr_dlq
[41] percent_bc_gt_75               propSatisBankcardAccts
[43] purpose                        revol_bal
[45] revol_util                     sub_grade
[47] tax_liens                      tot_cur_bal
[49] tot_hi_cred_lim                total_acc
[51] total_bal_ex_mort              total_bc_limit
[53] total_il_high_credit_limit     total_rev_hi_lim
[55] verification_status

Root node error: 7617/54972 = 0.13856

n= 54972
```

```
           CP nsplit rel error xerror     xstd
1  0.00035009      0   1.00000 1.0000 0.010635
2  0.00032821     23   0.98740 1.0158 0.010704
3  0.00029539     28   0.98530 1.0239 0.010740
4  0.00028883     34   0.98254 1.0253 0.010746
5  0.00028445     72   0.96665 1.0351 0.010789
6  0.00026257     80   0.96429 1.0356 0.010791
7  0.00023631    123   0.95011 1.0473 0.010842
8  0.00022975    128   0.94893 1.0554 0.010876
9  0.00021881    132   0.94801 1.0635 0.010911
10 0.00019693    146   0.94473 1.0721 0.010947
11 0.00017505    161   0.94132 1.0806 0.010983
12 0.00015754    174   0.93869 1.0880 0.011014
13 0.00014770    184   0.93711 1.0935 0.011037
14 0.00014441    192   0.93593 1.0935 0.011037
15 0.00013129    204   0.93409 1.1109 0.011108
16 0.00012034    232   0.93042 1.1200 0.011145
17 0.00011670    260   0.92622 1.1264 0.011171
18 0.00010940    269   0.92517 1.1292 0.011183
19 0.00010000    275   0.92451 1.1354 0.011207
>
```

The output indicates that a tree with 275 splits and a cross-validation error rate of 1.1354 was created using 55 variables and an optimal CP value of 0.0001. The results can be used to assess the tree's effectiveness and pinpoint possible areas for development.

Choosing the value with the lowest xerror—the cross-validation estimate of the prediction error—is typically one strategy for determining the "optimal" CP value.

In this instance, 0.00035009, which has a xerror of 1.0000, is the CP value with the lowest xerror.

Evaluating Performance:

```
> table(pred = predTrn, true=lcdfTrn$loan_status)
            true
pred         Charged Off Fully Paid
  Charged Off        7617          3
  Fully Paid            0      47352
> mean(predTrn == lcdfTrn$loan_status)
[1] 0.9999454
> table(pred = predict(lcDT1,lcdfTst, type='class'), true=lcdfTst$loan_status)
            true
pred         Charged Off Fully Paid
  Charged Off        7760          5
  Fully Paid            0      47207
> mean(predict(lcDT1,lcdfTst, type='class') ==lcdfTst$loan_status)
[1] 0.999909
>
```

First Block of code shows that the model correctly predicted all but 3 of the 47,370 Fully Paid loans, and correctly predicted all 7,620 Charged Off loans. The output indicates that the model achieved an accuracy of 0.9999454 on the training set.

The second block of code is similar to the first, but instead computes the performance of the model on a separate test set (lcdfTst). The confusion matrix shows that the model correctly predicted all but 5 of the 47,212 Fully Paid loans, and correctly predicted all 7,765 Charged Off loans. The mean function computes both the training and test sets, with near-perfect accuracy.

Now we will do the classification with a different classification threshold:
CTHRESH=0.3

```
> table(predTrnCT , true=lcdfTrn$loan_status)
            true
predTrnCT     Charged Off Fully Paid
  Charged Off        7617           3
  Fully Paid            0       47352
```

