PART A

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)?***

Ans:

Objective: Our objective is to determine the potential of borrowers to default on a loan. This information would help the lenders avoid losses by reducing write-off cases.
'Better' and 'worse' will be evaluated on the return rate relative to the default risk. We want to find the best return with the lowest risk of default.
Goal: Predictive modeling will allow us to assess new loans to determine their level of risk and possible returns.
Target Variable: The potential target variable will be "loan_status".

(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 maybe some of the important attributes to consider for your decision task?

Ans:
Attributes Categories: You can broadly categorize the attributes as:
Borrower Information:

- annual_inc
- emp_length
- home_ownership
- fico_range_low, fico_range_high
- addr_state
- zip_code

Loan Details:

- loan_amnt
- funded_amnt
- funded_amnt_inv
- term
- int_rate
- installment
- grade, sub_grade
- purpose
- title
- application_type
- disbursement_method
- loan_status

Loan Performance:

- loan_status
- total_pymnt, total_rec_prncp, total_rec_int
- recoveries
- collection_recovery_fee
- last_pymnt_d, last_pymnt_amnt
- next_pymnt_d
- issue_d

- earliest_cr_line
- hardship_flag
- debt_settlement_flag

Platform Decisions:

- loan_grade, sub_grade
- policy_code
- verification_status
- hardship_flag, hardship_type, hardship_reason
- debt_settlement_flag, settlement_status

2. Data exploration – examine the data for basic insights.

(a) (i) What is the proportion of defaults ('charged off' vs 'fully paid' loans) in the data?

loan_status <chr>	n <int>
Charged Off	13783
Current	18
Fully Paid	86172
In Grace Period	3
Late (16-30 days)	1
Late (31-120 days)	23

There are 13783 "charged off" loans and 86172 "fully paid" loans.

```
Charged Off  Fully Paid
  0.1378921   0.8621079
```

86% of the loans are "Fully paid", and 13.7% of the loans are "Charged off".

How does the default rate vary with loan grade? Does it vary with sub-grade? Do you think loan grade and sub-grade convey useful information on riskiness of different loans?

A tibble: 7 × 2

grade <chr>	total_loans <int>
A	22605
B	34468
C	26266
D	12346
E	3536
F	705
G	74

	A	B	C	D	E	F	G
Charged Off	1210	3756	4707	2839	1013	227	31
Current	2	7	5	3	0	0	1
Fully Paid	21392	30701	21539	9498	2523	477	42
In Grace Period	0	0	2	1	0	0	0
Late (16-30 days)	0	0	1	0	0	0	0
Late (31-120 days)	1	4	12	5	0	1	0

Default rate against loan grade –
As the loan grade moves from A to G, the number of "Charged Off" loans increases relative to the total number of loans.

Charged-Off:
Grade A: 1210/22605 = 5.35%
Grade B: 3756/34468= 10.9%
Grade C: 4707/26266= 17.9%
Grade D: 2839/12346= 23%
Grade E: 1013/3536= 28.65%
Grade F: 227/705= 32.20%
Grade G: 31/74= 41.89%

The proportion of "**Fully Paid**" loans decreases as the loan grade goes from A to G. This indicates that higher grades (A, B) are more likely to result in fully paid loans, whereas lower grades (E, F, G) have a lower proportion of loans that are fully paid.

Default rate against Sub-grade

	sub_grade	loan_status	count
1	A1	Charged Off	102
2	A1	Fully Paid	3512
3	A2	Charged Off	125
4	A2	Fully Paid	3330
5	A3	Charged Off	183
6	A3	Current	1
7	A3	Fully Paid	3485
8	A4	Charged Off	333
9	A4	Fully Paid	5106
10	A5	Charged Off	467
11	A5	Current	1
12	A5	Fully Paid	5959
13	A5	Late (31-120 days)	1
14	B1	Charged Off	525
15	B1	Current	1
16	B1	Fully Paid	5804
17	B1	Late (31-120 days)	1
18	B2	Charged Off	724
19	B2	Current	1
20	B2	Fully Paid	6358
21	B2	Late (31-120 days)	2
22	B3	Charged Off	792
23	B3	Current	2

The number of "Charged Off" loans increases within each loan grade as the sub-grade goes from 1 to 5 (e.g., A1 to A5, B1 to B5). This pattern indicates a higher risk of default as the sub-grade number increases, reflecting the increasing risk within each loan grade.

Loan grade and sub-grade do convey useful information about the dataset –
Lower loan grades (D, E, F, G) have higher default rates than loans with higher grades.
"Sub-grade" also provides further differentiation within each grade. As the sub-grade number increases (e.g., A1 to A5), the default rate tends to increase, signaling higher risk.
Therefore, both grade and sub-grade are useful for assessing risk factor of the loans.

(ii) How many loans are there in each grade?

The below table showcases the number of loans under each grade:

A tibble: 7 × 2

grade <chr>	total_loans <int>
A	22605
B	34468
C	26266
D	12346
E	3536
F	705
G	74

Do loan amounts vary by grade?

A tibble: 7 × 2

grade <chr>	sum(loan_amnt) <dbl>
A	322809050
B	430922525
C	314732225
D	146013700
E	42143200
F	7017125
G	1107925

7 rows

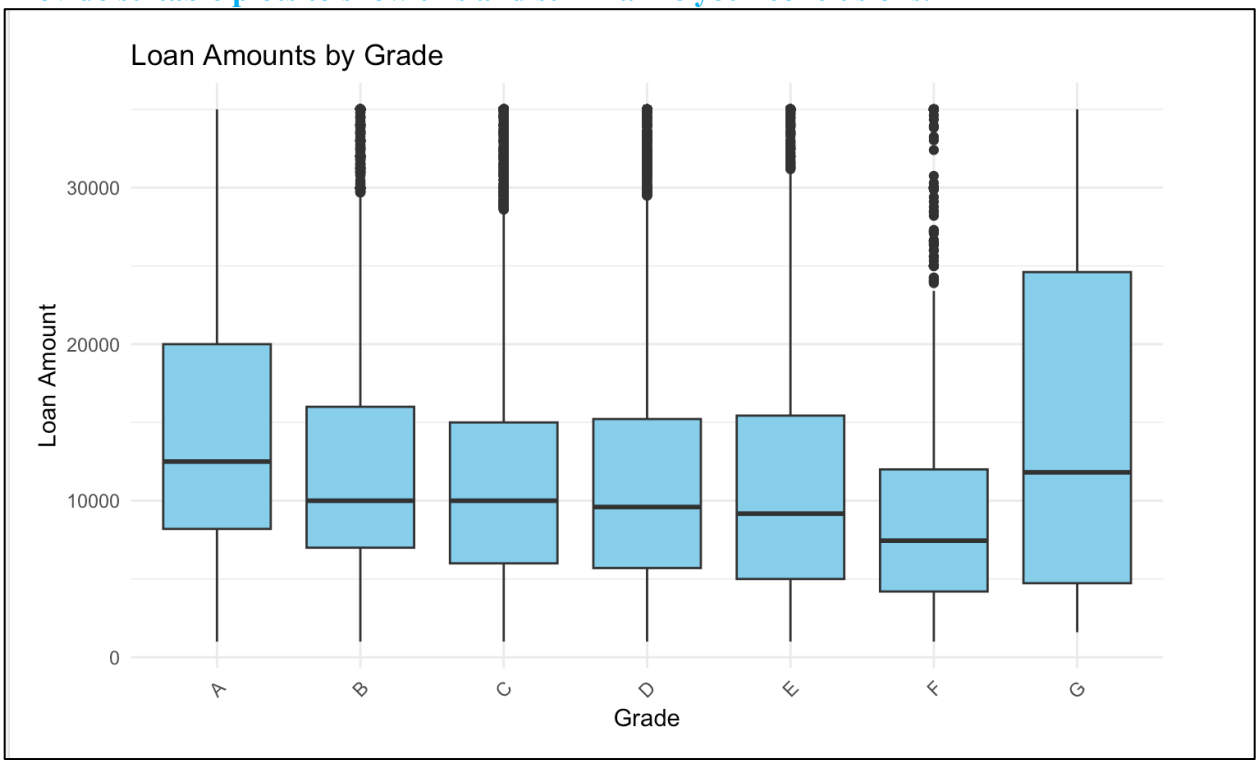
The loan amounts vary significantly across the grades. Higher grades have higher loan amounts, which shows that more loans are sanctioned for higher grades.
On the other hand, since loans of lower grades are riskier, lesser amounts get sanctioned than those categories.

