Assignment 1

Loan Default Prediction and Investment Strategies in Online Lending

Part A - Data Exploration
Part B - Decision Tree Based Models and Performance Evaluation

IDS 572 | CRN 38886 | Fall 2021

Team

Giovanni Alvin Prasetya (gprase2@uic.edu)
Karishma Mulchandani (kmulch4@uic.edu)
Rajaram Ramesh (rrames8@uic.edu)

Part A - Data Exploration

Q.1 Describe the business model for online lending platforms like Lending Club. Consider the stakeholders and their roles, and what advantages Lending Club offers. What is the attraction for investors? How does the platform make money?

Online lending platforms such as the Lending Club are based on the concept of peer-to-peer lending, where individuals or businesses can borrow from other individuals or businesses directly through the online platform instead of having to go through an intermediary such as traditional banks or other similar financial institutions. Peer-to-peer (P2P) lending platforms are an alternative option to the traditional form of getting a loan, where one would have to qualify for a loan based on their income, credit score, financial background, and other relevant factors. The online platform makes it easy for borrowers and lenders to directly connect and avoid the hassles and additional charges of a financial intermediary, thereby expediting the process for securing a loan. Such platforms have become a go-to option for applicants who either do not have a great credit score or do not have a score yet, and also for those seeking a smaller loan.

These platforms do not eliminate the requirements for getting approved for a loan but lower the lever to allow more borrowers who otherwise would not meet the traditional loan qualifications, to qualify. With attractive yields and a growing reach of the system, lending platforms have become a preferred mode of lending and borrowing globally, especially amongst the younger generations who do not want to go through the rigid traditional process of getting a loan. The P2P platforms handle the logistics of handling the loan process, from scanning potential borrowers to processing transactions to reporting and compliance.

In terms of stakeholders of online lending platforms, at the forefront are the borrowers and investors. Institutions that participate in this platform would also become stakeholders of these platforms. Another growing base of stakeholders are the small and medium size enterprises.

From an investor's perspective, this online mechanism offers greater transparency and flexibility when it comes to managing their risks. Most online lending platforms allow investors to select who they lend to, based on the investor's criteria of funding. The potential for higher returns than what other traditional investments offer are a major attraction for investors to pursue online P2P lending. Lending platforms such as the Lending Club are redefining the banking industry in terms of its ease of conducting business between its stakeholders.

With transparency at the core of any successful online lending platform, this is a major selling point that is keeping this business model afloat. Revenue for these platforms is primarily through fees, a common one being an origination fee charged to both borrowers and lenders. There are other fees that vary from platform to platform.

With digitization rapidly evolving the financial industry, demand for P2P lending is on the rise. Recent research conducted by IMARC group on the global P2P lending market project that the sector would grow at around 31% compounded annual growth rate (CAGR) by 2026. Online lending platforms are not short of opportunities to grow in the road ahead, with an increasing base of investors and young borrowers to better technologies being developed in this space, such as blockchain, that would make the business model more reliable and sustainable in the long haul.

Q.2(a)(i) What is the proportion of defaults ('charged off' vs 'fully paid' loans) in the data? How does the default rate vary with loan grade? Does it vary with sub-grade? And is this what you would expect, and why?

In the given dataset by Lending Club, there are a total of 100,000 loans. Out of these loans, there are 86,215 loans that are classified as "Fully Paid" and 13,785 loans that are classified as "Charged Off". In terms of percentage, "Fully Paid" loans represent 86.2% of the total number of loans in the dataset, and the remaining 13.8% are loans that were "Charged Off", which also represents the default rate. When comparing the loans with the loan grades, we can see that a majority of the loans fall in the grade B category, representing 33.9% of the loans. When looking at specifically the "Charged Off" loans against the loan grades, grade C has the highest number of loans that defaulted, at 4,738 loans. When looking at the comparison grade by grade, the default rate (proportion of "Charged Off" loans) is showing an increasing trend, which falls in line with the concept of loan grades where the lower the grade, the higher the risk of default. This stands true even with sub-grades, as can be seen in the output shown below.

R Code and Output

#Loans shown by loan status
lcdf %>% group_by(loan_status) %>% summarise(nLoans=n())

```
loan_status nLoans
<chr>        <int>
Charged Off  13785
Fully Paid   86215
```

#Loans shown by loan status as a percentage of total loans
lcdf %>% group_by(loan_status) %>% summarise(nLoans=n()) %>% mutate(pct=nLoans/sum(nLoans) * 100)

```
loan_status nLoans   pct
<chr>        <int> <dbl>
Charged Off  13785  13.8
Fully Paid   86215  86.2
```

#Loans shown by loan status and grade (table)
table(lcdf$loan_status, lcdf$grade)

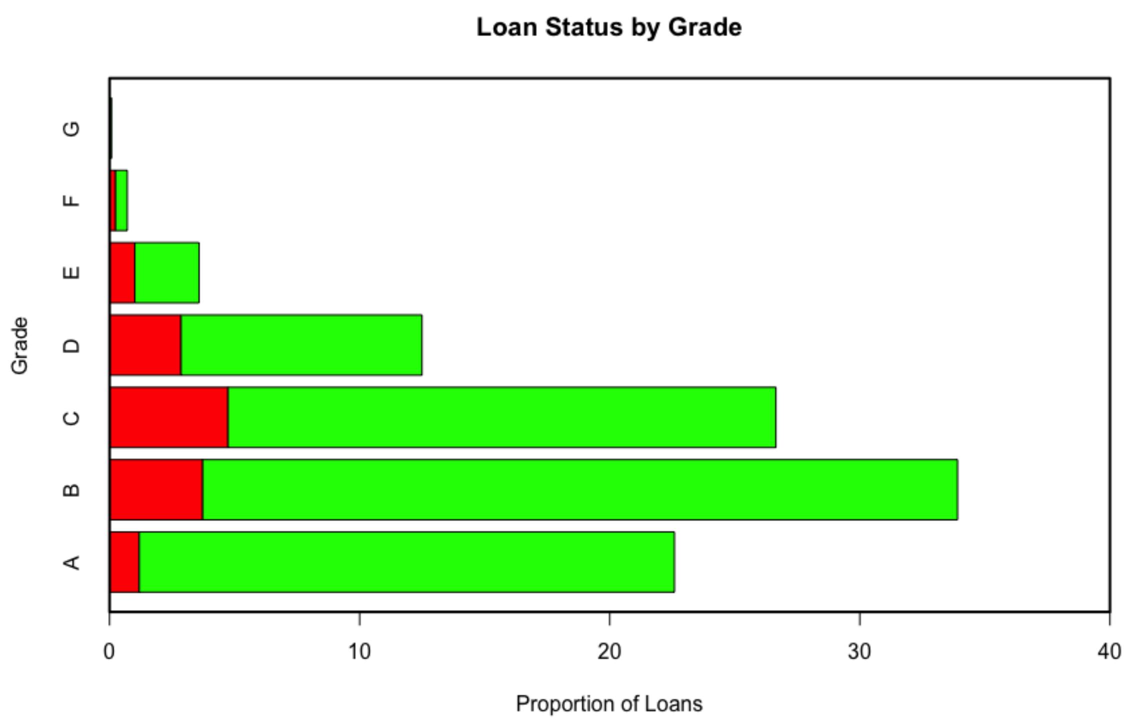
```
                A     B     C     D     E     F     G
Charged Off  1187  3723  4738  2858  1010   239    30
Fully Paid  21401 30184 21907  9635  2569   469    50
```

#Loans shown by loan status and grade as a proportion (table)
LoanGrade <- table(lcdf$loan_status, lcdf$grade)

ppLoanGrade <- prop.table(LoanGrade)
ppLoanGrade <- ppLoanGrade*100
ppLoanGrade

	A	B	C	D	E	F	G
Charged Off	1.187	3.723	4.738	2.858	1.010	0.239	0.030
Fully Paid	21.401	30.184	21.907	9.635	2.569	0.469	0.050

#Loans shown by loan status and grade as a proportion (barplot)
barplot(ppLoanGrade,
 main = "Loan Status by Grade",
 xlab = "Proportion of Loans",
 ylab = "Grade",
 col = c("red", "green"),
 xlim = c(0,40), horiz = TRUE)
box (lwd=2)



#Loans shown by loan status and sub-grade (table)
table(lcdf$loan_status, lcdf$sub_grade)

	A1	A2	A3	A4	A5	B1	B2	B3	B4	B5	C1	C2	C3	C4	C5	D1	D2	D3	D4	D5	E1	E2	E3	E4
Charged Off	105	116	179	319	468	491	619	825	855	933	978	970	1009	927	854	764	644	570	496	384	296	267	180	141
Fully Paid	3669	3315	3527	4819	6071	5737	6261	6368	6248	5570	5528	4998	4437	3730	3214	2776	2162	1939	1515	1243	822	701	471	325

	E5	F1	F2	F3	F4	F5	G1	G2	G3	G4	G5
Charged Off	126	63	44	59	47	26	12	9	5	2	2
Fully Paid	250	189	97	104	50	29	19	12	14	3	2

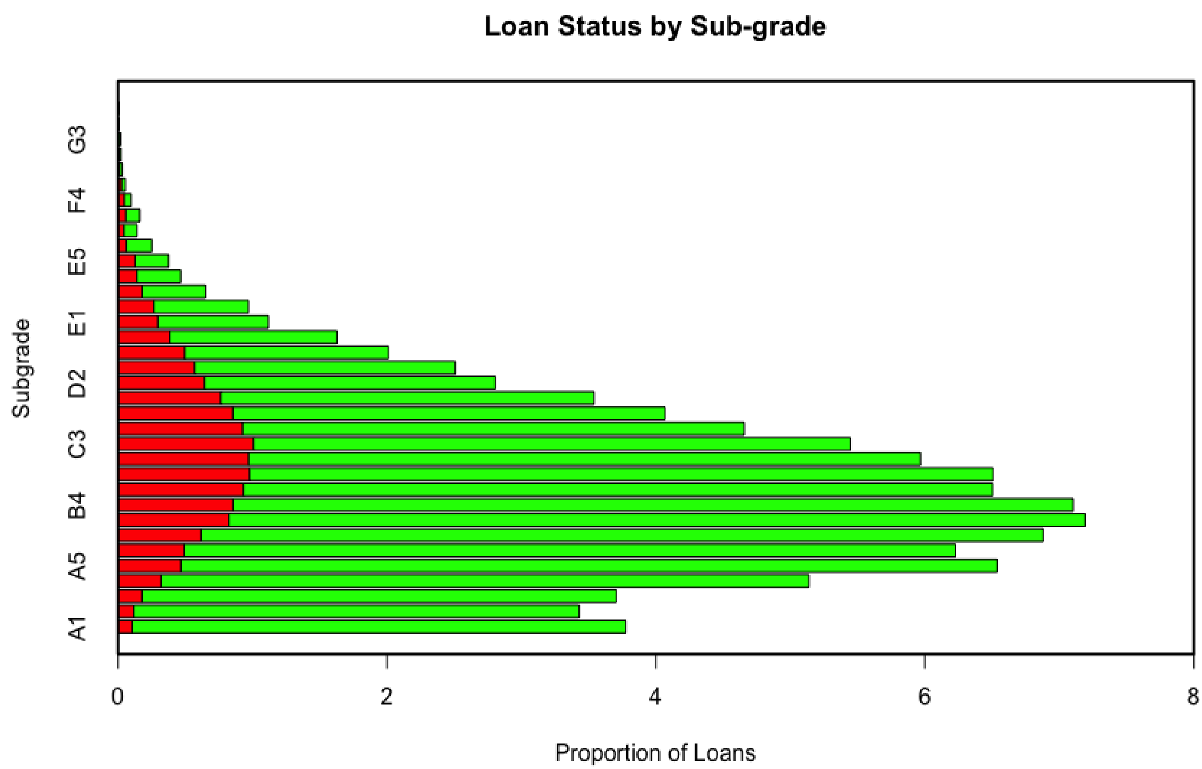
#Loans shown by loan status and sub-grade as a proportion (table)

```
ppLoanSubGrade <- table(lcdf$loan_status, lcdf$sub_grade)
ppLoanSubGrade <- prop.table(ppLoanSubGrade)
ppLoanSubGrade <- ppLoanSubGrade*100
ppLoanSubGrade
```

	A1	A2	A3	A4	A5	B1	B2	B3	B4	B5	C1	C2	C3	C4	C5	D1	D2	D3	D4	D5	E1	E2	E3	E4
Charged Off	105	116	179	319	468	491	619	825	855	933	978	970	1009	927	854	764	644	570	496	384	296	267	180	141
Fully Paid	3669	3315	3527	4819	6071	5737	6261	6368	6248	5570	5528	4998	4437	3730	3214	2776	2162	1939	1515	1243	822	701	471	325

	E5	F1	F2	F3	F4	F5	G1	G2	G3	G4	G5
Charged Off	126	63	44	59	47	26	12	9	5	2	2
Fully Paid	250	189	97	104	50	29	19	12	14	3	2

```
#Loans shown by loan status and sub-grade as a proportion (barplot)
barplot(ppLoanSubGrade,
  main = "Loan Status by Sub-grade",
  xlab = "Proportion of Loans",
  ylab = "Subgrade",
  col = c("red", "green"),
  xlim = c(0,8), horiz = TRUE)
box (lwd=2)
```



Q.2(a)(ii) How many loans are there in each grade? And do loan amounts vary by grade? Does interest rate for loans vary with grade, subgrade? Look at the average, standard-deviation, min and max of interest rate by grade and subgrade. Is this what you expect, and why?