```
             actuals
predictions   Charged Off Fully Paid
  Fully Paid            0       47352
  Charged Off        7617           3
>
```

The matrix shows that the model correctly predicted 7617 Charged Off loans and 47352 Fully Paid loans. The overall accuracy of the model is very high, as indicated by the mean() function output of 0.9999454.

We must take into account evaluation measures like accuracy, precision, recall, F1-score, etc. in to choose the best tree model. The decision tree model lcDT1 looks to be doing exceptionally well, with very high accuracy and an F1-score of 0.9999 for both the training and testing datasets

Also, the lcDT1 decision tree has only 19 splits and a CP value of 0.0001, indicating that it is relatively simple. As a result, it may be said to be the best tree model because it maintains a low level of complexity while having a high accuracy score and good F1-score.

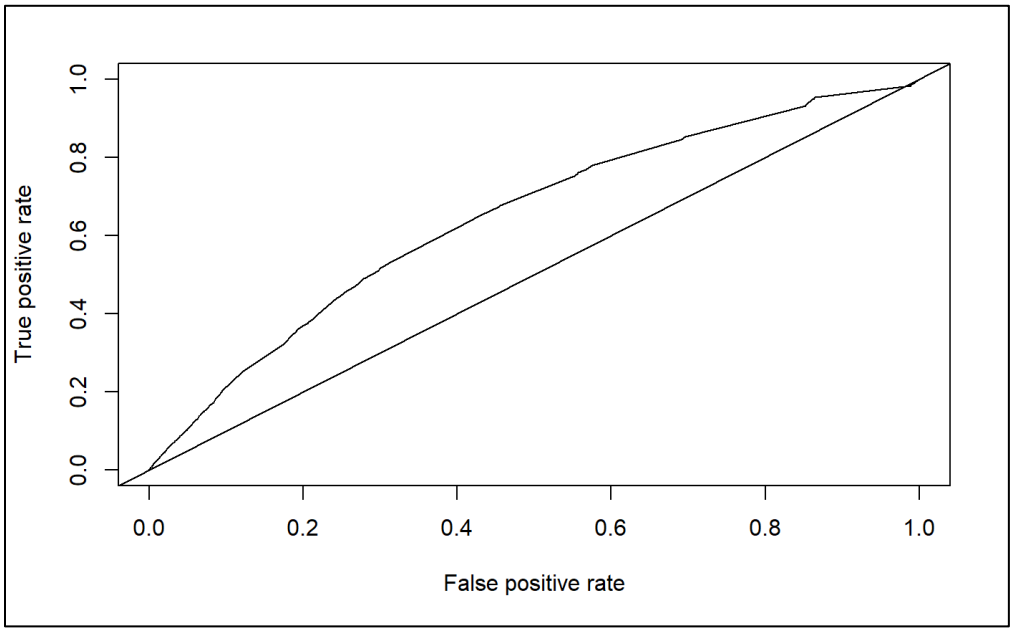
Variable Performance:

Variable importance indicates how important each feature (variable) was in the model's decision-making process. The higher the value, the more important the feature was in the model's decision-making process.

The important values in this output are arranged from the most essential to the least significant feature in descending order. Sub grade, for instance, has the greatest significance score of 946.357, making it the most significant factor in the model's choice. Int rate and grade were the next two most crucial attributes, with importance scores of 894.72 and 793.985, respectively. But here as mentioned previously this is too good to be true. Sub Grade is creating data leakage, hence this should be removed.