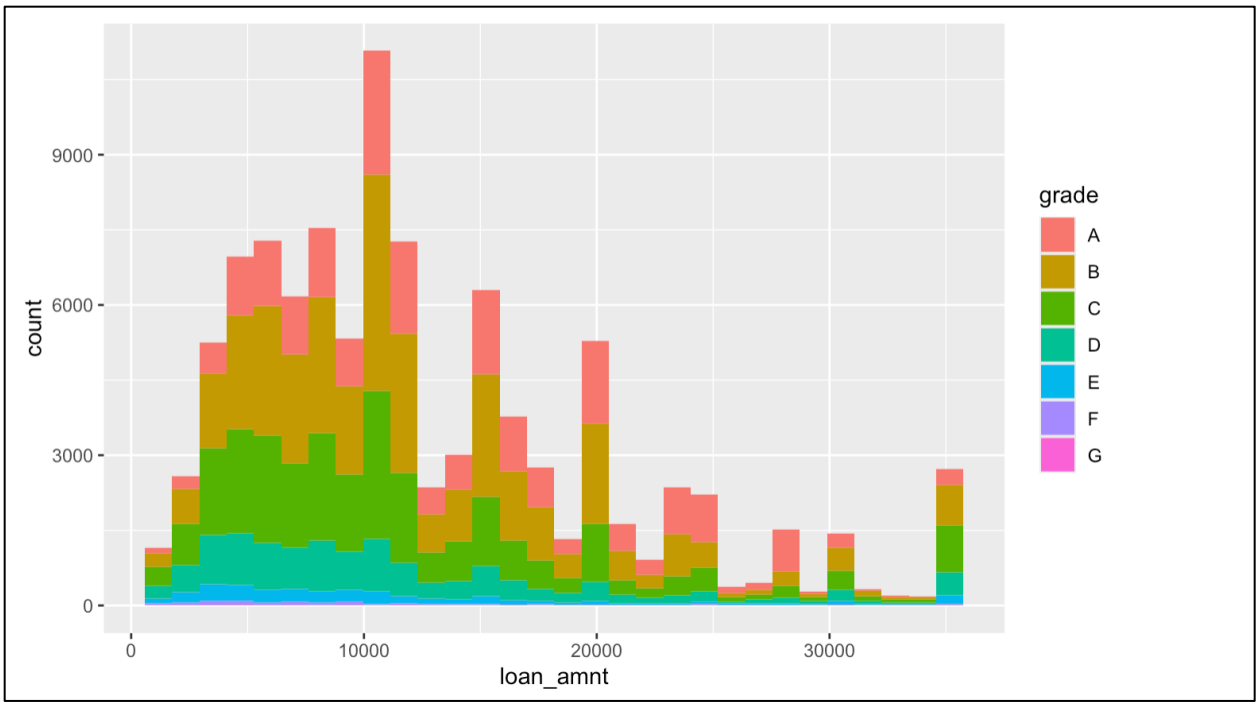
And how does this vary by "loan_status"?

	loan_status	avg_loan_amnt
1	Charged Off	12280.72
2	Current	14447.22
3	Fully Paid	12705.11
4	In Grace Period	11041.67
5	Late (16-30 days)	32000.00
6	Late (31-120 days)	14389.13

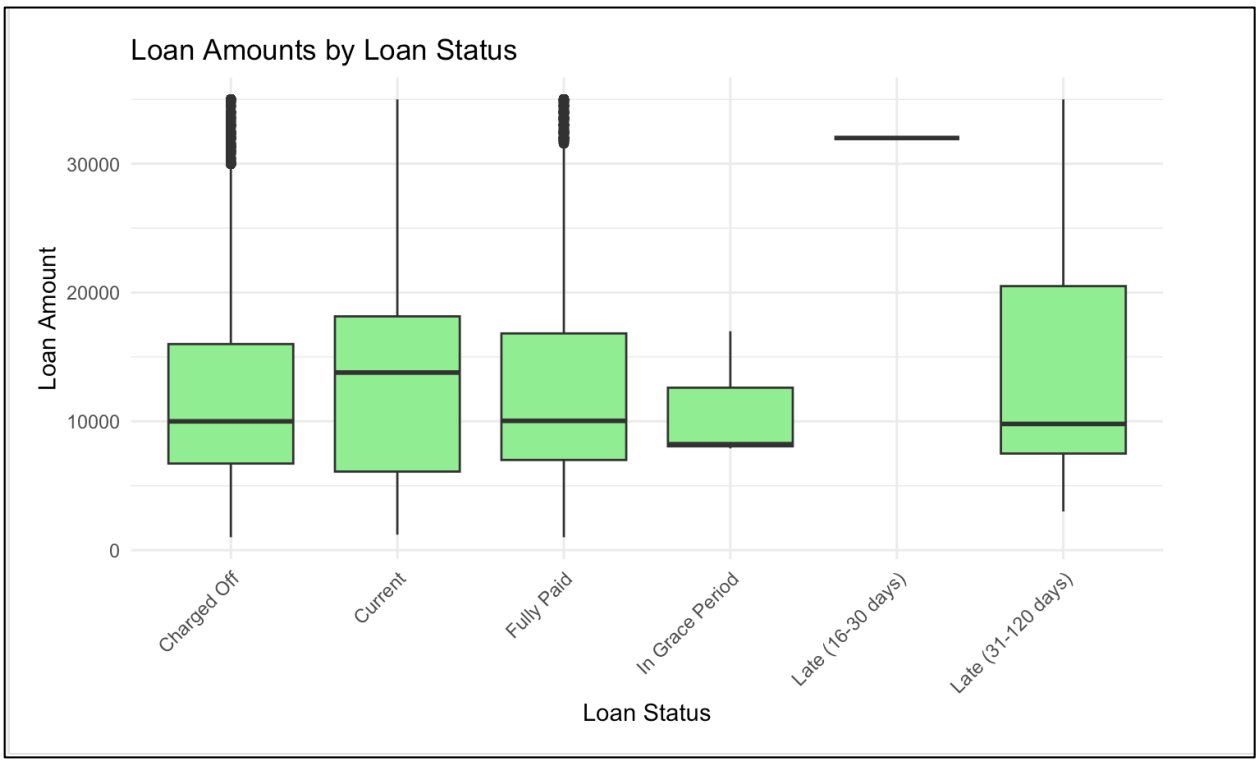
There is little difference between the average loan amount for the different loan statuses, except for the "Late" loan status, which indicates that larger loan amounts may face repayment issues and payments might be delayed.

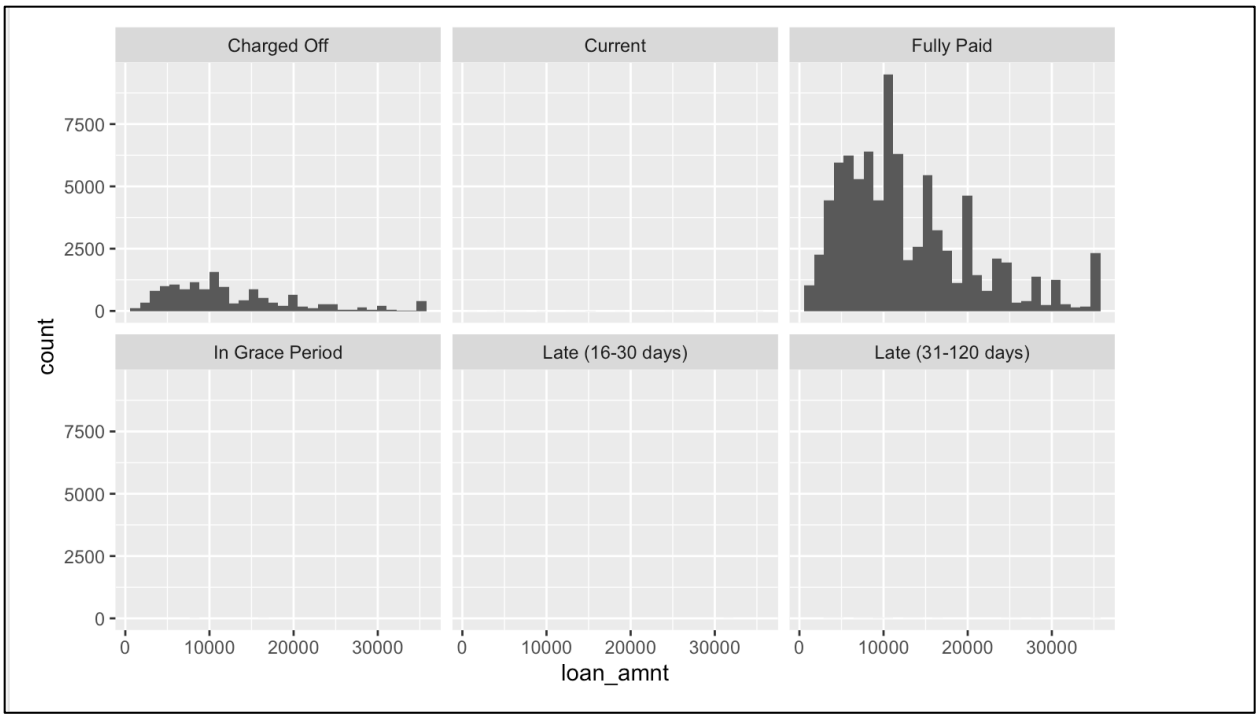
Provide suitable plots to show this and summarize your conclusions.