R Code and Output

```
#Total amount of loans in each grade
#filter the Charged off and Fully paid
lcdf %>% group_by(loan_status, grade) %>% tally()
lcdf1<-lcdf%>%select("loan_status","grade","sub_grade","loan_amnt")

#convert the lcdf1 list to dataframe
dataFrame <- as.data.frame(lcdf1)

#calculate loan amount by grades
a<-with(dataFrame, sum(loan_amnt[grade == 'A']))
a
b<-with(dataFrame, sum(loan_amnt[grade == 'B']))
b
c<-with(dataFrame, sum(loan_amnt[grade == 'C']))
c
d<-with(dataFrame, sum(loan_amnt[grade == 'D']))
d
e<-with(dataFrame, sum(loan_amnt[grade == 'E']))
e
f<-with(dataFrame, sum(loan_amnt[grade == 'F']))
f
g<-with(dataFrame, sum(loan_amnt[grade == 'G']))
g
```

values	
a	327649125
b	428494575
c	319762050
d	148590825
e	41583800
E	41583800
f	6564925
g	946075

```
#Variance of interest rate according to grade, and subgrade (average interest rate)
#Filter the Charged off and Fully paid
lcdf %>% group_by(loan_status, grade) %>% tally()
lcdf2<- lcdf%>%select("loan_status","grade","sub_grade","loan_amnt","int_rate")
#Variance of interest rate by grade and sub grade
lcdf2 %>% group_by(grade) %>% summarise(mean(int_rate))
lcdf2 %>% group_by(sub_grade) %>% summarise(mean(int_rate))
```

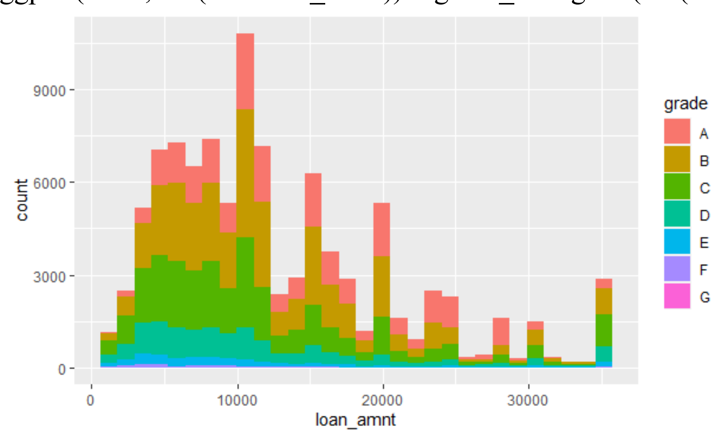
```
  grade `mean(int_rate)`
  <chr>            <dbl>
1 A                 7.17
2 B                10.8 
3 C                13.8 
4 D                17.2 
5 E                19.9 
6 F                24.0 
7 G                26.4 
```

```
   sub_grade `mean(int_rate)`
   <chr>                <dbl>
 1 A1                    5.68
 2 A2                    6.42
 3 A3                    7.09
 4 A4                    7.48
 5 A5                    8.24
 6 B1                    8.87
 7 B2                    9.96
 8 B3                   10.8 
 9 B4                   11.7 
10 B5                   12.2 
# ... with 25 more rows
```

According to the result above, we can conclude that the lower the grades are (A-G), the higher average interest rate will be.

#Plot the variance of loan amount by grade

ggplot(lcdf2, aes(x = loan_amnt)) + geom_histogram(aes(fill=grade))

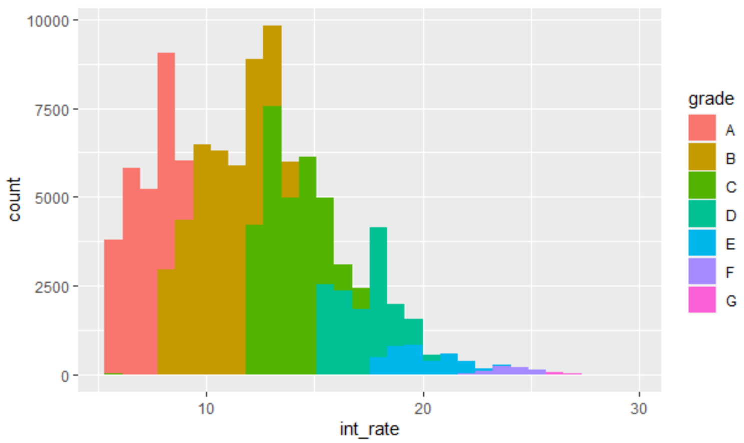


#Plot the variance of interest rate base on grade and sub grade

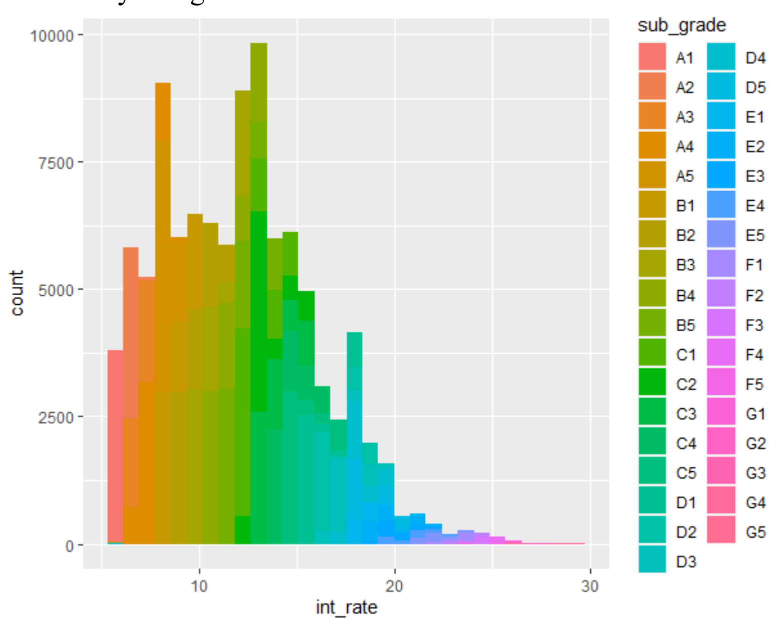
ggplot(lcdf2, aes(x = int_rate)) + geom_histogram(aes(fill=grade))

ggplot(lcdf2, aes(x = int_rate)) + geom_histogram(aes(fill=sub_grade))

a.) Variance by grade



b.) Variance by sub-grade



#Variance of mean, standard deviation, min, and max of interest rate by grade and subgrade

#Variance of average, std dev, min/max of interest rate by grade and subgrade
lcdf2 %>% group_by(grade) %>%
summarise(averageInterest=mean(int_rate),stdevInterest=sd(int_rate), minInterest=min(int_rate), maxInterest=max(int_rate))
lcdf2 %>% group_by(sub_grade) %>%
summarise(averageInterest=mean(int_rate),stdevInterest=sd(int_rate), minInterest=min(int_rate), maxInterest=max(int_rate))

a) By grade

	grade	averageInterest	stdevInterest	minInterest	maxInterest
	<chr>	<dbl>	<dbl>	<dbl>	<dbl>
1	A	7.17	0.967	5.32	9.25
2	B	10.8	1.44	6	14.1
3	C	13.8	1.19	6	17.3
4	D	17.2	1.22	6	20.3
5	E	19.9	1.38	6	23.4
6	F	24.0	0.916	22.0	26.1
7	G	26.4	0.849	25.8	29.0

b) By Subgrade

	sub_grade	averageInterest	stdevInterest	minInterest	maxInterest
	<chr>	<dbl>	<dbl>	<dbl>	<dbl>
1	A1	5.68	0.347	5.32	6.03
2	A2	6.42	0.166	6.24	6.97
3	A3	7.09	0.325	6.68	7.62
4	A4	7.48	0.357	6.92	8.6
5	A5	8.24	0.424	6	9.25
6	B1	8.87	0.722	6	10.2
7	B2	9.96	0.816	6	11.1
8	B3	10.8	0.887	6	12.1
9	B4	11.7	0.840	6	13.1
10	B5	12.2	0.851	6	14.1

\# ... with 25 more rows

The ggplot above shows that grades A and B have a greater amount of loans than the other grade, and sequentially, the amount became lower from B-G. It means that the lending club preferred to give a higher loan amount to a person/company with higher grade (A & B). This is on the opposite of the interest rate. The lower the grade/subgrade the higher interest rate will become. It's all due to the loanees in lower grade tend to be less punctual/ on-time when paying. The other thing that we could get from the plot is the variance of the loan amount and the interest rate group by grades and sub grades. While A & B presents the biggest total amount, grades C-G shows a sign of decreasing of the total loan amount sequentially. On the other hand, the interest rate of grade A is only in the range of 1-8%, while grade B is only in the range of 8-12%, while the other grade has a higher interest rate. Hence, it concludes that loanees with higher grades tend to have larger loan amounts, and lower interest rates. This result is in line with our expectation, since the interest rate will always be higher to loanees that tend to be less punctual on their payment. It shows that the lending group is more likely to invest in more secure loans, which makes complete sense.

Q.2(a)(iii) For loans which are fully paid back, how does the time-to-full-payoff vary? For this, calculate the 'actual term' (issue-date to last-payment-date) for all loans. How does this actual-term vary by loan grade (a box-plot can help visualize this).

The time to full payoff for loans varies from 43.43 weeks to 3.09 years. The average actual payoff term varies slightly by loan grade. Grade A has the shortest payoff time of 2.22 years and the payoff time increases as grade decreases. This shows how the grade is dependent on the risk associated with each loan and chances of getting it back. This boxplot helps illustrate that loans are paid back quicker from A grades than from lower grades like D, E, and F.