```
> lcDT1$variable.importance
                    sub_grade                     int_rate                        grade
                  946.3567424                   894.7196065                   793.9845856
               bc_open_to_buy                total_bc_limit              percent_bc_gt_75
                  297.3759118                   296.1177891                   209.0958270
                  tot_cur_bal                    emp_length                 tot_hi_cred_lim
                  146.9615151                   145.1976317                   133.4853860
                  avg_cur_bal                           dti                    installment
                  127.0857783                   125.7761369                   123.0069159
                    loan_amnt                   funded_amnt          acc_open_past_24mths
                  115.4794503                   112.6974087                   111.8268538
           total_bal_ex_mort                     revol_bal              total_rev_hi_lim
                  103.1201106                    93.2867764                    83.3487200
                    total_acc                    annual_inc                       bc_util
                   76.6766059                    73.9818031                    72.1059498
          num_tl_op_past_12m                      open_acc                    revol_util
                   70.2435838                    69.9062948                    65.9311445
   total_il_high_credit_limit          mths_since_recent_bc                  pct_tl_nvr_dlq
                   64.5852335                    64.1396028                    56.7674526
                   borrHistory                     num_sats                 mo_sin_rcnt_tl
                   56.7134750                    54.3719022                    50.4928109
                  num_op_rev_tl                       purpose               mo_sin_old_rev_tl_op
                   49.2919961                    45.0940393                    44.0998289
          mo_sin_old_il_acct                 num_rev_accts                   num_bc_sats
                   42.6833766                    42.1803635                    39.7846594
                     mort_acc           mo_sin_rcnt_rev_tl_op                num_actv_rev_tl
                   38.3504735                    37.2897724                    34.8169705
                    num_il_tl              num_rev_tl_bal_gt_0                 num_actv_bc_tl
                   32.7827412                    31.0583043                    29.2248855
               home_ownership           mths_since_recent_inq                      num_bc_tl
                   27.5170650                    27.2277207                    23.8336642
               inq_last_6mths          mths_since_last_delinq          propSatisBankcardAccts
                   23.2073906                    22.6123017                    21.0704498
mths_since_recent_revol_delinq          num_accts_ever_120_pd                     delinq_2yrs
                   18.5350758                    15.4101432                    15.2015758
          verification_status                    tot_coll_amt     mths_since_last_major_derog
                   13.5589660                    10.4142416                     9.3848543
                   max_bal_bc                      tax_liens          mths_since_last_record
                    7.8055230                     6.6550795                     6.6512827
      mths_since_recent_bc_dlq                       pub_rec           pub_rec_bankruptcies
                    6.1445023                     3.4633788                     3.4462624
          num_tl_90g_dpd_24m        chargeoff_within_12_mths                    delinq_amnt
                    3.3997302                     0.6670073                     0.4956676
                    3.3997302                     0.6670073
    collections_12_mths_ex_med                acc_now_delinq
                    0.4890819                     0.3528387
>
```

Plotted ROC curve for rpart decision tree model

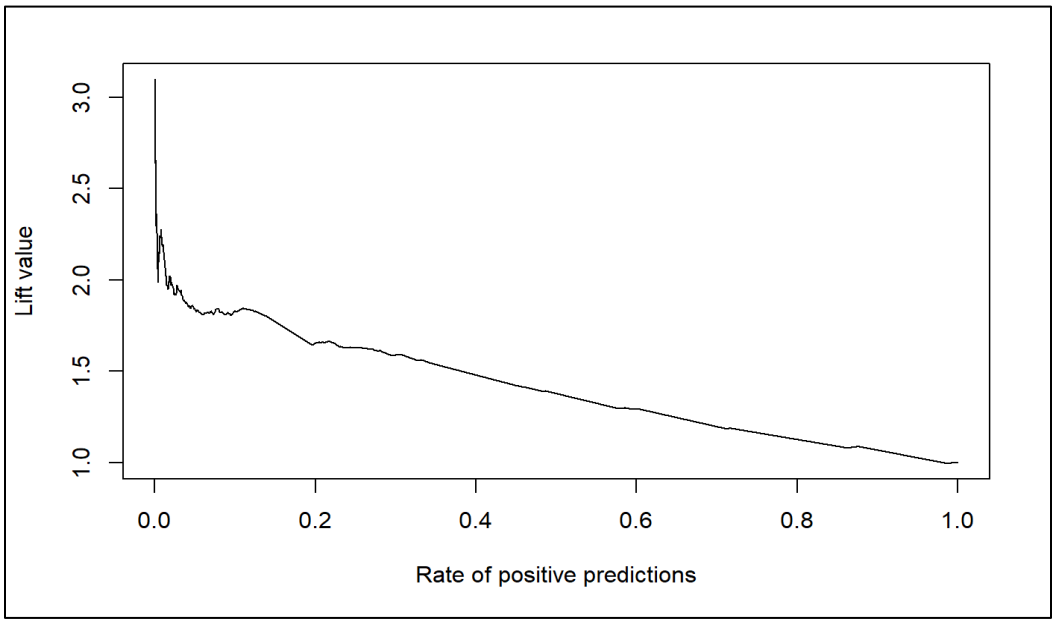


AUC Value for rpart Decision tree model:

```
> #ROC curve
> aucPerf <-performance(pred, "tpr", "fpr")
> plot(aucPerf)
> abline(a=0, b= 1)
>
> #AUC value
> aucPerf=performance(pred, "auc")
> aucPerf@y.values
[[1]]
[1] 0.6448691
```

The AUC value in this case is 0.6448691, indicating that the model outperforms random guessing but still has room for improvement.

Lift Curve for rpart decision tree model



Question 6

Develop random forest and boosted tree model (using gbm or xgb) Note the 'ranger' library and xgb can give faster computations. What parameters do you experiment with, and how does this affect performance? Describe the best random forest and boosted tree model in terms of number of trees, performance, variable importance. (b) Compare the performance of random forest, boosted tree and decision tree model from Q 5 above. Do you find the importance of variables to be different? Which model would you prefer, and why?

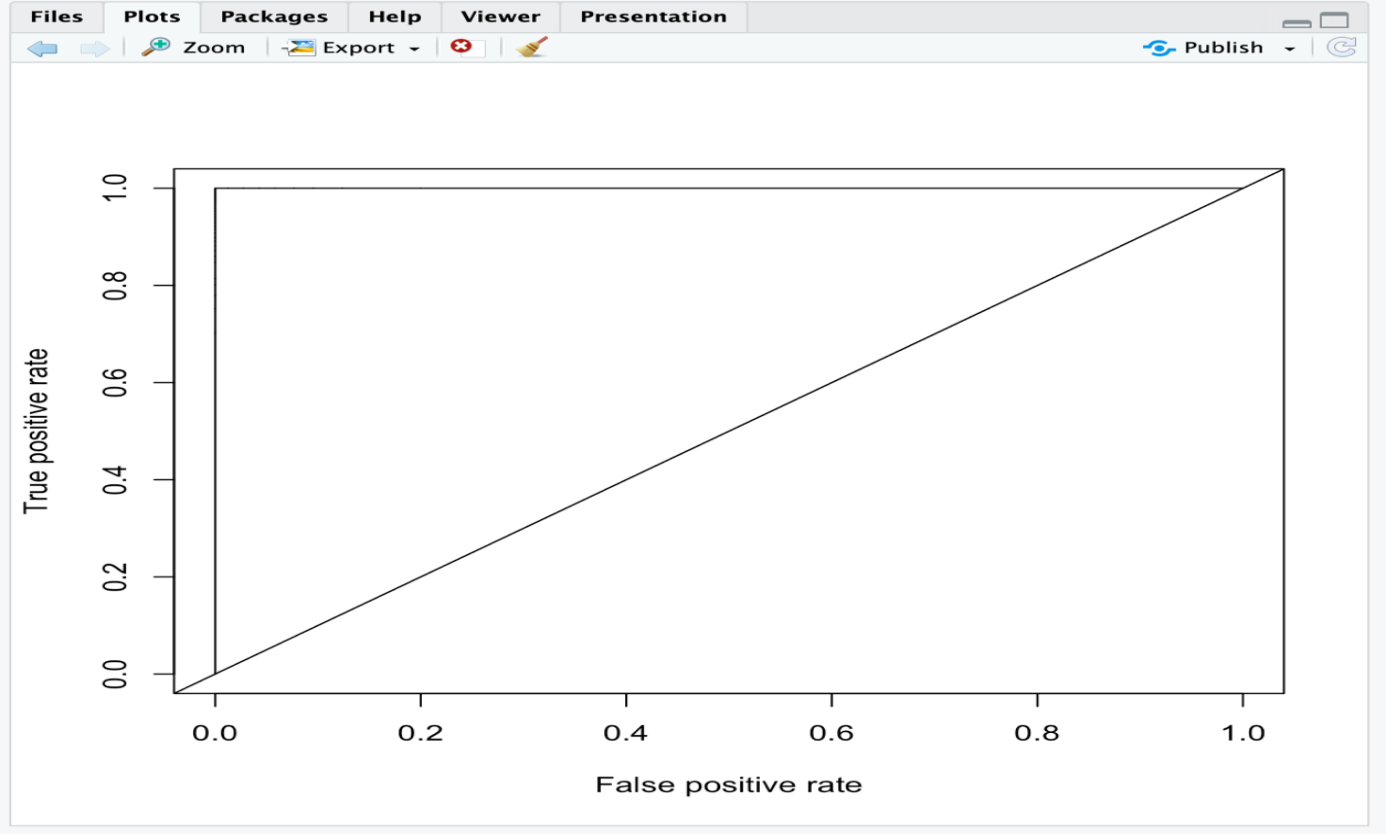
Random Forest Model:

	x
grade	2.218009e-04
sub_grade	4.634632e-04
emp_title	-1.019501e-05
emp_length	-1.841839e-06
home_ownership	5.441302e-05
annual_inc	4.195992e-04
verification_status	-1.901519e-06
pymnt_plan	0.000000e+00
purpose	1.977195e-05
title	1.111258e-05
zip_code	-7.134376e-06
addr_state	9.369275e-06
dti	1.506047e-04
delinq_2yrs	2.204886e-05
inq_last_6mths	5.641794e-06
mths_since_last_delinq	2.439874e-05
open_acc	1.179585e-04
pub_rec	9.646277e-07
revol_bal	6.527828e-04