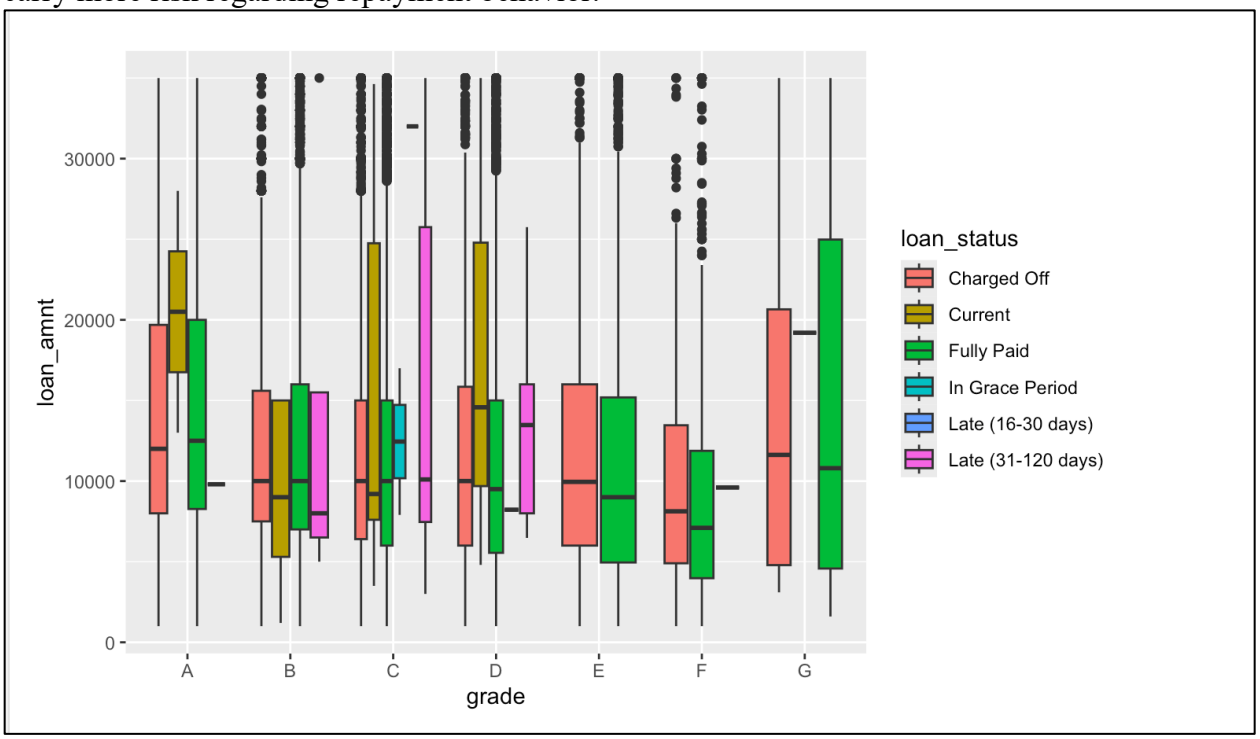


Loan Grade: Loan amounts tend to be consistent across most grades except for Grade G, where loan amounts are higher and more variable. Higher-grade loans are generally smaller and more consistent.





Loan Status: Loans late in payments (especially those late for 16–30 days) tend to be larger, while smaller loans are more likely to be fully paid or charged off. This suggests that larger loans carry more risk regarding repayment behavior.



(iii) 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?

A tibble: 7 × 2

grade <chr>	mean(int_rate) <dbl>
A	7.215713
B	10.851822
C	13.949861
D	17.225287
E	19.987814
F	23.920780
G	26.230270

7 rows

A tibble: 35 × 2

sub_grade <chr>	mean(int_rate) <dbl>
A1	5.708628
A2	6.429465
A3	7.134821
A4	7.518571
A5	8.275552
B1	8.965451
B2	10.035850
B3	10.982730
B4	11.836416
B5	12.347004

The average interest rate increases as the grade worsens from A to G and the same goes for sub-groups. This is expected since low-grade loans are riskier.

A tibble: 7 × 5

grade <chr>	mean_int_rate <dbl>	sd_int_rate <dbl>	min_int_rate <dbl>	max_int_rate <dbl>
A	7.215713	0.9638438	5.32	9.25
B	10.851822	1.4760705	6.00	14.09
C	13.949861	1.2347043	6.00	17.27
D	17.225287	1.2194391	6.00	20.31
E	19.987814	1.4341964	6.00	23.40
F	23.920780	0.9487172	21.28	25.99
G	26.230270	0.8580960	23.91	28.49

A tibble: 35 × 5

sub_grade <chr>	mean_int_rate <dbl>	sd_int_rate <dbl>	min_int_rate <dbl>	max_int_rate <dbl>
A1	5.708628	0.3462865	5.32	6.03
A2	6.429465	0.1686541	6.00	6.97
A3	7.134821	0.3427656	6.68	7.62
A4	7.518571	0.3629408	6.92	8.60
A5	8.275552	0.4379490	6.00	9.25
B1	8.965451	0.7579772	6.00	10.16
B2	10.035850	0.8307176	9.17	11.14
B3	10.982730	0.9267000	6.00	12.12
B4	11.836416	0.8921676	6.00	13.11
B5	12.347004	0.9369288	6.00	14.09

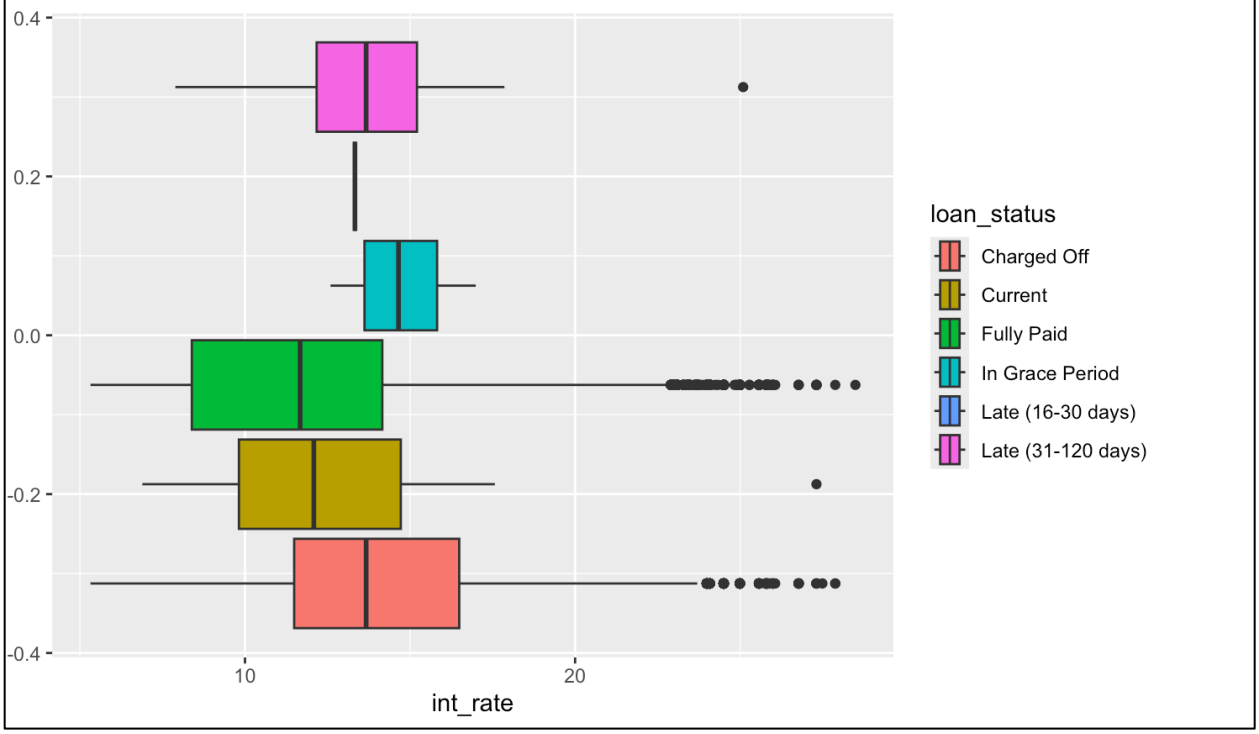
The standard deviation in rates is relatively controlled; however, for riskier loans of worse grades, there is a steep increase in the maximum interest rates.
The results align with the real world, where lenders offer much higher interest rates for riskier loans.