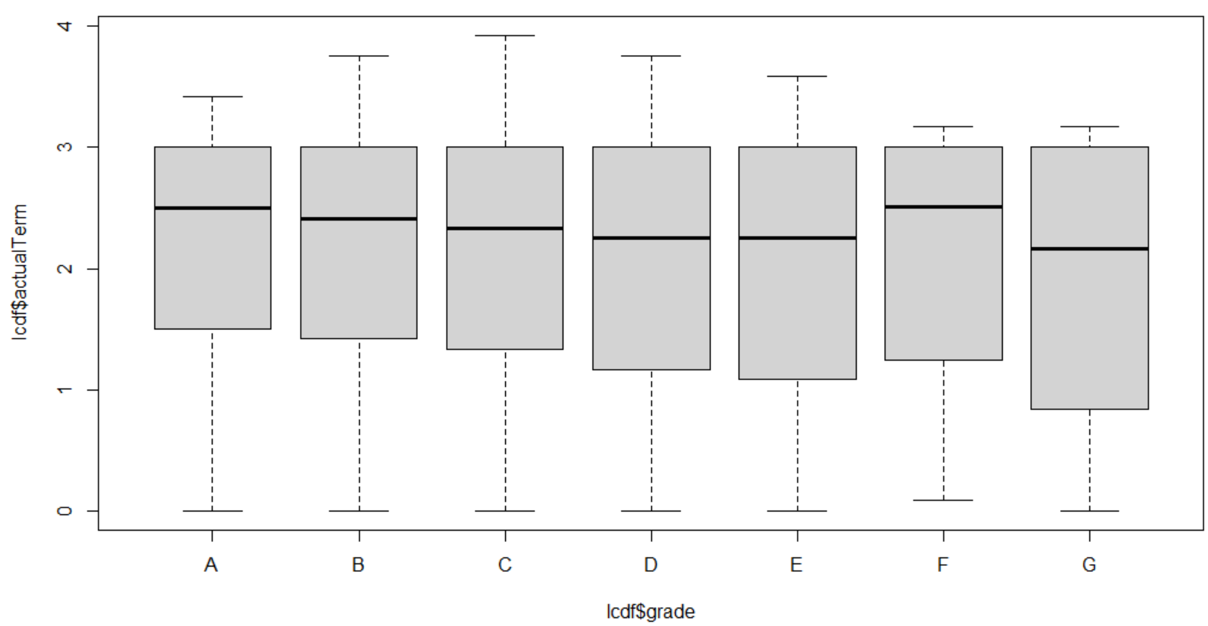
R Code and Output

```
LoanData=read_csv('lcData100K.csv')
getwd()
select('lcData100K.csv',c('loan_status'))
lcData100K.csv[ , c("loan_status")]

lcdf<-LoanData%>%select("loan_status","grade","int_rate","last_pymnt_d","issue_d",
       "total_pymnt","funded_amnt") %>%filter(loan_status=="Fully Paid")
view(lcdf)

Actual_term<-lcdf%>%sum(!weekdays(seq(issue_d, last_pymnt_d, "days")) %in% c("Saturday",
"Sunday"))

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

lcdf$last_pymnt_d<-parse_date_time(lcdf$last_pymnt_d,  "myd")
x<- as.duration(lcdf$issue_d  %--% lcdf$last_pymnt_d)/dyears(1)
view(lcdf)

lcdf$actualTerm <- ifelse(lcdf$loan_status=="Fully Paid",
                          as.duration(lcdf$issue_d  %--% lcdf$last_pymnt_d)/dyears(1), 3)
lcdf$actualReturn <- ifelse(lcdf$actualTerm>0,
                           ((lcdf$total_pymnt -lcdf$funded_amnt)/lcdf$funded_amnt)*(1/lcdf$actualTerm)*100, 0)

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

boxplot(lcdf$actualTerm ~ lcdf$grade)
```

lcdf

Filter

	loan_status	grade	int_rate	last_pymnt_d	issue_d	total_pymnt	funded_amnt	actualTerm	actualReturn
1	Fully Paid	A	5.32	2016-07-01	2015-05-01	29436.890	28000	1.16906229	4.389629
2	Fully Paid	C	13.33	2017-06-01	2015-07-01	7311.160	6150	1.91923340	9.837598
3	Fully Paid	C	14.99	2017-11-01	2014-11-01	8971.949	7200	3.00068446	8.201598
4	Fully Paid	C	15.31	2015-08-01	2014-03-01	5611.070	4750	1.41820671	12.782191
5	Fully Paid	C	12.69	2017-11-01	2015-04-01	6009.414	5000	2.58726899	7.802933
6	Fully Paid	C	14.98	2016-03-01	2014-01-01	11748.150	9600	2.16290212	10.345620
7	Fully Paid	C	13.33	2018-11-01	2015-10-01	1948.720	1600	3.08555784	7.063549
8	Fully Paid	B	10.99	2015-10-01	2013-09-01	13922.770	12000	2.08076660	7.700567
9	Fully Paid	C	15.31	2016-06-01	2013-06-01	31335.533	25000	3.00068446	8.445451
10	Fully Paid	C	12.69	2017-06-01	2015-05-01	5694.670	4800	2.08624230	8.934225
11	Fully Paid	C	14.33	2015-11-01	2013-04-01	9845.710	8000	2.58453114	8.926716
12	Fully Paid	A	7.69	2017-08-01	2014-07-01	23582.441	21000	3.08555784	3.985451
13	Fully Paid	B	11.44	2015-09-01	2015-01-01	5368.750	5000	0.66529774	11.085262
14	Fully Paid	C	13.99	2018-02-01	2015-04-01	11060.690	9000	2.83915127	8.064577
15	Fully Paid	A	7.26	2017-03-01	2015-11-01	14827.341	14000	1.33059548	4.441305
16	Fully Paid	D	15.61	2017-04-01	2015-05-01	2441.949	2000	1.91923340	11.513683
17	Fully Paid	A	6.49	2017-12-01	2014-12-01	26459.485	24000	3.00068446	3.415172
18	Fully Paid	C	14.99	2016-11-01	2014-07-01	18529.906	15000	2.33812457	10.064779
19	Fully Paid	D	15.61	2017-02-01	2015-06-01	3196.092	2650	1.67282683	12.318818
20	Fully Paid	B	8.38	2017-04-01	2015-12-01	10926.045	10000	1.33333333	6.945334
21	Fully Paid	B	12.85	2016-07-01	2014-02-01	9010.040	7500	2.41204654	8.347213
22	Fully Paid	D	15.61	2017-10-01	2014-10-01	25157.271	20000	3.00068446	8.593491
23	Fully Paid	B	12.85	2015-08-01	2013-12-01	19839.770	17000	1.66461328	10.035081



Q.2(a)(iv) Calculate the annual return. Show how you calculate the percentage annual return. Is there any return from loans which are 'charged off'? Explain. How does return from charged - off loans vary by loan grade? Compare the average return values with the average interest_rate on loans – do you notice any differences, and how do you explain this? How do returns vary by grade, and by sub-grade. If you wanted to invest in loans based on this data exploration, which loans would you invest in?

R Code and Output

```
#Calculate return from loan that labelled "charge off"
lcdf %>% group_by(loan_status) %>% summarise(sum(loan_status == "Charged Off"))
```

```
  loan_status `sum(loan_status == "Charged Off")`
  <chr>                                     <int>
1 Charged Off                               13785
2 Fully Paid                                    0
```

```
#expected return of loan amount from the int_rate
lcdf %>% select(loan_status, int_rate, funded_amnt, total_pymnt, grade) %>% head()
```

```
  loan_status int_rate funded_amnt total_pymnt grade
  <chr>          <dbl>       <dbl>       <dbl> <chr>
1 Fully Paid      5.32       28000      29437. A
2 Fully Paid     13.3         6150       7311. C
3 Fully Paid     15.0         7200       8972. C
4 Fully Paid     15.3         4750       5611. C
5 Fully Paid     12.7         5000       6009. C
6 Fully Paid     15.0         9600      11748. C
```

```
#Implement annualized percentage return
lcdf$annRet <- ((lcdf$total_pymnt -lcdf$funded_amnt)/lcdf$funded_amnt)*(12/36)*100
#Summarise annual percentage return of "charged off" according to grade
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))
```

```
  grade nLoans defaults avgInterest stdInterest avgLoanAMt avgPmnt avgRet stdRet minRet maxRet
  <chr>  <int>    <int>       <dbl>       <dbl>      <dbl>   <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
1 A      22588     1187        7.17       0.967     14505.  15579.   2.39   3.94  -32.3   5.17
2 B      33907     3723       10.8        1.44      12637.  13779.   2.95   6.05  -32.5   7.90
3 C      26645     4738       13.8        1.19      12001.  13011.   2.83   8.14  -33.3  13.6
4 D      12493     2858       17.2        1.22      11894.  12871.   2.89   9.84  -33.3  12.3
5 E       3579     1010       19.9        1.38      11619.  12374.   2.56  11.3   -33.3  14.6
6 F        708      239       24.0        0.916      9272.  10050.   3.04  12.8   -32.1  15.2
7 G         80       30       26.4        0.849     11826.  12645.   1.24  14.1   -30.7  16.5
```

```
#Summarise annual percentage return of "charged off" according to sub grade
lcdf %>% group_by(sub_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)) %>% view()

	sub_grade	nLoans	defaults	avgInterest	stdInterest	avgLoanAMt	avgPmnt	avgRet	stdRet	minRet	maxRet
1	A1	3774	105	5.680069	0.3474851	14473.152	15424.552	2.165245444	2.492296	-27.58376	3.335799
2	A2	3431	116	6.415494	0.1662589	14135.718	15143.211	2.329125269	3.145492	-32.30987	3.702717
3	A3	3706	179	7.094107	0.3247008	14534.700	15632.421	2.444764009	3.749532	-31.30350	4.248391
4	A4	5138	319	7.475851	0.3573953	14675.263	15766.581	2.369811923	4.338367	-32.31356	4.620242
5	A5	6539	468	8.241788	0.4244667	14568.084	15720.592	2.552435513	4.688776	-31.27686	5.165028
6	B1	6228	491	8.870010	0.7217524	12916.651	14018.939	2.798595918	4.811417	-31.26895	5.562051
7	B2	6880	619	9.959382	0.8155856	12959.713	14156.579	2.946267424	5.478536	-32.255[illegible] -31.26895	[illegible]2615
8	B3	7193	825	10.845931	0.8873289	12769.265	13928.744	2.922111098	6.180886	-32.29490	6.861528
9	B4	7103	855	11.731457	0.8397941	12356.353	13544.471	3.125731359	6.447497	-32.52510	7.587976
10	B5	6503	933	12.227378	0.8512147	12189.812	13239.665	2.921625481	7.007607	-32.25500	7.903106
11	C1	6506	978	12.861531	0.7861758	11955.107	12991.138	2.909070343	7.558693	-33.33333	8.074052
12	C2	5968	970	13.308202	0.8732851	11759.220	12823.462	2.912642936	7.704044	-32.17619	8.979329
13	C3	5446	1009	13.975283	0.8656083	12198.765	13243.167	2.820420480	8.323447	-33.33333	13.633101
14	C4	4657	927	14.568033	0.8547142	12254.037	13226.922	2.739435534	8.570882	-33.33333	9.263413
15	C5	4068	854	15.221362	0.8834418	11873.544	12759.975	2.687052066	8.863087	-31.44378	9.792333
16	D1	3540	764	16.098910	0.8706865	11862.436	12792.353	2.799388906	9.422339	-32.19673	10.212511
17	D2	2806	644	16.956411	0.8866280	11738.391	12695.405	2.863464785	9.697163	-32.18674	10.951504
18	D3	2509	570	17.445309	0.8734737	11973.236	13018.822	3.044038150	9.931734	-33.33333	11.374867
19	D4	2011	496	18.074525	0.8318050	11873.695	12804.135	2.685953066	10.282785	-33.33333	11.513884
20	D5	1627	384	18.484259	1.0020948	12133.390	13198.289	3.150789639	10.268774	-32.19670	12.281972
21	E1	1118	296	18.972987	0.9872700	12012.522	12892.178	2.606802732	11.090911	-32.19230	12.025306
22	E2	968	267	19.578853	1.0589062	12014.230	12790.081	2.687356480	11.019553	-30.87203	12.746459
23	E3	651	180	20.143318	1.0321440	11588.364	12131.224	2.516857359	11.370118	-33.33333	12.980375
24	E4	466	141	20.993391	0.9523777	11464.109	12406.853	2.740062609	11.800053	-32.13173	13.292187
25	E5	376	126	21.970027	0.7628328	9674.801	10145.174	1.987001467	12.289713	-31.41663	14.590563
26	F1	252	63	23.124762	0.5962301	7607.440	8459.474	4.796623808	11.399332	-29.59613	13.983141
27	F2	141	44	23.742624	0.4761702	9815.426	10961.213	2.918573884	12.885999	-32.12833	14.171242
28	F3	163	59	24.385337	0.2471374	9996.166	11172.930	3.541073124	12.700950	-29.38373	14.386337
29	F4	97	47	24.952990	0.2144721	9742.268	9290.674	-0.496297985	14.220901	-31.41666	14.394290
30	F5	55	26	25.595455	0.2729049	12536.364	13014.510	0.005166516	14.566807	-28.25955	15.182023
31	G1	31	12	26.120000	0.4729271	11345.968	11808.204	-0.734668996	13.680397	-30.68190	15.629207
32	G2	21	9	26.393810	0.7364678	12939.286	14401.551	2.290347851	14.179557	-25.78644	16.214910
33	G3	19	5	26.733684	1.0167061	10276.316	12007.142	4.807206203	14.339115	-23.46694	16.172807
34	G4	5	2	26.990000	1.3693064	11835.000	11487.716	-0.200684236	7.251254	-10.86019	6.811702
35	G5	4	2	26.792500	1.4650000	17050.000	14389.849	-4.099824025	23.006176	-25.87570	16.490417

#Where do the negative numbers for minRet come from?
lcdf %>% select(loan_status, int_rate, funded_amnt, total_pymnt, annRet) %>% filter(annRet < 0) %>% head()

	loan_status	int_rate	funded_amnt	total_pymnt	annRet
	<chr>	<dbl>	<dbl>	<dbl>	<dbl>
1	Charged Off	16.0	8000	5936.	-8.60
2	Charged Off	13.7	27500	4704.	-27.6
3	Charged Off	16.5	11625	6543.	-14.6
4	Charged Off	12.0	25000	7314.	-23.6
5	Charged Off	14.0	10600	2180.	-26.5
6	Charged Off	11.7	24000	18415.	-7.76