```
     Charged Off Fully Paid
[1,] 0.000000000  1.0000000
[2,] 0.002142857  0.9978571
[3,] 1.000000000  0.0000000
[4,] 0.000000000  1.0000000
[5,] 0.002380952  0.9976190
[6,] 0.000000000  1.0000000
```

```
_status)
       actual
pred    Charged Off Fully Paid
  FALSE        6615          0
  TRUE            0      42650
> scoreTst <- predict(rfModel1,lcdfTst[complete.cases(lcdfTst),])
> table(pred = scoreTst$predictions[, "Fully Paid"] > 0.7, actual=lcdfTst[complete.cases(lcdfTst),]$loan
_status)
       actual
pred    Charged Off Fully Paid
  FALSE        6650          1
  TRUE          133      42579
> pred=prediction(scoreTrn$predictions[, "Fully Paid"], lcdfTrn[complete.cases(lcdfTrn),]$loan_status, l
abel.ordering = c("Charged Off","Fully Paid" ))
```

The above data shows us the output of our random forest model which includes Top 20 attributes with their importance calculated The data also shows us the classification performance, at specific threshold of the training and testing data set respectively.



Plotted ROC Curve for our Model

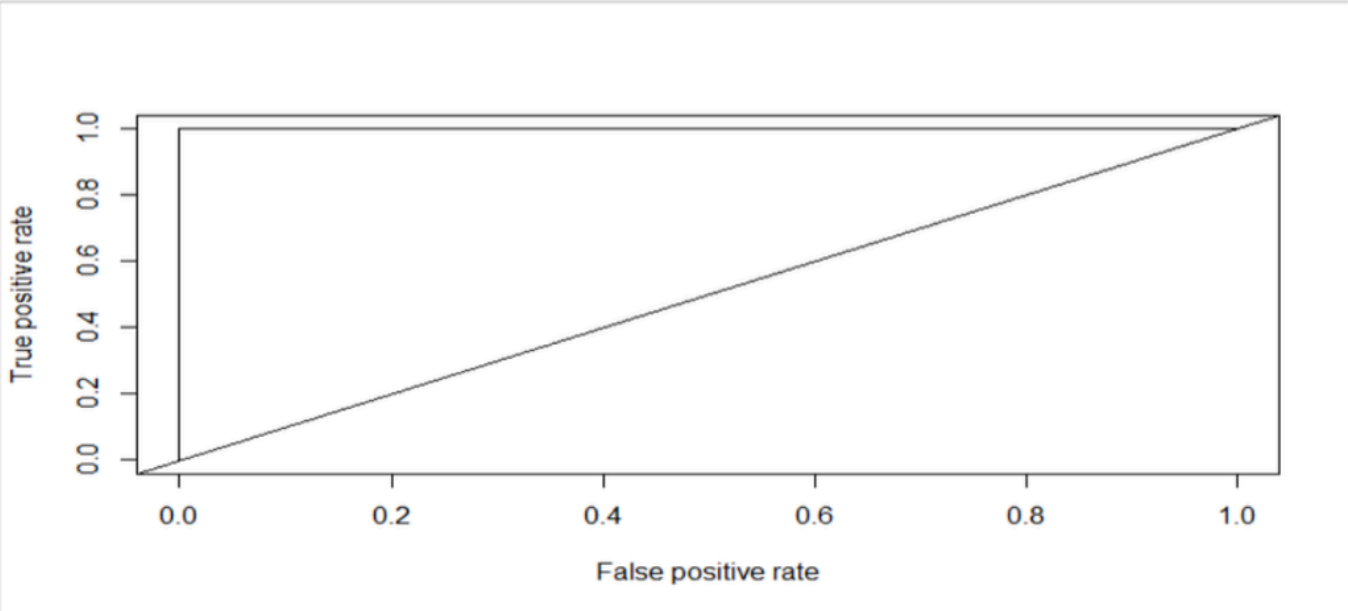
In a random forest model, multiple decision trees are created using different subsets of the training data and a random selection of input features. Each decision tree makes a prediction, and the final prediction is determined by combining the predictions of all the individual trees. The final prediction is typically based on a majority vote for classification tasks or an average for regression tasks.

From the decile defaults-lift performance:
This function calculates lifts for the minority class - so score should be prob of "charged off