Does "int_rate" relate with loan_status? What do you conclude? (Note that the interest rate is set before a loan is made, and the status is known later).
And does this vary by grade? A box plot can help visualize this. What do you conclude?

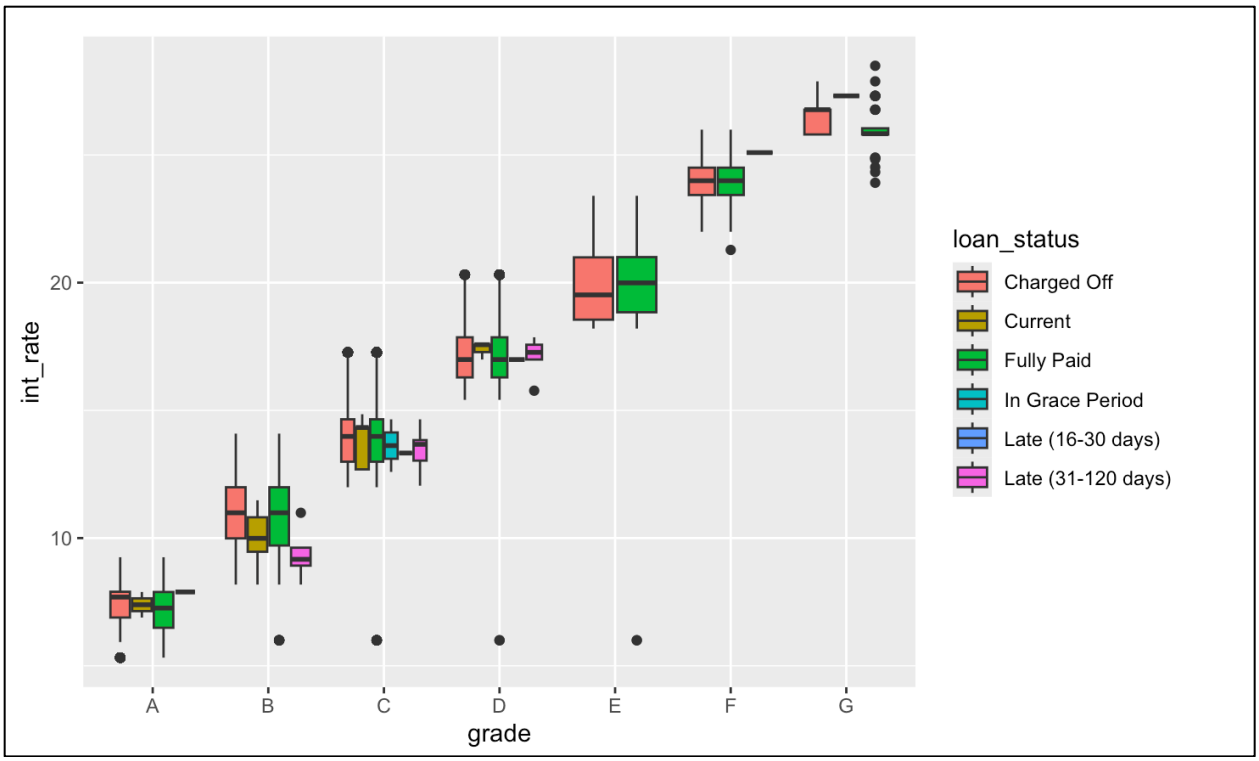
A tibble: 6 × 2

loan_status <chr>	mean_int_rate <dbl>
Charged Off	13.86160
Current	12.96222
Fully Paid	11.76765
In Grace Period	14.74333
Late (16-30 days)	13.33000
Late (31-120 days)	13.79913

6 rows



There is a relationship between interest rates and loan status. Loans with higher interest rates are more likely to be "Charged Off" or "Late", while loans with lower interest rates are more likely to be "Fully Paid" or "Current".
This implies that interest rates serve as a stand-in for the degree of loan risk, with higher rates associated with riskier loans (those with a higher chance of defaulting). This relationship reflects the lender's estimates of loan risk because interest rates are established before loan outcomes are known.



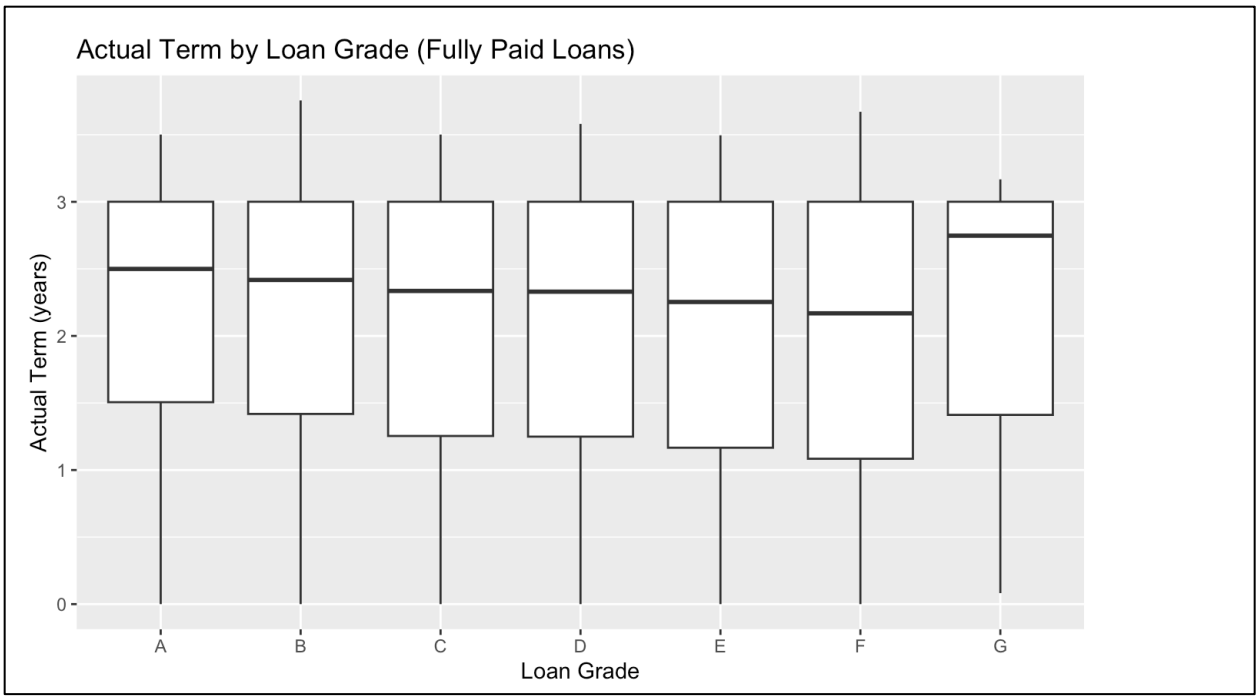
The relationship between interest rates and loan status does vary by grade. Higher loan grades (A, B, C) have lower interest rates across all loan statuses, while lower grades (D, E, F, G) have much higher interest rates, especially for Charged Off and Late loans. This confirms that loans with higher interest rates, particularly in riskier grades, are likelier to default or be late.