```
#Are these all from 'Charged Off' loans?
lcdf %>% select(loan_status, int_rate, funded_amnt, total_pymnt, annRet) %>% filter(annRet < 0) %>%
count(loan_status)
```

	loan_status	n
	<chr>	<int>
1	Charged Off	12146

```
#Calculate the annual return percentage
lcdf$last_pymnt_d<-paste(lcdf$last_pymnt_d, "-01", sep = "")
lcdf$last_pymnt_d<-parse_date_time(lcdf$last_pymnt_d,  "myd")
lcdf$actualTerm <- ifelse(lcdf$loan_status=="Fully Paid", as.duration(lcdf$issue_d  %--%
lcdf$last_pymnt_d)/dyears(1), 3)
lcdf$actualReturn <- ifelse(lcdf$actualTerm>0,
((lcdf$total_pymnt-lcdf$funded_amnt)/lcdf$funded_amnt)*(1/lcdf$actualTerm)*100, 0)
lcdf %>% select(loan_status, int_rate, funded_amnt, total_pymnt, annRet, actualTerm, actualReturn)
%>%  head()
```

	loan_status	int_rate	funded_amnt	total_pymnt	annRet	actualTerm	actualReturn
	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
1	Fully Paid	5.32	28000	29437.	1.71	1.17	4.39
2	Fully Paid	13.3	6150	7311.	6.29	1.92	9.84
3	Fully Paid	15.0	7200	8972.	8.20	3.00	8.20
4	Fully Paid	15.3	4750	5611.	6.04	1.42	12.8
5	Fully Paid	12.7	5000	6009.	6.73	2.59	7.80
6	Fully Paid	15.0	9600	11748.	7.46	2.16	10.3

There are a total of 13785 loans that have been “charged off”. According to the tables, there are some loans labelled “Charged Off” that provide positive return, although cumulatively the negative return is a lot bigger than the other one. We can conclude that a “Charged Off” loan can be categorized as a loss. For charged off loans, there are no actual pattern could be found when it comes to grade/sub-grade relation. Ideally, average interest rates and average return increases as loan grade decreases. But in the reality, it doesn’t work linearly. Subgrade D3 has the lowest annual return while F2 & C3 similarly has the highest annual return.

	loan_status	int_rate	funded_amnt	total_pymnt	annRet	actualTerm	actualReturn	sub_grade
1	Charged Off	24.08	3000	4230.92	13.676889	3	13.676889	F2
2	Charged Off	[illegible]	11500	16203.42	13.633101	3	13.633101	C3
3	Charged Off	22.15	6000	8237.45	12.430278	3	12.430278	E5
4	Charged Off	19.47	12000	16421.51	12.281972	3	12.281972	D5
5	Charged Off	19.72	14075	18986.57	11.631901	3	11.631901	D5
6	Charged Off	22.15	7000	9430.67	11.574619	3	11.574619	E5
7	Charged Off	23.10	5000	6709.73	11.398200	3	11.398200	E4
8	Charged Off	19.47	10275	13755.33	11.290608	3	11.290608	D5
9	Charged Off	23.43	3000	4003.20	11.146667	3	11.146667	F1
10	Charged Off	18.99	9925	13187.04	10.955634	3	10.955634	E1
11	Charged Off	23.10	10575	13936.94	10.597132	3	10.597132	E4

This table above amplifies the fact that even on the lower grade subgrade, there are some companies that could manage to capitalize the loan into positive return. Average Interest rate and average return values ideally increases linearly together while grade/subgrade becomes lower.

	grade	nLoans	defaults	avgInterest	stdInterest	avgLoanAMt	avgPmnt	avgRet	stdRet	minRet	maxRet
	<chr>	<int>	<int>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
1	A	22588	1187	7.17	0.967	14505.	15579.	2.39	3.94	-32.3	5.17
2	B	33907	3723	10.8	1.44	12637.	13779.	2.95	6.05	-32.5	7.90
3	C	26645	4738	13.8	1.19	12001.	13011.	2.83	8.14	-33.3	13.6
4	D	12493	2858	17.2	1.22	11894.	12871.	2.89	9.84	-33.3	12.3
5	E	3579	1010	19.9	1.38	11619.	12374.	2.56	11.3	-33.3	14.6
6	F	708	239	24.0	0.916	9272.	10050.	3.04	12.8	-32.1	15.2
7	G	80	30	26.4	0.849	11826.	12645.	1.24	14.1	-30.7	16.5

This table above provides the fact that average return doesn't increase ideally according to its grade/subgrade. The average return of grade F shows higher numbers than the other grades, while the interest rates are increasing linearly. This result is surprising since the average return of the grades doesn't have a measurable pattern. Here we could predict that there are some companies on the lower grade that performed above average compared to the other grades. Based on this data exploration, investing in companies with F grade (specifically F2) could be a good option. It may be a risky investment, but it has the probability to return above average. The other option we have is taking the less risky investment on grade A since it has a low interest rate compared to the other grade, and could give you a secure return.

Q.2(a)(v) What are people borrowing money for (purpose)? Examine how many loans, average amounts, etc. by purpose? Do loan amounts vary by purpose? Do defaults vary by purpose? Does loan-grade assigned by Lending Club vary by purpose?

R Code and Output

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

```
   purpose              n
   <chr>              <int>
 1 car                  928
 2 credit_card        24989
 3 debt_consolidation 57622
 4 home_improvement    5654
 5 house                354
 6 major_purchase      1823
 7 medical             1119
 8 moving               691
 9 other               5091
10 renewable_energy      58
11 small_business       893
12 vacation             678
13 wedding              100
```

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

```
>lcdf %>% group_by(purpose) %>% summarise(nLoans=n(), defaults=sum(loan_status=="Charged Off"), defaultRate=defaults/nLoans, avgIntRate=mean(int_rate), avgLoanAmt=mean(loan_amnt), avgActRet = mean(actualReturn), avgActTerm=mean(actualTerm))
```

```
   purpose            nLoans defaults defaultRate avgIntRate avgLoanAmt avgActRet avgActTerm

   <chr>              <int>    <int>     <dbl>      <dbl>      <dbl>     <dbl>      <dbl>

 1 car                  928      107     0.115       11.5      7955.      5.55       2.16

 2 credit_card        24989     2865     0.115       10.6     13660.      4.83       2.31

 3 debt_consolidation 57622     8319     0.144       12.2     13228.      5.28       2.22

 4 home_improvement    5654      682     0.121       11.8     11911.      5.42       2.22

 5 house                354       63     0.178       15.3     12757.      6.31       2.04

 6 major_purchase      1823      266     0.146       12.1      9948.      4.82       2.27

 7 medical             1119      172     0.154       14.3      7313.      6.38       2.22

 8 moving               691      144     0.208       16.1      6882.      5.90       2.22

 9 other               5091      838     0.165       14.7      8305.      6.40       2.28
```

10 renewable_energy	58	11	0.190	15.7	8807.	7.46	2.00
11 small_business	893	203	0.227	16.8	13603.	5.73	2.30
12 vacation	678	101	0.149	14.5	5674.	6.74	2.18
13 wedding	100	14	0.14	18.0	9124.	9.56	2.16

#Does loan-grade assigned by Lending Club vary by purpose?
>table(lcdf$purpose, lcdf$grade)

	A	B	C	D	E	F	G
car	253	306	238	92	27	8	4
credit_card	8349	9809	5008	1518	266	37	2
debt_consolidation	11573	19745	16497	7534	1954	292	27
home_improvement	1457	1777	1496	673	215	33	3
house	37	74	83	74	48	27	11
major_purchase	441	553	479	252	70	26	2
medical	84	270	382	251	97	34	1
moving	10	96	207	234	108	32	4
other	324	1036	1702	1321	551	139	18
renewable_energy	3	5	22	18	8	2	0
small_business	15	100	249	300	159	62	8
vacation	42	127	257	180	59	13	0
wedding	0	9	25	46	17	3	0

The purposes people are borrowing money for are: car, credit_card, debt_consolidation, home_improvement, house, major_purchase, medical, moving, other, renewable_energy, small_business,vacation, wedding. It can be seen that the major portion of loans are taken for the purpose of debt consolidation (57.6%) and credit card (24.9%) and lowest amount contribution is from renewable energy. The average Loan Amount varies by purpose (ranging from the lowest average of $5,674 for vacations to the highest average of $13,660 for small business). Across all purpose categories, the fewest number of loans are in loan grades E, F and G. Except in the credit card category, the majority of loans are in grades B, C and D (with a slightly less number of loans in grade A). The credit card category is the only purpose with the most amount of loans in loan grade A compared to the other grades. The debt consolidation category has the maximum number of defaults, which is also the greatest category for the purpose of loan. The percentage of defaults by purpose are from the small business category, with the loans charged off, which is the highest at 22.7%.

Q.2(a)(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.

For this part, we decided to look more into the following combinations of characteristics and attributes: employment length and loan status, employment length and loan grade, employment length and loan purpose, and employment length and home ownership status. Focusing on employment length gave us an insight on how this characteristic can have an impact on the attributes listed above. The third output below shows the "Charged Off" loans for each level of employment length, and as a result, the default rate is seen to be about steady within the 12-15% default rate range for the period, with a somewhat decreasing trend. The tables below speak for themselves and give a good view of the role the employment length factor plays, especially with loan purposes and home ownership status.

R Code and Output

```
#Converting emp_length to factor, arranged in ascending number of years
lcdf$emp_length <- factor(lcdf$emp_length, levels = c("< 1 year","1 year","2 years","3 years","4 years","5 years","6 years","7 years","8 years","9 years","10+ years","n/a"))
```

```
#Loans shown by employment length
lcdf %>% group_by(emp_length) %>% tally()
```

```
   emp_length     n
   <fct>      <int>
 1 < 1 year    8104
 2 1 year      6649
 3 2 years     8987
 4 3 years     8046
 5 4 years     5892
 6 5 years     6046
 7 6 years     4712
 8 7 years     5124
 9 8 years     4990
10 9 years     3908
11 10+ years  31394
12 n/a         6148
```

```
#Loans shown by employment length and loan status
table(lcdf$loan_status, lcdf$emp_length)
```

```
            < 1 year 1 year 2 years 3 years 4 years 5 years 6 years 7 years 8 years 9 years 10+ years   n/a
Charged Off     1204    960    1206    1088     775     841     632     712     698     522      3851  1296
Fully Paid      6900   5689    7781    6958    5117    5205    4080    4412    4292    3386     27543  4852
```

```
#Loans shown as a percentage of Charged Off loans for each level of employment length
cc = table(lcdf$loan_status, lcdf$emp_length)
(cc[1,]/(cc[1,] + cc[2,]))*100
```

```
 < 1 year    1 year   2 years   3 years   4 years   5 years   6 years   7 years   8 years   9 years 10+ years       n/a
14.85686  14.43826  13.41938  13.52225  13.15343  13.91002  13.41256  13.89539  13.98798  13.35722  12.26668  21.08003
```

```
#Loans shown by employment length and loan grade
table(lcdf$grade, lcdf$emp_length)
```

	< 1 year	1 year	2 years	3 years	4 years	5 years	6 years	7 years	8 years	9 years	10+ years	n/a
A	1786	1395	2030	1836	1332	1375	1020	1137	1165	888	7540	1084
B	2664	2229	2982	2715	1951	1978	1633	1768	1709	1351	11017	1910
C	2164	1846	2432	2142	1631	1622	1277	1336	1294	1021	8072	1808
D	1076	880	1112	995	736	801	591	662	610	466	3598	966
E	342	252	345	288	192	214	149	185	181	149	983	299
F	60	39	73	64	49	53	39	31	24	29	174	73
G	12	8	13	6	1	3	3	5	7	4	10	8