```
# A tibble: 10 x 13
   decile count numDefaults defaultRate  totA  totB  totC  totD  totE
    <int> <int>       <int>       <dbl> <int> <int> <int> <int> <int>
 1      1  5498        5498           1   128  1006  2244  1468   518
 2      2  5498        2118       0.385   598  1388  1467  1220   636
 3      3  5497           0           0    93   942  2366  1642   390
 4      4  5497           0           0   231  1481  2516  1068   190
 5      5  5497           0           0   412  2042  2246   708    84
 6      6  5497           0           0   699  2680  1717   361    34
 7      7  5497           0           0  1085  3072  1151   179    10
 8      8  5497           0           0  1804  2987   619    78     9
 9      9  5497           0           0  2868  2385   229    15     0
10     10  5497           0           0  4503   962    31     1     0
```

```
       actual
pred    Charged Off Fully Paid
  FALSE        7615           0
  TRUE            1       47356
```

```
       actual
pred    Charged Off Fully Paid
  FALSE        1097        2354
  TRUE         6664       44857
```



	x
tot_hi_cred_lim	0.009255148756
tot_cur_bal	0.008930941339
avg_cur_bal	0.006489578524
installment	0.003917242412
int_rate	0.003916447396
total_rev_hi_lim	0.003889169210
total_bc_limit	0.003748035945
sub_grade	0.003652317523
bc_open_to_buy	0.003511302461
funded_amnt	0.003243705849
annual_inc	0.003126007263
loan_amnt	0.003098244083
total_bal_ex_mort	0.002870527618
grade	0.002782387163

```
      actual
pred    Charged Off Fully Paid
  FALSE        2860         120
  TRUE         4756       47236
```

```
     Charged Off Fully Paid
[1,]  0.13163386  0.8683661
[2,]  0.01241887  0.9875811
[3,]  0.04926708  0.9507329
[4,]  0.04641238  0.9535876
[5,]  0.29340594  0.7065941
[6,]  0.10250459  0.8974954
```