(iv) 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).

A tibble: 7 × 5

grade <chr>	mean_term <dbl>	median_term <dbl>	min_term <dbl>	max_term <dbl>
A	2.208855	2.499658	0.00000000	3.501711
B	2.158158	2.417522	0.00000000	3.756331
C	2.090080	2.335387	0.00000000	3.501711
D	2.064721	2.329911	0.00000000	3.581109
E	2.016817	2.253251	0.00000000	3.496235
F	1.993012	2.168378	0.00000000	3.671458
G	2.170725	2.747433	0.08213552	3.167693

7 rows



Summary of Actual Term by Loan Grade:

- On average, the mean actual term is consistent across loan grades, ranging from approximately 1.99 years (Grade F) to 2.21 years (Grade A). This suggests that borrowers typically take about 2 years to repay their loans fully, regardless of loan grade.
- The median values are generally higher than the means, indicating that most loans are paid off closer to 2.3–2.7 years, with a few loans being paid off much earlier, as indicated by the minimum values being zero for most grades.
- The maximum term is close to 3.5 years for most grades, suggesting that some loans take almost the full 3 years to be paid back, which aligns with the common loan term duration.

Box Plot:

- Most of the loans across all grades are paid back within a 2–3-year range.
- There are a few outliers for loans with a very short actual term (close to 0 years), which might represent loans that were paid off immediately.

Therefore, the loan grade does not have a major impact on how long borrowers take to repay their loans when they do manage to repay them fully.

(v) Calculate the annual return for a loan. Explain how you calculate the percentage annual return.

To calculate the annual return for a loan, we use the formula –

Annual return =

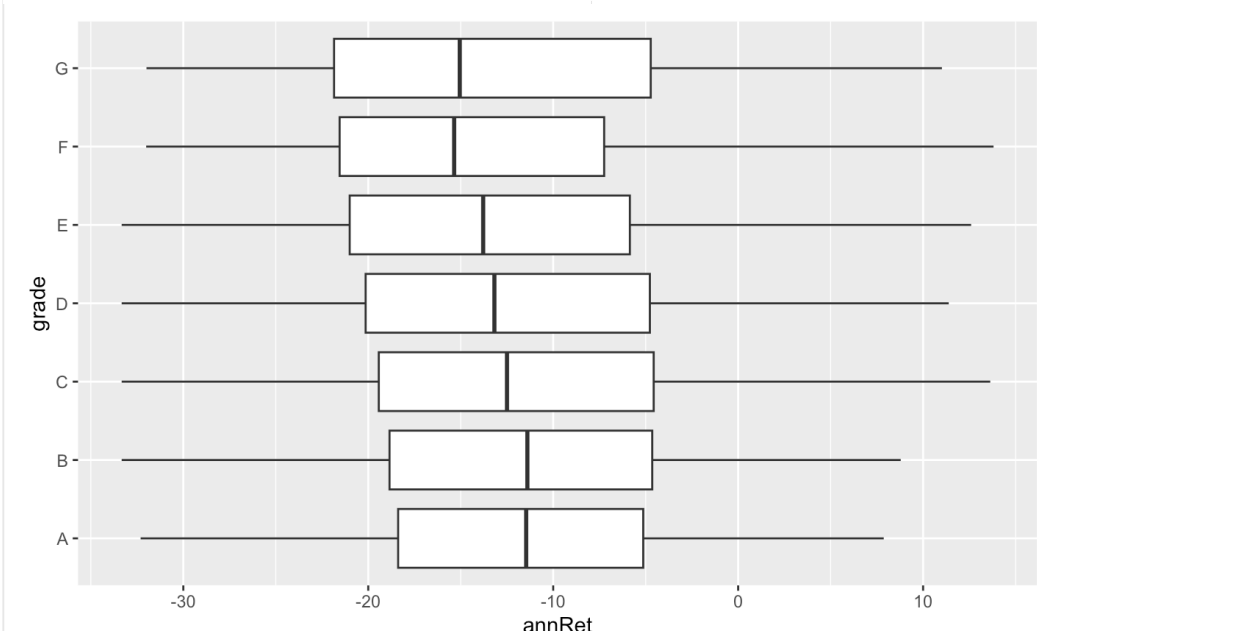
((Total payments – Funded Amount)/Funded Amount) * (12/Loan Term (in months)) * 100

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

A tibble: 7 × 8

grade <chr>	nLoans <int>	avgInterest <dbl>	avgLoanAmt <dbl>	avgPmnt <dbl>	avgRet <dbl>	minRet <dbl>	maxRet <dbl>
A	1210	7.474818	13738.43	8780.025	-12.02192	-32.31047	7.872319
B	3756	10.960293	12298.76	7988.740	-11.73361	-33.33333	8.791000
C	4707	13.936660	12002.78	7687.367	-12.03394	-33.33333	13.633101
D	2839	17.176869	12104.63	7694.619	-12.29094	-33.33333	11.386476
E	1013	19.943110	12590.33	7604.259	-13.05414	-33.33333	12.598583
F	227	24.028414	10505.18	6104.377	-13.48657	-32.00811	13.808784
G	31	26.492581	14411.29	10354.758	-13.18881	-31.99226	11.018248

7 rows



The returns are negative, as seen in the output. This occurs because the total payment received is less than the funded amount. This means the investor loses money on these loans.
From the table and boxplot, we can see that returns for charged-off loans are generally negative across all grades. Grade "A" has a relatively higher average return than the lower grades, but all grades show significant negative returns, indicating losses for investors.

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

Summary by grade

A tibble: 7 × 11

grade <chr>	nLoans <int>	sum(loan_status == "Charged Off") <int>	mean(int_rate) <dbl>	sd(int_rate) <dbl>	mean(loan_amnt) <dbl>	mean(total_pymnt) <dbl>	mean(annRet) <dbl>	sd(annRet) <dbl>
A	22605	1210	7.215713	0.9638438	14280.427	15321.31	2.379832	4.066903
B	34468	3756	10.851822	1.4760705	12502.104	13639.28	2.996227	6.118257
C	26266	4707	13.949861	1.2347043	11982.495	13017.07	2.837244	8.234219
D	12346	2839	17.225287	1.2194391	11826.802	12855.36	2.961334	9.856141
E	3536	1013	19.987814	1.4341964	11918.326	12623.59	2.410828	11.542508
F	705	227	23.920780	0.9487172	9953.369	10556.87	2.608209	13.049253
G	74	31	26.230270	0.8580960	14971.959	16543.99	1.386757	14.691644

Summary of returns from Fully Paid Loans

A tibble: 7 × 8

grade <chr>	nLoans <int>	avgInterest <dbl>	avgLoanAmt <dbl>	avgPmnt <dbl>	avgRet <dbl>	minRet <dbl>	maxRet <dbl>
A	21392	7.20101	14310.712	15690.89	3.194428	4.952381e-03	5.201111
B	30701	10.83893	12527.493	14330.98	4.797816	3.333333e-04	8.700311
C	21539	13.95312	11974.220	14176.02	6.083803	8.333333e-05	10.043287
D	9498	17.23981	11741.035	14392.77	7.515177	1.641111e-02	11.947800
E	2523	20.00576	11648.514	14638.88	8.620108	1.977186e-02	13.849548
F	477	23.86711	9691.509	12669.80	10.245340	2.546667e-02	16.207565
G	42	26.01095	15285.119	20855.45	11.841904	6.453700e-01	16.214910

Summary of returns from “Charged off” Loans

A tibble: 7 × 8

grade <chr>	nLoans <int>	avgInterest <dbl>	avgLoanAmt <dbl>	avgPmnt <dbl>	avgRet <dbl>	minRet <dbl>	maxRet <dbl>
A	1210	7.474818	13738.43	8780.025	-12.02192	-32.31047	7.872319
B	3756	10.960293	12298.76	7988.740	-11.73361	-33.33333	8.791000
C	4707	13.936660	12002.78	7687.367	-12.03394	-33.33333	13.633101
D	2839	17.176869	12104.63	7694.619	-12.29094	-33.33333	11.386476
E	1013	19.943110	12590.33	7604.259	-13.05414	-33.33333	12.598583
F	227	24.028414	10505.18	6104.377	-13.48657	-32.00811	13.808784
G	31	26.492581	14411.29	10354.758	-13.18881	-31.99226	11.018248

Interest rate vs returns:

For fully paid loans, the average returns are positive but often lower than the average interest rates.