```
#Loans shown by employment length and loan purpose
table(lcdf$purpose, lcdf$emp_length)
```

	< 1 year	1 year	2 years	3 years	4 years	5 years	6 years	7 years	8 years	9 years	10+ years	n/a
car	104	67	90	85	52	70	42	42	50	26	245	55
credit_card	2260	1726	2323	2078	1485	1463	1237	1255	1213	962	7366	1621
debt_consolidation	4489	3838	5136	4588	3402	3500	2622	3015	2940	2331	18435	3326
home_improvement	302	285	432	386	314	365	265	275	288	215	2104	423
house	41	22	43	37	21	31	18	20	19	8	81	13
major_purchase	149	108	191	189	125	118	95	95	71	69	506	107
medical	87	72	109	92	53	63	46	52	59	38	374	74
moving	148	60	82	59	46	45	24	28	24	13	116	46
other	422	349	435	387	288	279	259	245	232	173	1619	403
renewable_energy	6	5	4	9	1	1	3	3	5	1	18	2
small_business	59	64	77	87	64	57	52	52	45	36	270	30
vacation	29	41	55	40	30	44	41	37	38	34	242	47
wedding	8	12	10	9	11	10	8	5	6	2	18	1

```
#Loans shown by employment length and home ownership status
table(lcdf$home_ownership, lcdf$emp_length)
```

	< 1 year	1 year	2 years	3 years	4 years	5 years	6 years	7 years	8 years	9 years	10+ years	n/a
MORTGAGE	2629	2296	3317	3193	2475	2698	2219	2471	2526	1985	18490	2689
OWN	672	576	814	775	561	587	470	473	467	366	3487	1189
RENT	4803	3777	4856	4078	2856	2761	2023	2180	1997	1557	9417	2270

Q.2(a)(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 analyses as in the questions above (as reasonable based on the derived variables).

1. The proportion of tot_hi_cred_lim compared to Loan status (fully paid or charged off). Tot_hi_cred_lim stands for total high credit limit. The higher the total credit limit of the borrower company, the higher also this company credit score. The simple formula for the attribute is the amount of the credit limits per open account

   The code use for this attribute is:
   lcdf$TotalHiCredLimits <- ifelse(lcdf$tot_hi_cred_lim>0, lcdf$tot_hi_cred_lim/lcdf$open_acc, 0)

2. The proportion of avg_cur_bal compared to Loan status (fully paid or charged off). Avg_cur_bal stands for average current balance of all accounts. This attribute emphasizes the relation of num_actv_bc_tl (number of currently active bankcard accounts) and the average current balance. This attribute could help the lender to estimate the current balance of the borrower

   The code use for this attribute is:
   lcdf$EstBalance <- ifelse(lcdf$Avg_cur_bal>0, lcdf$avg_cur_bal/lcdf$num_actv_bc_tl, 0)

3. The proportion of revol_bal compared to Loan status (fully paid or charged off). Revol_bal stands for Total credit revolving balance. This attribute is made by dividing the revol_bal with num_op_rev_tl (Number of open revolving accounts). The purpose of this attribute is to estimate how much the credit available for open revolving accounts owned by the borrower.

   The code use for this attribute is:
   lcdf$EstRevolvcredit <- ifelse(lcdf$revol_bal>0, lcdf$revol_bal/lcdf$ num_op_rev_tl, 0)

Q.2(c) Are there missing values? What is the proportion of missing values in different variables? Explain how you will handle missing values for different variables. You should consider what the variable is about, and what missing values may arise from – for example, a variable monthsSinceLastDeliquency 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?

R Code and Output

```
#How many variables are there in the data file?
dim(lcdf)

[1] 100000    145

#Drop variables with all empty values
lcdf <- lcdf %>% select_if(function(x){!all(is.na(x))})

#How many variables remain?
dim(lcdf)

[1] 100000    108

#Initially we had 145 Variables, after running the code we kept 108 variables

#Missing value proportions showing only those columns where there are missing values
colMeans(is.na(lcdf))[colMeans(is.na(lcdf))>0]
```

```
               emp_title                           title          mths_since_last_delinq          mths_since_last_record
                 0.06705                         0.00012                         0.49919                         0.82423
              revol_util                    last_pymnt_d              last_credit_pull_d     mths_since_last_major_derog
                 0.00041                         0.00064                         0.00004                         0.71995
             open_acc_6m                    open_act_il                     open_il_12m                     open_il_24m
                 0.97313                         0.97313                         0.97313                         0.97313
         mths_since_rcnt_il                    total_bal_il                         il_util                     open_rv_12m
                 0.97393                         0.97313                         0.97694                         0.97313
             open_rv_24m                      max_bal_bc                        all_util                          inq_fi
                 0.97313                         0.97313                         0.97313                         0.97313
             total_cu_tl                    inq_last_12m                     avg_cur_bal                  bc_open_to_buy
                 0.97313                         0.97313                         0.00002                         0.00964
                 bc_util             mo_sin_old_il_acct            mths_since_recent_bc        mths_since_recent_bc_dlq
                 0.01044                         0.03620                         0.00911                         0.74329
   mths_since_recent_inq mths_since_recent_revol_delinq                    num_rev_accts               num_tl_120dpd_2m
                 0.10612                         0.64746                         0.00001                         0.03824
            pct_tl_nvr_dlq                 percent_bc_gt_75                    hardship_dpd                settlement_term
                 0.00016                         0.01034                         0.99955                         0.99535
```

#Missing value percentages showing only those columns where there are missing values
colMeans(is.na(lcdf))[colMeans(is.na(lcdf))>0]*100

```
               emp_title                           title          mths_since_last_delinq          mths_since_last_record
                   6.705                           0.012                          49.919                          82.423
              revol_util                    last_pymnt_d              last_credit_pull_d     mths_since_last_major_derog
                   0.041                           0.064                           0.004                          71.995
             open_acc_6m                    open_act_il                     open_il_12m                     open_il_24m
                  97.313                          97.313                          97.313                          97.313
         mths_since_rcnt_il                    total_bal_il                         il_util                     open_rv_12m
                  97.393                          97.313                          97.694                          97.313
             open_rv_24m                      max_bal_bc                        all_util                          inq_fi
                  97.313                          97.313                          97.313                          97.313
             total_cu_tl                    inq_last_12m                     avg_cur_bal                  bc_open_to_buy
                  97.313                          97.313                           0.002                           0.964
                 bc_util             mo_sin_old_il_acct            mths_since_recent_bc        mths_since_recent_bc_dlq
                   1.044                           3.620                           0.911                          74.329
   mths_since_recent_inq mths_since_recent_revol_delinq                    num_rev_accts               num_tl_120dpd_2m
                  10.612                          64.746                           0.001                           3.824
            pct_tl_nvr_dlq                 percent_bc_gt_75                    hardship_dpd                settlement_term
                   0.016                           1.034                          99.955                          99.535
```

Q.3 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 will you exclude from the model.

#Drop some columns that would not useful and those that would cause a leakage
lcdf <- lcdf %>% 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, hardship_dpd, settlement_term, application_type))

#To drop other variables,
#varsToRemove <- c("last_pymnt_d", "last_pymnt_amnt","annRet")
#lcdf <- lcdf %>% select(-varsToRemove)

Q.4 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?

R Code and Output

```
#deploy aucAll variable considering both numeric and factor variable
aucAll<- sapply(lcdf %>% mutate_if(is.factor, as.numeric) %>% select_if(is.numeric), auc,
response=lcdf$loan_status)

#determine variable with auc > 0.5 and using tidy from broom package
aucAll[aucAll>0.5]
tidy(aucAll) %>% arrange(desc(aucAll))
tidy(aucAll[aucAll > 0.5]) %>% view()
```

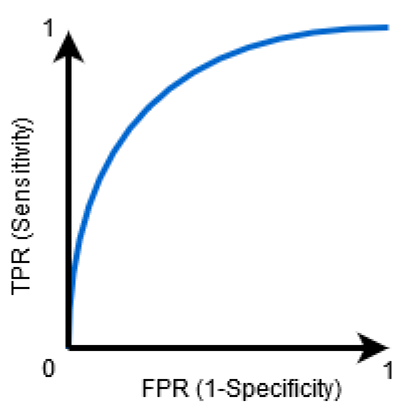
	names	x
1	loan_amnt	0.5211402
2	funded_amnt	0.5211402
3	funded_amnt_inv	0.5211474
4	int_rate	0.6581483
5	installment	0.5071865
6	annual_inc	0.5767804
7	dti	0.5682696
8	inq_last_6mths	0.5514872
9	mths_since_last_delinq	0.5024373
10	mths_since_last_record	0.5129245
11	open_acc	0.5079482
12	revol_bal	0.5367332
13	revol_util	0.5314844
14	total_acc	0.5184907
15	total_pymnt	0.7557938
16	total_pymnt_inv	0.7557987
17	total_rec_prncp	0.8285596
18	total_rec_int	0.5415626
19	recoveries	0.8784911
20	collection_recovery_fee	0.8599202
21	last_pymnt_amnt	0.7684163
22	tot_cur_bal	0.5611950
23	open_il_12m	0.5489335
24	mths_since_rcnt_il	0.5531434
25	total_bal_il	0.5073102
26	il_util	0.5486808
27	open_rv_24m	0.5997121

28	max_bal_bc	0.5473745
29	all_util	0.5582317
30	total_rev_hi_lim	0.5655743
31	inq_fi	0.5445042
32	total_cu_tl	0.5103094
33	inq_last_12m	0.5915851
34	acc_open_past_24mths	0.5825897
35	avg_cur_bal	0.5691553
36	bc_open_to_buy	0.5743476
37	bc_util	0.5435189
38	mo_sin_old_il_acct	0.5303673
39	mo_sin_old_rev_tl_op	0.5511155
40	mo_sin_rcnt_rev_tl_op	0.5538335
41	mo_sin_rcnt_tl	0.5596704
42	mort_acc	0.5583196

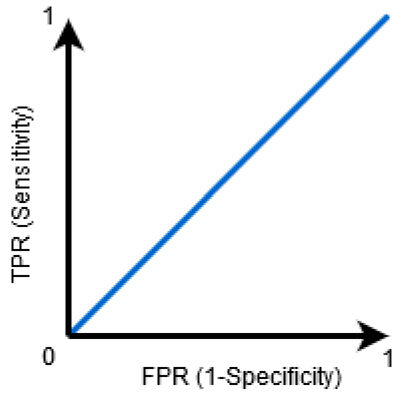
43	mths_since_recent_bc	0.5551020
44	mths_since_recent_bc_dlq	0.5055822
45	mths_since_recent_inq	0.5489350
46	num_bc_tl	0.5152625
47	num_il_tl	0.5099021
48	num_op_rev_tl	0.5176556
49	num_rev_accts	0.5078333
50	num_sats	0.5077449
51	pct_tl_nvr_dlq	0.5123979
52	tot_hi_cred_lim	0.5735512
53	total_bal_ex_mort	0.5169192
54	total_bc_limit	0.5730079
55	total_il_high_credit_limit	0.5116315

Univariate analysis is the simplest form of data analysis where the data being analyzed contains only one variable. Since it's a single variable it doesn't deal with causes or relationships. The main purpose of univariate analysis is to describe the data and find patterns that exist within it.

Here in this case, we use auc function to determine the response of dependent variable (loan_status) from other predictive variables. The table above shows the auc value of each variable considering the numeric and factor variables. Predictive variable recoveries show the highest auc value compared to other variables (x = 0.8784911). The area under the ROC curve (AUC) results were considered excellent for AUC values between 0.9-1, good for AUC values between 0.8-0.9, fair for AUC values between 0.7-0.8, poor for AUC values between 0.6-0.7 and failed for AUC values between 0.5-0.6. When AUC = 1, then the classifier is able to perfectly distinguish between all the Positive and the Negative class points correctly. If, however, the AUC had been 0, then the classifier would be predicting all Negatives as Positives, and all Positives as Negatives.



When 0.5<AUC<1, there is a high chance that the classifier will be able to distinguish the positive class values from the negative class values. This is so because the classifier is able to detect more numbers of True positives and True negatives than False negatives and False positives.



When AUC=0.5, then the classifier is not able to distinguish between Positive and Negative class points. Meaning either the classifier is predicting random class or constant class for all the data points. So, the higher the AUC value for a classifier, the better its ability to distinguish between positive and negative classes. If we ignore the data leakage issue, we can consider that the recoveries variable has the best value for predicting the loan amounts. While the recoveries variable has the highest auc value, the comprehensiveness of this data still needs to be asked. Hence, to determine the other suitable predictive variables, we have to consider the quality of the data itself. On the list there are some variables that could be used as a predictor such as total_pymnt (total payment), int_rate (interest rate), last_pymnt_amnt (last payment amount), and collection_recovery_fee.

Part B - Decision Tree Based Models and Performance Evaluation

Q.5(a) Split the data into training and validation sets. What proportions do you consider, why?

To build decision tree models for Lending club, The first thing we want to do is splitting the main dataset into training and validation dataset. Consider the size of the dataset we have, the training data should be the one with the highest portion. Training data set contains the data that we use to train the model. The purpose of the training dataset is to build a predictive model of the loans. Therefore, we put a 70% portion of the whole data to the training dataset. On the other hand, the purpose of the validation data set is to evaluate the performance of the model used. It can also be an indicator of overfitting and other defects during the training of the model. Therefore, we put a 30% portion of the data to the validation dataset. Ideally there should be a test dataset in the model created but considering the complexity of lending club data we can use validation proportion as a representation of model evaluator.

Q.5(b) Train decision tree models (use both rpart, c50). [If something looks too good, it may be due to leakage – make sure you address this] 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. How do you evaluate performance – which measure do you consider, and why?

After deploying the datasets into the training and validation set, we use rpart and c50 tree models to train the data. The implementation of the rpart tree model begins with creating a weighted tree for the training set. The purpose of generating the tree is to classify the data and handle the heterogeneous data from the main dataset. The model resulted in a high accuracy (84% according to confusionmatrix), 95% specificity and overfit. To simplify the size of the tree, we prune the weighted tree and add a complexity parameter beginning with 0.002. It results in 84% accuracy and specificity of 94%. It shows a little reduction in accuracy and specificity, but it also reduces the complexity of the main tree. Following the training set, we make a cross validation of the running data. With the same parameters, cross validation prediction results in 83% accuracy and 94% specificity. Compared to the previous result, there's a slight reduction in accuracy. We can imply from these results that the accuracy of the validation is fairly high which means there's a good chance of overfit on this model. At the final part of the evaluation, we use a decile chart to examine the response of the data by each category. Here, we use "bucket" to distinguish every group of data. The model exhibiting a good staircase decile analysis is one we can consider moving forward with and the results show that.

R Code and Output

#Create weighted tree for training set
myweights = ifelse(Trainingdf$loan_status == "Charged Off", 3, 1)
Wghtd_lcDT <- rpart(loan_status ~., data=Trainingdf, method="class", weights = myweights,
parms = list(split = "information"), control = rpart.control(cp=0.001))

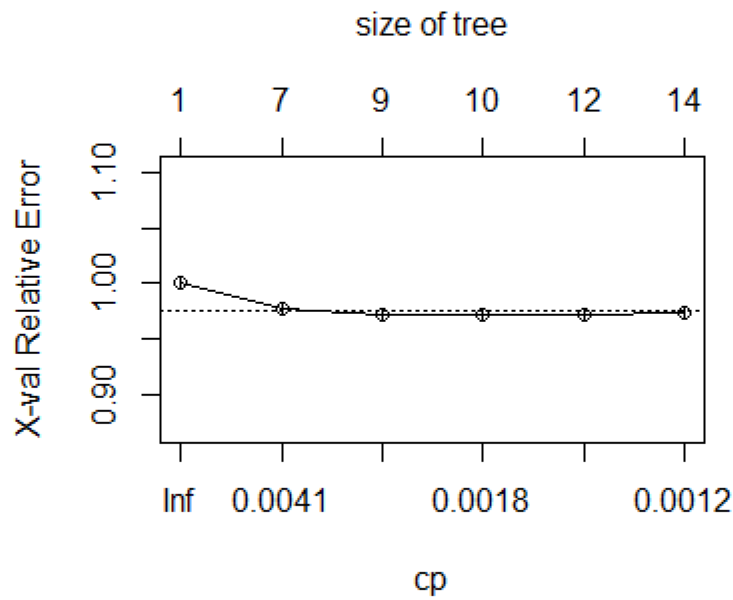
pred_wghtTrn=predict(Wghtd_lcDT,Trainingdf, type='class')

#Create Confusion table
confusionMatrix(table(predWghtTrain = pred_wghtTrn, true=Trainingdf$loan_status))