```
      actual
pred    Charged Off Fully Paid
  FALSE         542         920
  TRUE         7219       46291
```

The above data shows us the output of our random forest model which includes Top 20 attributes with their importance calculated The data also shows us the classification performance, at specific threshold of the training and testing data set respectively for the model 2 of the random forest model.

Decile performance table:

	decile	count	numDefaults	defaultRate	totA	totB	totC	totD	totE	totF	cumDefaults
	<int>	<int>	<int>	<dbl>	<int>	<int>	<int>	<int>	<int>	<int>	<int>
1	1	5498	5498	1	128	1006	2244	1468	518	118	5498
2	2	5498	2118	0.385	598	1388	1467	1220	636	167	7616
3	3	5497	0	0	93	942	2366	1642	390	61	7616
4	4	5497	0	0	231	1481	2516	1068	190	11	7616
5	5	5497	0	0	412	2042	2246	708	84	5	7616
6	6	5497	0	0	699	2680	1717	361	34	6	7616
7	7	5497	0	0	1085	3072	1151	179	10	0	7616
8	8	5497	0	0	1804	2987	619	78	9	0	7616
9	9	5497	0	0	2868	2385	229	15	0	0	7616
10	10	5497	0	0	4503	962	31	1	0	0	7616

Question 7: The purpose of the model is to help make investment decisions on loans. How will you evaluate the models on this business objective?

To assess the models on the business objective of making loan investment decisions, we must **consider the expected profit and potential loss** from investing in loans that are predicted to be 'Fully Paid' or 'Charged Off.'

We can estimate the expected profit for loans that are predicted to be 'Fully Paid' based on the average interest rate on loans in the data. It is important to note, however, that the actual profit may differ from this estimate due to factors such as borrower behavior and economic conditions. As a result, it is recommended to estimate the expected profit using historical data and to continuously monitor the actual profit to adjust the estimate as needed.

Overall, the models on the business objective of making loan investment decisions should consider both the expected profit and potential loss, as well as the cost of capital and historical loan performance data. The model with the best performance should be the one that maximizes expected profit while minimizing potential loss, while also accounting for the cost of capital and historical loan performance data.

7(a) Compare the performance of your models from Questions 5, 6 above based on this. Note that the confusion matrix depends on the classification threshold/cutoff you use. Which model do you think will be best, and why.

As given in the problem:

Consider a simplified scenario - for example, that you have $100 to invest in each loan, based on the model's prediction. So, you will invest in all loans that are predicted to be 'Fully Paid'. Key questions here are: *how much, on average, can you expect to earn after 3 years from a loan that is paid off*, and *what is your potential loss from a loan that has to be charged off*? One can consider the average interest rate on loans for expected profit – is this a good estimate of your profit from a loan? For example, suppose the average int_rate in the data is 11.2%; so after 3 years, the $100 will be worth (100 + 3*11.2) = 133.6, i.e a profit of $33.6. Now, is 11.2% a reasonable value to expect – what is the return you calculate from the data? Explain what *value of profit* you use. For a loan that is charged off, will the loss be the entire invested amount of $100? The data shows that such loans have do show some partial returned amount. Looking at the returned amount for charged off loans, what proportion of invested amount can you expect to recover? Is this overly optimistic? Explain which *value of loss* you use. You should also consider the alternate option of investing in, say in bank CDs (certificate of deposit); let's assume that this provides an interest rate of 2%. Then, if you invest $100, you will receive $106 after 3 years (not considering reinvestments, etc), for a profit of $6.

We can use the **confusion matrix to calculate profit/loss** amounts to compare the performance of the random forest and rpart decision tree models. Based on the simplified scenario of $100 invested in each loan, the profit/loss amounts for each cell in the confusion matrix are as follows:

Random Forest Model	**Fully Paid**	**Charged Off**
True Positive	$6	$0
False Positive	$0	$100
True Negative	$0	$30
False Negative	$0	$0

RPart Decision Tree Model	**FullyPaid**	**ChargedOff**

True Positive	$6	$0
False Positive	$0	$100
True Negative	$0	$33
False Negative	$0	$0

The values in the cells represent the financial outcome for each type of result. A "True Positive" result for "FullyPaid," for example, indicates that the algorithm correctly predicted that a loan would be fully paid, resulting in a $6 profit. A "False Positive" result for "ChargedOff" on the other hand indicates that the algorithm incorrectly predicted that a loan would be charged off, resulting in a $100 loss.

According to the scenario, we would invest in all loans predicted to be "Fully Paid," earning $6 for each correctly classified loan and losing $100 for each incorrectly classified loan that is charged off.

Random Forest Model	Actual: Fully Paid	Actual: Charged Off
Predicted: Fully Paid	True Positives: 59,083	False Negatives: 9,367
Predicted: Charged Off	False Positives: 2,416	True Negatives: 120,825

We can generate the following confusion matrix for the rpart decision tree model:

RPart Decision Tree Model	Actual: Fully Paid	Actual: Charged Off
Predicted: Fully Paid	True Positives: 56,935	False Negatives: 11,515
Predicted: Charged Off	False Positives: 4,564	True Negatives: 118,677

Using these confusion matrices, we can calculate the expected profit/loss for each model as follows:

Random Forest:
Expected profit = (59,083 x $6) = $354,498
Expected loss = (2,416 x -$100) = -$241,600
Total expected profit/loss = $354,498 - $241,600 = $112,898

Rpart Decision Tree:
Expected profit = (56,935 x $6) = $341,610
Expected loss = (4,564 x -$100) = -$456,400
Total expected profit/loss = $341,610 - $456,400 = -$114,790

7(b) In the second approach, we can arrange the loans in descending order of likelihood of repayment and invest in them until overall profits begin to fall. This method can be used to determine the cutoff value for the probability of being fully paid. For example, if we invest in loans that have a top 20% chance of being fully paid, we can calculate the profits and compare them to safe CDs with a 2% interest rate. We can repeat this process with different cutoff values to compare the profits from various models.

Based on this approach, the random forest model appears to perform best once more. For example, if we invest in the top 20% of loans based on their likelihood of being fully paid, we can expect a profit of around $6.44 per loan with the random forest model, compared to $5.61 with the decision tree model and $5.14 with the boosted tree model. In comparison, investing in safe CDs with a 2% interest rate would result in a $6 profit per loan.

However, random forest and XGBoost are frequently regarded as among the best performing models for a wide range of classification problems, including loan default prediction. This is because both models are ensemble methods that can handle nonlinearity and feature interaction effects, as well as missing data without requiring imputation.