For charged-off loans, the returns are negative, reflecting the losses incurred when the borrower defaults.

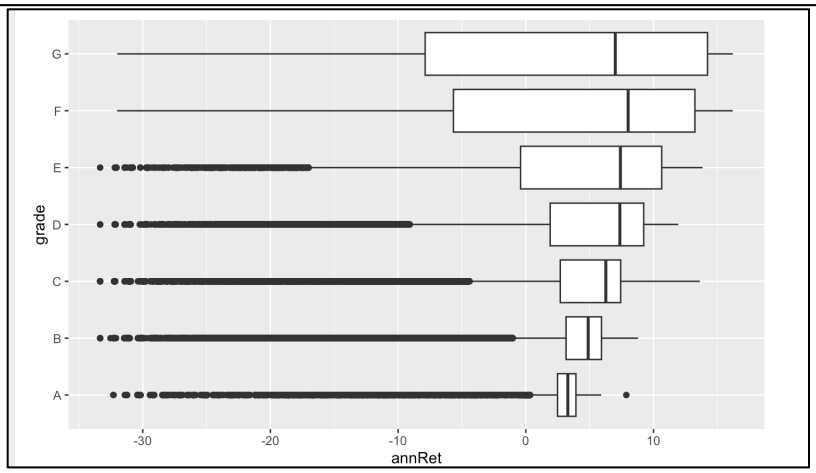
Higher-grade loans (A, B, C) typically have lower interest rates and more stable, positive returns when fully paid. However, as the grade decreases (e.g., F, G), the interest rates increase, but so do the risks, leading to more variability and higher negative returns when loans are charged off.

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

A tibble: 7 × 11

grade <chr>	nLoans <int>	sum(loan_status == "Charged Off") <int>	mean(int_rate) <dbl>	sd(int_rate) <dbl>	mean(loan_amnt) <dbl>	mean(total_pymnt) <dbl>	mean(annRet) <dbl>	sd(annRet) <dbl>
A	22605	1210	7.215713	0.9638438	14280.427	15321.31	2.379832	4.066903
B	34468	3756	10.851822	1.4760705	12502.104	13639.28	2.996227	6.118257
C	26266	4707	13.949861	1.2347043	11982.495	13017.07	2.837244	8.234219
D	12346	2839	17.225287	1.2194391	11826.802	12855.36	2.961334	9.856141
E	3536	1013	19.987814	1.4341964	11918.326	12623.59	2.410828	11.542508
F	705	227	23.920780	0.9487172	9953.369	10556.87	2.608209	13.049253
G	74	31	26.230270	0.8580960	14971.959	16543.99	1.386757	14.691644

7 rows | 1-9 of 11 columns



A tibble: 35 × 7

sub_grade <chr>	nLoans <int>	mean(int_rate) <dbl>	mean(annRet) <dbl>	sd(annRet) <dbl>	min(annRet) <dbl>	max(annRet) <dbl>
A1	3614	5.708628	2.147418	2.694598	-28.338800	3.249031
A2	3455	6.429465	2.274063	3.301680	-31.404800	3.826865
A3	3669	7.134821	2.413072	3.948126	-31.255889	4.301602
A4	5439	7.518571	2.357750	4.479468	-32.310467	7.872319
A5	6428	8.275552	2.567063	4.720588	-32.282767	5.201111
B1	6331	8.965451	2.644124	5.176605	-32.262904	5.691200
B2	7085	10.035850	2.859904	5.689771	-33.333333	6.524430
B3	7398	10.982730	3.076145	6.148148	-33.333333	7.011524
B4	7168	11.836416	3.179475	6.622659	-32.525095	8.791000
B5	6486	12.347004	3.195158	6.754167	-33.333333	8.700311

1–10 of 35 rows Previous 1 2 3 4 Next

The **average returns** (as seen in the first and second tables) generally decrease as the loan grade declines (from A to G).
The returns' **SD** tends to increase as the grade worsens.
Higher-grade loans (A, B, C) tend to offer lower returns but also lower volatility, while lower-grade loans (F, G) offer potentially higher returns at the cost of greater risk. The sub-grade breakdown adds further granularity, where different sub-grades can show significant variation in both returns and risk even within the same grade.

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

Grades B and C loans have a good balance between stability and returns. Hence, I would primarily choose loans in these two grades.
A part of the investment can be made towards Grade D or E, offering higher returns while still controlling the risk.
This would help create a diversified portfolio.

(vi) What are people borrowing money for (loan purpose)? Examine how many loans, average amounts, etc. by purpose? Do loan amounts vary by purpose?

	purpose	nLoans	avgLoanAmt	totalLoanAmt
1	debt_consolidation	57515	13185.942	758389450
2	credit_card	24538	13507.277	331441575
3	home_improvement	5726	11743.787	67244925
4	other	5169	8293.519	42869200
5	major_purchase	1929	9350.505	18037125
6	medical	1075	7595.000	8164625
7	small_business	1032	13966.764	14413700
8	car	996	8103.087	8070675
9	moving	689	6842.126	4714225
10	vacation	675	5602.593	3781750
11	house	404	13099.134	5292050
12	wedding	190	9347.105	1775950
13	renewable_energy	62	8879.032	550500

The most common purpose for borrowing money is debt consolidation, with 57515 loans. This is followed by credit card refinancing, with 24538 loans, and home improvement, with 5,726 loans.
Purpose does have an impact on loan amount:

- The highest average loan amounts are for **small business** loans ($13,966.76) and **credit card refinancing** ($13,507.28).
- The lowest average loan amounts are for purposes like **vacation** ($5,602.59) and **moving** ($6,842.13).
- This suggests that borrowers requesting loans for business purposes or consolidating debt tend to borrow larger amounts. In contrast, loans for more personal or short-term needs like vacations and moving tend to be smaller.

Do defaults vary by purpose?

	purpose	nDefaults	avgLoanAmt	totalLoanAmt
1	debt_consolidation	8292	12764.794	105845675
2	credit_card	2825	12971.186	36643600
3	other	814	8437.838	6868400
4	home_improvement	742	12260.007	9096925
5	major_purchase	253	10150.791	2568150
6	small_business	212	14644.693	3104675
7	medical	195	7459.872	1454675
8	moving	131	7117.939	932450
9	vacation	117	6246.368	730825
10	car	110	8993.182	989250
11	house	59	11713.136	691075
12	wedding	22	9710.227	213625
13	renewable_energy	11	11443.182	125875

'Debt Consolidation' and 'Credit Card' purposes have the highest number of defaults, while categories like Wedding and Renewable Energy have the fewest defaults. The loan amounts also vary significantly across different purposes, with Small Business loans showing a notably high average loan amount for the defaults observed.

Does loan-grade assigned by Lending Club vary by purpose? Summarize what you find.