```
> pred_wghtTrn=predict(Wghtd_lcDT,Trainingdf, type='class')
>
> #Create Confusion table
> confusionMatrix(table(predWghtTrain = pred_wghtTrn, true=Trainingdf$loan_status))
Confusion Matrix and Statistics

             true
predWghtTrain Charged Off Fully Paid
  Charged Off        1489       2940
  Fully Paid         8156      57415

               Accuracy : 0.841
                 95% CI : (0.839, 0.844)
    No Information Rate : 0.862
    P-Value [Acc > NIR] : 1

                  Kappa : 0.137

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.1544
            Specificity : 0.9513
         Pos Pred Value : 0.3362
         Neg Pred Value : 0.8756
             Prevalence : 0.1378
         Detection Rate : 0.0213
   Detection Prevalence : 0.0633
      Balanced Accuracy : 0.5528

       'Positive' Class : Charged Off
```

##Details About the Training Set
#Print performance and tree size for different complexity parameter values
printcp(Wghtd_lcDT)

#Plot the weighted decision tree
plotcp(Wghtd_lcDT)




```
#Variable importance as given by a decision tree model
Wghtd_lcDT$variable.importance

#Prune Tree based on cp
prune_lcDT <- prune(Wghtd_lcDT, cp=0.002)

#Evaluate performance base on validation dataset
predVal=predict(prune_lcDT,Validationdf, type='class')
table(predictValidation = predVal, true=Validationdf$loan_status)
mean(predVal == Validationdf$loan_status)

#Deploy the Confusion table
confusionMatrix(table(predictValidation = predVal, true=Validationdf$loan_status))
```

```
> table(predictValidation = predVal, true=Validationdf$loan_status)
                 true
predictValidation Charged Off Fully Paid
      Charged Off         655       1652
      Fully Paid         3485      24208
> mean(predVal == Validationdf$loan_status)
[1] 0.83
>
> #Deploy the confusion table
> confusionMatrix(table(predictValidation = predVal, true=Validationdf$loan_status))
Confusion Matrix and Statistics

                 true
predictValidation Charged Off Fully Paid
      Charged Off         655       1652
      Fully Paid         3485      24208

               Accuracy : 0.829
                 95% CI : (0.824, 0.833)
    No Information Rate : 0.862
    P-Value [Acc > NIR] : 1

                  Kappa : 0.116

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.1582
            Specificity : 0.9361
         Pos Pred Value : 0.2839
         Neg Pred Value : 0.8742
             Prevalence : 0.1380
         Detection Rate : 0.0218
   Detection Prevalence : 0.0769
      Balanced Accuracy : 0.5472

       'Positive' Class : Charged Off
```

##Lifts for Weighted Rpart tree
# 'scores' from applying the model to the data
predTrnProb=predict(prune_lcDT, Trainingdf, type='prob')
head(predTrnProb)

#Create a data-frame with only the model scores and the actual class
trainScore <- Trainingdf %>% select("loan_status")
trainScore$score<-predTrnProb[, 1]

#View on trainScore dataframe
head(trainScore)

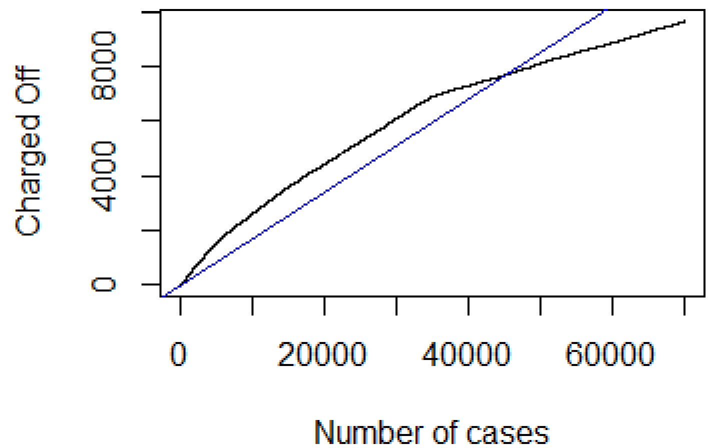
#Sort by score variables
trainScore<-trainScore[order(trainScore$score, decreasing=TRUE),]

#Determine the cumulative summary of "default" outcome values
trainScore$cumDefault<-cumsum(trainScore$loan_status == "Charged Off")

#First 10 row in trainScore

trainScore[1:10,]

#Plot the cumDefault values (y-axis) by numCases (x-axis)
plot(trainScore$cumDefault, type = "l", xlab='Number of cases', ylab='Charged Off')
abline(0,max(trainScore$cumDefault)/56714, col="blue") #diagonal line

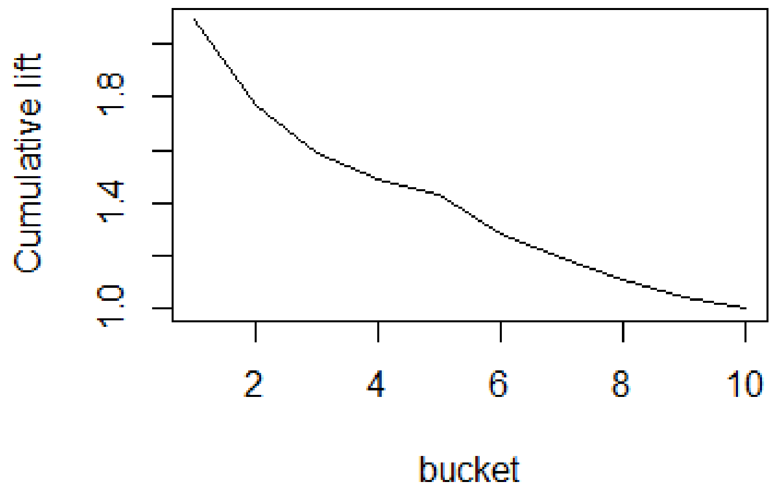


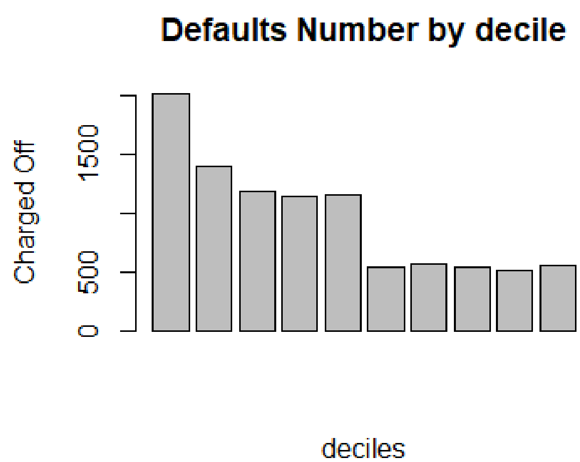
##Creating decile lift table to measure the predictive model of Rpart
#Divide the data into 10 for decile lift equal groups
trainScore["bucket"]<- ntile(-trainScore[,"score"], 10)

#Group the data by the 'buckets', and obtain summary statistics
Liftsdata <- trainScore %>% group_by(bucket) %>% summarize(count=n(),
numDefaults=sum(loan_status=="Charged Off"),
defRate=numDefaults/count,
cumDefRate=cumsum(numDefaults)/cumsum(count),
lift = cumDefRate/(sum(trainScore$loan_status=="Charged
Off")/nrow(trainScore)))

#View on the lifts data table
view(Liftsdata)

#Plot various type of chart (barplot and cummulative lift),
plot(Liftsdata$bucket, Liftsdata$lift, xlab="deciles", ylab="Cumulative Decile Lift", type="l")
plotLift(trainScore$score, trainScore$loan_status == "Charged Off")
barplot(Liftsdata$numDefaults, main="Defaults Number by decile", xlab="deciles", ylab="Charged Off")





Following the rpart model, we also implemented the c50 tree model to gain a prediction based on loan status. We used the same method and process with rpart, by separating the datasets and validating the results with cross validation and decile chart. The results according to the confusion matrix are 82% for accuracy and 89% for specificity. On the other hand, cross validation results show 79% for accuracy and 87% for specificity. The decile chart shows a staircase pattern which means the model works well for the data.