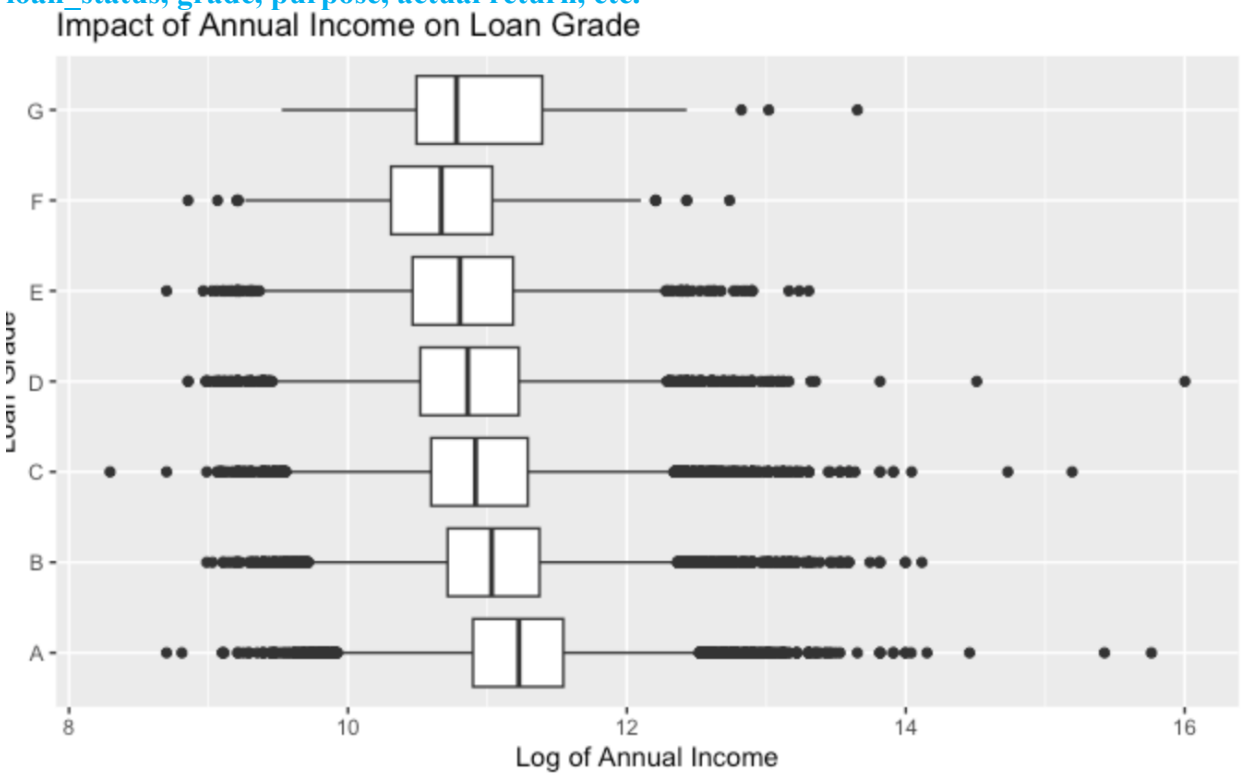
Loan purposes like credit card, debt consolidation, and home improvement are more likely to have loans across various grades, reflecting different levels of risk.

Higher-risk grades (E, F, and G) are less frequent across all purposes, which indicates that Lending Club may be more cautious when assigning higher-risk loans to certain purposes.

Lower-risk loans, like those for **car** and **medical expenses**, tend to be concentrated in the A and B grades. However, there are instances of loans in higher grades, indicating a mix of borrower profiles for each purpose.

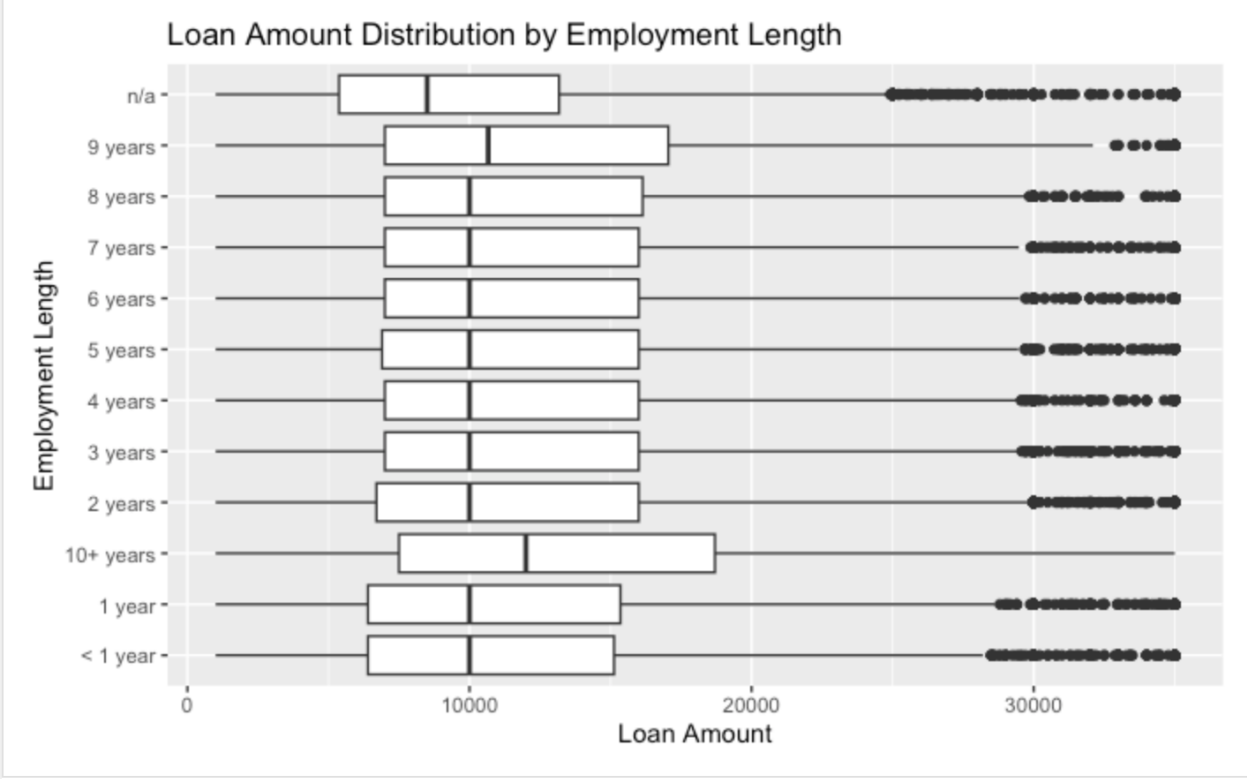
	purpose	grade	nLoans	avgLoanAmt	avgInterest
1	car	A	269	8790.892	7.090967
2	car	B	352	7723.011	10.898097
3	car	C	233	7576.824	13.820129
4	car	D	103	8556.068	17.127087
5	car	E	30	9020.833	19.753667
6	car	F	8	7931.250	23.617500
7	car	G	1	6675.000	25.800000
8	credit_card	A	8062	14878.833	7.120409
9	credit_card	B	9756	12904.777	10.733645
10	credit_card	C	4930	12771.050	13.959318
11	credit_card	D	1475	12662.542	17.237939
12	credit_card	E	272	12887.316	19.882169
13	credit_card	F	41	10263.415	23.628293
14	credit_card	G	2	12375.000	25.800000

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



Higher incomes are associated with better loan grades, such as A and B, as these borrowers are seen as lower risk.

Lower loan grades (C to G) have a wider range of income levels, indicating that these grades are assigned based on a variety of risk factors beyond just income.
The plot provides a clear visual of how income influences loan grading, with a general trend of higher income correlating with better loan terms (higher grades).



Overall, Annual Income and Employment Length are strong predictors of loan amount, loan grade, and loan status. Higher income and longer employment often result in better loan terms, larger loan amounts, and lower default rates. FICO Score plays a crucial role in determining loan grade, interest rates, and default risk.
Together, these characteristics will help lenders assess borrower risk and determine loan terms accordingly.

(viii) 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 attributes:

- Inquiry rate: Borrowers with a higher inquiry rate are more likely to be in late payment or default statuses, making this a strong predictor of default risk
- Open account rate: Borrowers with a higher number of open accounts relative to their total accounts are more likely to struggle with repayments, as evidenced by higher late payment rates.
- Loan-to-Income Ratio: A higher loan-to-income ratio correlates with a higher likelihood of late payments or defaults, especially for borrowers in lower loan grades.

Hence, these derived attributes provide significant insights into borrower behavior and risk, making them valuable for predicting loan defaults. By incorporating them into a model, lenders can more accurately assess the likelihood of a borrower defaulting.

loan_status <chr>	mean_inq_rate <dbl>	mean_open_acc_rate <dbl>	mean_loan_to_income_ratio <dbl>
Charged Off	Inf	Inf	Inf
Current	0.312500	Inf	0.2284723
Fully Paid	Inf	Inf	0.1953378
In Grace Period	NaN	NaN	0.2040103
Late (16-30 days)	NaN	NaN	0.2560000
Late (31-120 days)	1.111111	Inf	0.2145823

6 rows

grade <chr>	mean_inq_rate <dbl>	mean_open_acc_rate <dbl>	mean_loan_to_income_ratio <dbl>
A	Inf	Inf	0.1874570
B	Inf	Inf	0.1939925
C	Inf	Inf	Inf
D	Inf	Inf	0.2118429
E	Inf	Inf	0.2194913
F	Inf	Inf	0.2090334
G	0.1428571	0.9	0.2290988

7 rows

mean_inq_rate <dbl>	mean_open_acc_rate <dbl>	mean_loan_to_income_ratio <dbl>
0.4170824	0.7583691	0.2159541

(b) Are there missing values in the data?