```
#Build a decision tree model with C50 function
c5_dtm <- C5.0(loan_status ~ ., data=Trainingdf, control=C5.0Control(minCases=50), weights =
myweights)

#Set a Prediction for Training, and validation dataset
predTrn_c5dtm <- predict(c5_dtm, Trainingdf, type='class')
predVal_c5dtm <- predict(c5_dtm, Validationdf, type='class')
```

#Determine the mean of predictive variable Training & validation
mean(predTrn_c5dtm==Trainingdf$loan_status)
mean(predVal_c5dtm==Validationdf$loan_status)

##Predictions for Training
predTrn_c5dtm <- predict(c5_dtm, Trainingdf, type='class')
confusionMatrix(table(predictC50Train = predTrn_c5dtm, true=Trainingdf$loan_status))

```
> predTrn_c5dtm <- predict(c5_dtm, Trainingdf, type='class')
> confusionMatrix(table(predictC50Train = predTrn_c5dtm, true=Trainingdf$loan_status))
Confusion Matrix and Statistics

               true
predictC50Train Charged Off Fully Paid
    Charged Off        3868       6943
    Fully Paid         5777      53412

               Accuracy : 0.818
                 95% CI : (0.815, 0.821)
    No Information Rate : 0.862
    P-Value [Acc > NIR] : 1

                  Kappa : 0.272

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.4010
            Specificity : 0.8850
         Pos Pred Value : 0.3578
         Neg Pred Value : 0.9024
             Prevalence : 0.1378
         Detection Rate : 0.0553
   Detection Prevalence : 0.1544
      Balanced Accuracy : 0.6430

       'Positive' Class : Charged Off
```

##Predictions for Validation
predVal_c5dtm <- predict(c5_dtm, Validationdf, type='class')
confusionMatrix(table(predictC50Validation = predVal_c5dtm, true=Validationdf$loan_status))

```
> confusionMatrix(table(predictC50Validation = predVal_c5dtm, true=Validationdf$loan_status))
Confusion Matrix and Statistics

                    true
predictC50Validation Charged Off Fully Paid
         Charged Off        1115       3422
         Fully Paid         3025      22438

               Accuracy : 0.785
                 95% CI : (0.78, 0.79)
    No Information Rate : 0.862
    P-Value [Acc > NIR] : 1

                  Kappa : 0.132

 Mcnemar's Test P-Value : 8.14e-07

            Sensitivity : 0.2693
            Specificity : 0.8677
         Pos Pred Value : 0.2458
         Neg Pred Value : 0.8812
             Prevalence : 0.1380
         Detection Rate : 0.0372
   Detection Prevalence : 0.1512
      Balanced Accuracy : 0.5685

       'Positive' Class : Charged Off
```

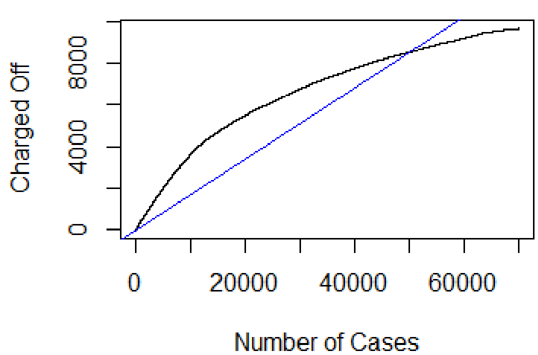
##Lifts for Weighted Rpart tree
#Acquire the scores of the model applied
c5predTrnScr=predict(c5_dtm, Trainingdf, type='prob')

#Selects the OUTCOME column into trainingScr
trainingScr <- Trainingdf %>% select("loan_status")
trainingScr$score<-c5predTrnScr[, 1]

#Sort by the highest score
trainingScr<-trainingScr[order(trainingScr$score, decreasing=TRUE),]

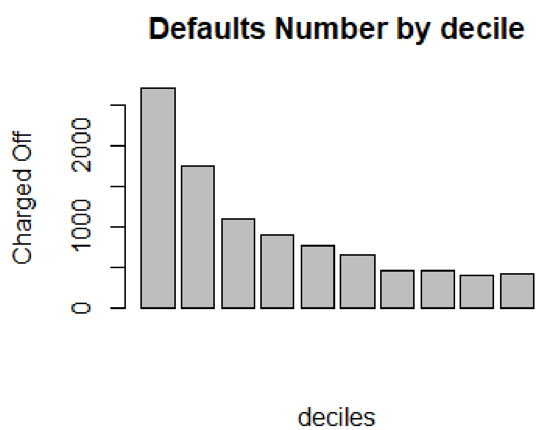
#Generate the cumulative sum of "default" OUTCOME values
trainingScr$cumDefault<-cumsum(trainingScr$loan_status == "Charged Off")

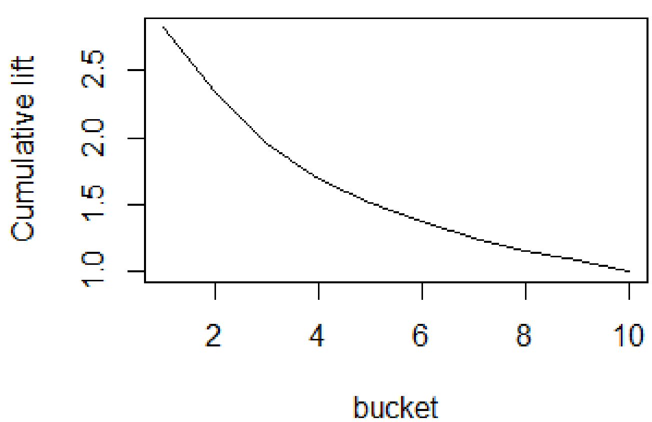
#Plot the cumDefault values (y-axis) by numCases (x-axis)
plot(trainingScr$cumDefault, type = "l", xlab='Number of Cases', ylab='Charged Off')
abline(0,max(trainingScr$cumDefault)/56714, col="blue")



```
##Creating decile lift table to measure the predictive model of C50
#Divide the data into 10 (for decile lift) equal groups
trainingScr["bucket"]<- ntile(-trainingScr[,"score"], 10)

#Group the data by the 'buckets', and obtain summary statistics
c5Liftsdata <- trainingScr %>% group_by(bucket) %>% summarize(count=n(),
numDefaults=sum(loan_status=="Charged Off"),
                                        defRate=numDefaults/count,
cumDefRate=cumsum(numDefaults)/cumsum(count),
                                        lift = cumDefRate/(sum(trainingScr$loan_status=="Charged
Off")/nrow(trainingScr)) )

#View at the table
c5Liftsdata

#Various plots can be done, for example
plot(c5Liftsdata$bucket, c5Liftsdata$lift, xlab="deciles", ylab="Cumulative Decile Lift", type="l")
plotLift(trainingScr$score, trainingScr$loan_status == "Charged Off")
barplot(c5Liftsdata$numDefaults, main="numDefaults by decile", xlab="deciles", ylab ="Charged Off")
```

Results





Q.5(c) 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 uni-variate analyses in question 4 above? Briefly describe how variable importance is obtained (the process used in decision trees).

According to the results, the rpart model is a better model compared to c50. Accuracy and specificity wise, the rpart model had a higher score at 84% and 95%, respectively. This method of analysis has a similarity with univariate analyses, which is making a classification for several data, and making a predictive model from it. Univariate analysis works by examining the effects of a single variable (loan status) on a dataset and calculates a score of response from the other variable, while decision tree model also examines the same thing and calculates a response but in different ways (by splitting the data into training, validation, and test set).

Q.6 Develop a random forest model. (Note the 'ranger' library can give faster computations) What parameters do you experiment with, and does this affect performance? Describe the best model in terms of number of trees, performance, variable importance. Compare the performance of random forest and best decision tree model from Q5 above. Do you find the importance of variables to be different? Which model would you prefer, and why?

On developing this model, we use min.node.size function to set the depth of the trees that we'll make. Since the purpose of our model is to make a classification of the data, we set the value of the node size to default value (1). The amount of data divided by loan status shows an imbalanced proportion between "Fully charged" and "Charged Off". Therefore, we set the weight ratio of the training data to 5:1. After running this model, we obtain an accuracy of 87% for the training set and 86% for the validation data set. It shows that the random model has a better accuracy than the previous model (rpart and C5.0) which means it works a bit better for classification purposes. In our opinion, every decision tree model has its own advantages according to the purposes. There are several things to consider when choosing the best model (i.e. complexity of the data, the target, etc). In this case, the random forest model is a better option from the previous model in terms of accuracy, but it may be differ when being used for other purposes.

```
##Build a Random Forest model based on min.node.size=1 parameters.
#The purpose is to made a model for classification
library(ranger)
myweights = ifelse(Trainingdf$loan_status == "Charged Off", 5, 1)
RFmodel <- ranger(loan_status ~., data=Trainingdf, num.trees =200, min.node.size=1,
            importance='impurity', case.weights= myweights)

#Confusion matrix for RFmodel
RFmodel[["confusion.matrix"]]
```

```
> #Confusion matrix for RFmodel
> RFmodel[["confusion.matrix"]]
              predicted
true            Charged Off Fully Paid
  Charged Off          1620       8023
  Fully Paid            990      59365
```

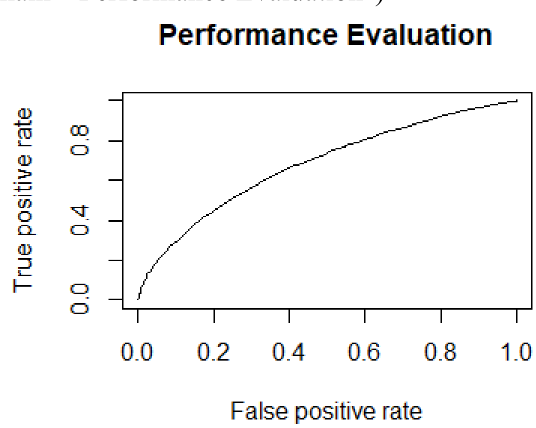
#Generate score for Validation dataset
scoresRFVal <- predict(RFmodel, Validationdf)
#Confusion table validation
table(scoresRFVal$predictions,Validationdf$loan_status)

```
> #confusion table validation
> table(scoresRFVal$predictions,Validationdf$loan_status)

              Charged Off Fully Paid
  Charged Off         178        385
  Fully Paid         3962      25475
```

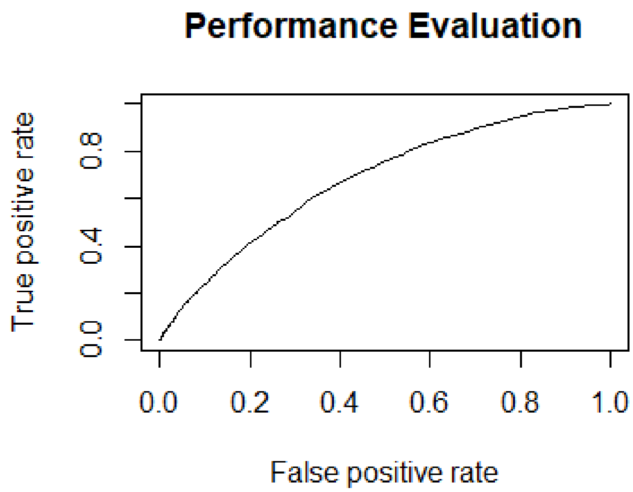
##Predictions for Validation
predVal_RFmodel <- predict(RFmodel, Validationdf, type='response')
predVal_RFmodel

##ROC Testing
#Perform ROC to review the quality of the model
rgModelROC <- ranger(loan_status ~., data=Trainingdf, num.trees =200, min.node.size=1, importance='impurity', case.weights= myweights, probability = TRUE)
scoresRFVal1 <- predict(rgModelROC, Validationdf, type="response")

#Apply the ROCR function to get a prediction object for "charged off"
rocPredVal <- prediction(scoresRFVal1 [["predictions"]][,2], Validationdf$loan_status, label.ordering = c('Charged Off','Fully Paid'))
#Plot the performance
performanceROCVal <- performance(rocPredVal, "tpr", "fpr")
plot(performanceROCVal, main="Performance Evaluation")



```
#Now apply the prediction function from ROCR to get a prediction object for fully paid
rocPredVal <- prediction(scoresRFVal1 [["predictions"]][,1], Validationdf$loan_status, label.ordering =
c('Fully Paid','Charged Off'))

#Plot the performance
performanceROCVal1 <- performance(rocPredVal, "tpr", "fpr")
plot(performanceROCVal1, main = "Performance Evaluation")
```



Q.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. Evaluate 5 different thresholds and analyze performance. Which model do you think will be best, and why.

R Code and Output

```
#Loan Analysis
#Before starting the analysis, we made a copy of the original datasets to match the variable needed,
#Since the original datasets has been modified for other case
lcdfnew <- read_csv('lcData100K.csv')

#Remove loans with a status other than charged off and Fully Paid
lcdfnew <- lcdfnew %>% filter(loan_status == "Fully Paid" | loan_status == "Charged Off")

#Changing emp_length to factor
lcdfnew$emp_length <- factor(lcdfnew$emp_length, levels=c("n/a", "< 1 year","1 year","2 years",
                    "3 years",  "4 years",   "5 years",   "6 years",  "7 years" ,  "8 years", "9 years", "10+ years"
))

#Regrouping purpose
lcdfnew$purpose <- fct_recode(lcdfnew$purpose, other="wedding", other="renewable_energy")

#Filtering home ownership
```

```
lcdfnew <- lcdfnew %>% filter(home_ownership == "MORTGAGE"
                    | home_ownership == "OWN"
                    | home_ownership == "RENT")

lcdfnew <- lcdfnew %>% mutate_if(is.character, as.factor)
lcdfnew <- lcdfnew %>% mutate(loan_status=as.factor(loan_status))

##Begin analyzing the loans
#Loans group by grade
lcdfnew %>% 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))
```

	grade	nLoans	defaults	avgInterest	stdInterest	avgLoanAMt	avgPmnt
	<fct>	<int>	<int>	<dbl>	<dbl>	<dbl>	<dbl>
1	A	22588	1187	7.17	0.967	14505.	15579.
2	B	33907	3723	10.8	1.44	12637.	13779.
3	C	26645	4738	13.8	1.19	12001.	13011.
4	D	12493	2858	17.2	1.22	11894.	12871.
5	E	3579	1010	19.9	1.38	11619.	12374.
6	F	708	239	24.0	0.916	9272.	10050.
7	G	80	30	26.4	0.849	11826.	12645.

```
#Define & calculate the annualized percentage return
lcdfnew$annRet <- ((lcdfnew$total_pymnt -lcdfnew$funded_amnt)/lcdfnew$funded_amnt)*(12/36)*100

#Summarize by grade
lcdfnew %>% 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))
```

	grade	nLoans	defaults	avgInterest	stdInterest	avgLoanAMt	avgPmnt	avgRet	stdRet	minRet	maxRet
	<fct>	<int>	<int>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
1	A	22588	1187	7.17	0.967	14505.	15579.	2.39	3.94	-32.3	5.17
2	B	33907	3723	10.8	1.44	12637.	13779.	2.95	6.05	-32.5	7.90
3	C	26645	4738	13.8	1.19	12001.	13011.	2.83	8.14	-33.3	13.6
4	D	12493	2858	17.2	1.22	11894.	12871.	2.89	9.84	-33.3	12.3
5	E	3579	1010	19.9	1.38	11619.	12374.	2.56	11.3	-33.3	14.6
6	F	708	239	24.0	0.916	9272.	10050.	3.04	12.8	-32.1	15.2
7	G	80	30	26.4	0.849	11826.	12645.	1.24	14.1	-30.7	16.5

```
#Find out the actual loan term in months, to track loans that returned early
lcdfnew$last_pymnt_d<-paste(lcdfnew$last_pymnt_d, "-01", sep = "")
lcdfnew$last_pymnt_d<-parse_date_time(lcdfnew$last_pymnt_d, "mYd")

#Define actual term
```

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

#Then, considering this actual term, the actual annual return is
lcdfnew$actualReturn <- ifelse(lcdfnew$actualTerm>0, ((lcdfnew$total_pymnt -
lcdfnew$funded_amnt)/lcdfnew$funded_amnt)*(1/lcdfnew$actualTerm), 0)

#loan performance by grade
lcdfnew %>% group_by(grade) %>% summarise(nLoans=n(), defaults=sum(loan_status=="Charged
Off"), defaultRate=defaults/nLoans,
                              avgInterest= mean(int_rate), avgLoanAmt=mean(loan_amnt),
avgRet=mean(annRet), avgActualRet=mean(actualReturn)*100,
                              avgActualTerm=mean(actualTerm), minActualRet=min(actualReturn)*100,
maxActualRet=max(actualReturn)*100)
```

	grade	nLoans	defaults	defaultRate	avgInterest	avgLoanAmt	avgRet	avgActualRet	avgActualTerm	minActualRet	maxActualRet
	<fct>	<int>	<int>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
1	A	22588	1187	0.0526	7.17	14505.	2.39	3.86	2.24	-32.3	16.4
2	B	33907	3723	0.110	10.8	12637.	2.95	5.18	2.24	-32.5	30.9
3	C	26645	4738	0.178	13.8	12001.	2.83	5.73	2.25	-33.3	34.8
4	D	12493	2858	0.229	17.2	11894.	2.89	6.53	2.25	-33.3	33.6
5	E	3579	1010	0.282	19.9	11619.	2.56	6.64	2.29	-33.3	40.8
6	F	708	239	0.338	24.0	9272.	3.04	7.25	2.40	-32.1	40.2
7	G	80	30	0.375	26.4	11826.	1.24	5.85	2.32	-30.7	44.4

```
#Summarize loan performance by grade and loan status
lcdfnew %>% group_by(grade, loan_status) %>% summarise(nLoans=n(),
defaults=sum(loan_status=="Charged Off"), defaultRate=defaults/nLoans,
                                   avgInterest= mean(int_rate), avgLoanAmt=mean(loan_amnt),
avgRet=mean(annRet), avgActualRet=mean(actualReturn),
                                   avgActualTerm=mean(actualTerm), minActualRet=min(actualReturn),
maxActualRet=max(actualReturn))
```

	grade	loan_status	nLoans	defaults	defaultRate	avgInterest	avgLoanAmt	avgRet	avgActualRet	avgActualTerm	minActualRet
	<fct>	<fct>	<int>	<int>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
1	A	Charged Off	1187	1187	1	7.44	13633.	-11.6	-0.116	3	-0.323
2	A	Fully Paid	21401	0	0	7.16	14554.	3.17	0.0472	2.20	0
3	B	Charged Off	3723	3723	1	10.9	12415.	-11.5	-0.115	3	-0.325
4	B	Fully Paid	30184	0	0	10.7	12665.	4.73	0.0725	2.15	0
5	C	Charged Off	4738	4738	1	13.9	12076.	-12.0	-0.120	3	-0.333
6	C	Fully Paid	21907	0	0	13.8	11985.	6.03	0.0956	2.09	0
7	D	Charged Off	2858	2858	1	17.2	12113.	-12.4	-0.124	3	-0.333
8	D	Fully Paid	9635	0	0	17.2	11829.	7.42	0.121	2.03	0
9	E	Charged Off	1010	1010	1	19.8	12012.	-12.7	-0.127	3	-0.333
10	E	Fully Paid	2569	0	0	20.0	11464.	8.56	0.142	2.01	0
11	F	Charged Off	239	239	1	24.1	9809.	-11.9	-0.119	3	-0.321
12	F	Fully Paid	469	0	0	23.9	8999.	10.6	0.170	2.09	0.0662
13	G	Charged Off	30	30	1	26.4	10028.	-14.4	-0.144	3	-0.307
14	G	Fully Paid	50	0	0	26.4	12905	10.6	0.180	1.91	0

… with 1 more variable: maxActualRet <dbl>

#ProfitValue
lcdfnew %>% group_by(loan_status) %>% summarise(avgInt=mean(int_rate),avgActInt = mean(actualReturn))

```
loan_status avgInt avgActInt
<fct>        <dbl>     <dbl>
Charged Off   13.9   -0.120
Fully Paid    11.7    0.0802
```

Q.7(b) Another approach is to directly consider how the model will be used – you can order the loans in descending order of prob(fully-paid). Then, you can consider starting with the loans which are most likely to be fully-paid and go down this list till the point where overall profits begin to decline (as discussed in class). Conduct an analysis to determine what threshold/cutoff value of prob(fully-paid) you will use and what is the total profit from different models. Also compare the total profits from using a model to that from investing in the safe CDs. Explain your analyses and calculations. Which model do you find to be best and why. And how does this compare with what you found to be best in part (a) above.

We decided to settle for the cutoff score to be at 0.556, where the default rate was at 21%. With the scores beyond this point dropping below 0.5, and the default rate crossing 25%, we felt the risk would be greater. When comparing investing in safe CDs versus a model, risk plays a big factor. CDs are insured and are considered a safe investment.

R Code and Output

```
#Get the 'scores' from applying the model to the data
predTrnProb2=predict(prune_lcDT, Trainingdf, type='prob')

trnSc2 <- Trainingdf %>%  select("loan_status")
trnSc2$score<-predTrnProb2[, 2]

#sort by score
trnSc2<-trnSc2[order(trnSc2$score, decreasing=TRUE),]

trnSc2[1:50,]
```

```
   loan_status score
   <fct>       <dbl>
 1 Fully Paid  0.797
 2 Fully Paid  0.797
 3 Fully Paid  0.797
 4 Fully Paid  0.797
 5 Fully Paid  0.797
 6 Fully Paid  0.797
 7 Fully Paid  0.797
 8 Fully Paid  0.797
 9 Fully Paid  0.797
10 Fully Paid  0.797
# … with 40 more rows
```

trnSc2 %>% group_by(score, loan_status) %>% summarise(nloans = n())

```
   score loan_status nloans
   <dbl> <fct>        <int>
 1 0.388 Charged Off    550
 2 0.388 Fully Paid    1045
 3 0.438 Charged Off    770
 4 0.438 Fully Paid    1802
 5 0.459 Charged Off    383
 6 0.459 Fully Paid     976
 7 0.556 Charged Off    259
 8 0.556 Fully Paid     972
 9 0.569 Charged Off    457
10 0.569 Fully Paid    1811
11 0.571 Charged Off   1065
12 0.571 Fully Paid    4248
13 0.627 Charged Off   3364
14 0.627 Fully Paid   16932
15 0.663 Charged Off     54
16 0.663 Fully Paid     319
17 0.797 Charged Off   2743
18 0.797 Fully Paid   32250
```

trnSc2 %>% group_by(score) %>% summarise(nLoans=n(), defaults=sum(loan_status=="Charged Off")) %>% mutate(prctCharged_off=defaults/nLoans*100)

```
  score nLoans defaults prctCharged_off
  <dbl>  <int>    <int>           <dbl>
1 0.388   1595      550           34.5
2 0.438   2572      770           29.9
3 0.459   1359      383           28.2
4 0.556   1231      259           21.0
5 0.569   2268      457           20.1
6 0.571   5313     1065           20.0
7 0.627  20296     3364           16.6
8 0.663    373       54           14.5
9 0.797  34993     2743            7.84
```

Citations

Kagan, J. (2021, May 19). Peer-to-peer lending. Investopedia. Retrieved September 23, 2021, from https://www.investopedia.com/terms/p/peer-to-peer-lending.asp.

Beachey, A. (2021, July 22). Advantages and disadvantages of peer to peer lending. NerdWallet. Retrieved September 23, 2021, from https://www.nerdwallet.com/uk/loans/what-is-peer-to-peer-lending/#advantages-for-the-lender-investor.

Pritchard, J. (2021, June 3). Borrowing With Peer-to-Peer Loans: How It Works. The Balance. Retrieved September 23, 2021, from https://www.thebalance.com/how-peer-to-peer-loans-work-315730.

MarketWatch. (2021, July 1). Peer to Peer (P2P) Lending Market 2021-2026: Size, Outlook, Share, Industry Trends, Key Players, and Forecast. MarketWatch. Retrieved September 23, 2021, from https://www.marketwatch.com/press-release/peer-to-peer-p2p-lending-market-2021-2026-size-outlook-share-industry-trends-key-players-and-forecast-2021-07-01?tesla=y.

AUC-ROC curve in machine learning clearly explained. Analytics Vidhya. (2020, July 20). Retrieved September 26, 2021, from https://www.analyticsvidhya.com/blog/2020/06/auc-roc-curve-machine-learning/.

Describe the difference between univariate, bivariate and multivariate analysis? (n.d.). Retrieved September 26, 2021, from https://www.modernanalyst.com/Careers/InterviewQuestions/tabid/128/ID/4904/Describe-the-difference-between-univariate-bivariate-and-multivariate-analysis.aspx.

Basic principles of ROC analysis. Metz CE Semin Nucl Med. 1978 Oct; 8(4):283-298