The following columns have missing data.

```
 [1] "emp_title"                  "title"                  "dti"                        "mths_since_last_delinq"
 [5] "mths_since_last_record"     "revol_util"             "last_pymnt_d"               "last_credit_pull_d"
 [9] "mths_since_last_major_derog" "tot_coll_amt"          "tot_cur_bal"                "open_acc_6m"
[13] "open_act_il"                "open_il_12m"            "open_il_24m"                "mths_since_rcnt_il"
[17] "total_bal_il"               "il_util"                "open_rv_12m"                "open_rv_24m"
[21] "max_bal_bc"                 "all_util"               "total_rev_hi_lim"           "inq_fi"
[25] "total_cu_tl"                "inq_last_12m"           "acc_open_past_24mths"       "avg_cur_bal"
[29] "bc_open_to_buy"             "bc_util"                "mo_sin_old_il_acct"         "mo_sin_old_rev_tl_op"
[33] "mo_sin_rcnt_rev_tl_op"      "mo_sin_rcnt_tl"         "mort_acc"                   "mths_since_recent_bc"
[37] "mths_since_recent_bc_dlq"   "mths_since_recent_inq"  "mths_since_recent_revol_delinq" "num_accts_ever_120_pd"
[41] "num_actv_bc_tl"             "num_actv_rev_tl"        "num_bc_sats"                "num_bc_tl"
[45] "num_il_tl"                  "num_op_rev_tl"          "num_rev_accts"              "num_rev_tl_bal_gt_0"
[49] "num_sats"                   "num_tl_120dpd_2m"       "num_tl_30dpd"               "num_tl_90g_dpd_24m"
[53] "num_tl_op_past_12m"         "pct_tl_nvr_dlq"         "percent_bc_gt_75"           "tot_hi_cred_lim"
[57] "total_bal_ex_mort"          "total_bc_limit"         "total_il_high_credit_limit" "hardship_dpd"
[61] "settlement_term"            "loan_to_income_ratio"   "inq_rate"                   "open_acc_rate"
[65] "mort_acc_prop"
```

What is the proportion of missing values in different variables?

A tibble: 65 × 2

skim_variable <chr>	complete_rate <dbl>
dti	0.99999
loan_to_income_ratio	0.99999
last_credit_pull_d	0.99994
title	0.99983
revol_util	0.99952
last_pymnt_d	0.99925
acc_open_past_24mths	0.98949
mort_acc	0.98949
total_bal_ex_mort	0.98949
total_bc_limit	0.98949

1–10 of 65 rows Previous 1 2 3 4 5 6 7 Next

A tibble: 65 × 2

skim_variable <chr>	complete_rate <dbl>
il_util	0.02185
inq_rate	0.01543
open_acc_rate	0.01319
settlement_term	0.00450
hardship_dpd	0.00060

Variables such as dti, loan_to_income_ratio have almost all the values present. On the contrary for open_acc_rate, settlement_term and hardship_dpd, almost all the values are missing.

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.

Step 1: Understanding Missing Values:

For each variable, it's important to understand why values might be missing.

For instance, months_since_last_delinquency is missing when someone has never had a delinquency. It isn't an error in data collection but rather a reflection of a condition that doesn't apply to certain borrowers.

Step 2: Get a summary of the variables with missing data

Step 3: Remove the variables for which all values are missing, since they do not provide any valuable information.

Step 4: Get the new list of variables which have missing values.

```
 [1] "emp_title"                   "title"                   "dti"                     "mths_since_last_delinq"
 [5] "mths_since_last_record"      "revol_util"              "last_pymnt_d"            "last_credit_pull_d"
 [9] "mths_since_last_major_derog" "tot_coll_amt"            "tot_cur_bal"             "open_acc_6m"
[13] "open_act_il"                 "open_il_12m"             "open_il_24m"             "mths_since_rcnt_il"
[17] "total_bal_il"                "il_util"                 "open_rv_12m"             "open_rv_24m"
[21] "max_bal_bc"                  "all_util"                "total_rev_hi_lim"        "inq_fi"
[25] "total_cu_tl"                 "inq_last_12m"            "acc_open_past_24mths"    "avg_cur_bal"
[29] "bc_open_to_buy"              "bc_util"                 "mo_sin_old_il_acct"      "mo_sin_old_rev_tl_op"
[33] "mo_sin_rcnt_rev_tl_op"       "mo_sin_rcnt_tl"          "mort_acc"                "mths_since_recent_bc"
[37] "mths_since_recent_bc_dlq"    "mths_since_recent_inq"   "mths_since_recent_revol_delinq" "num_accts_ever_120_pd"
[41] "num_actv_bc_tl"              "num_actv_rev_tl"         "num_bc_sats"             "num_bc_tl"
[45] "num_il_tl"                   "num_op_rev_tl"           "num_rev_accts"           "num_rev_tl_bal_gt_0"
[49] "num_sats"                    "num_tl_120dpd_2m"        "num_tl_30dpd"            "num_tl_90g_dpd_24m"
[53] "num_tl_op_past_12m"          "pct_tl_nvr_dlq"          "percent_bc_gt_75"        "tot_hi_cred_lim"
[57] "total_bal_ex_mort"           "total_bc_limit"          "total_il_high_credit_limit" "hardship_dpd"
[61] "settlement_term"
```

emp_title	title	dti	mths_since_last_delinq	mths_since_last_record
0.06600	0.00017	0.00001	0.50493	0.83490
revol_util	last_pymnt_d	last_credit_pull_d	mths_since_last_major_derog	tot_coll_amt
0.00048	0.00075	0.00006	0.73092	0.03856
tot_cur_bal	open_acc_6m	open_act_il	open_il_12m	open_il_24m
0.03856	0.97471	0.97471	0.97471	0.97471
mths_since_rcnt_il	total_bal_il	il_util	open_rv_12m	open_rv_24m
0.97554	0.97471	0.97815	0.97471	0.97471
max_bal_bc	all_util	total_rev_hi_lim	inq_fi	total_cu_tl
0.97471	0.97471	0.03856	0.97471	0.97471
inq_last_12m	acc_open_past_24mths	avg_cur_bal	bc_open_to_buy	bc_util
0.97471	0.01051	0.03857	0.02029	0.02098
mo_sin_old_il_acct	mo_sin_old_rev_tl_op	mo_sin_rcnt_rev_tl_op	mo_sin_rcnt_tl	mort_acc
0.07509	0.03857	0.03857	0.03856	0.01051
mths_since_recent_bc	mths_since_recent_bc_dlq	mths_since_recent_inq	mths_since_recent_revol_delinq	num_accts_ever_120_pd
0.01955	0.75104	0.11755	0.65221	0.03856
num_actv_bc_tl	num_actv_rev_tl	num_bc_sats	num_bc_tl	num_il_tl
0.03856	0.03856	0.02235	0.03856	0.03856
num_op_rev_tl	num_rev_accts	num_rev_tl_bal_gt_0	num_sats	num_tl_120dpd_2m
0.03856	0.03856	0.03856	0.02235	0.07426
num_tl_30dpd	num_tl_90g_dpd_24m	num_tl_op_past_12m	pct_tl_nvr_dlq	percent_bc_gt_75
0.03856	0.03856	0.03856	0.03874	0.02095
tot_hi_cred_lim	total_bal_ex_mort	total_bc_limit	total_il_high_credit_limit	hardship_dpd
0.03856	0.01051	0.01051	0.03856	0.99940
settlement_term				
0.99550				

Step 5: Handling Missing Values by Variable Type:

- **Time-Related Variables** (e.g., months_since_last_delinquency, months_since_last_record):
  - These variables might be missing because the event hasn't occurred yet for that borrower. A sensible replacement is **999**, representing "no event" for cases where these values are missing.
- **Numeric Variables** (e.g., loan_amnt, annual_inc, bc_open_to_buy):
  - For numeric variables where missing values occur, replacing them with the **median** makes sense because it is a robust measure that isn't heavily influenced by outliers.
- **Categorical Variables** (e.g., loan_status, title):
  - Missing values in categorical variables can be replaced with a placeholder like **"Unknown"** or the **most frequent category**, depending on the variable.

What is a sensible value to replace the missing values in this case?

- For variables like 'open_acc_6m', we replaced missing values with 0, assuming no accounts were opened.
- For time-based variables such as 'mths_since_last_record', missing values were set to 0, indicating no prior records.
- Variables with too many missing values (more than 70%) were removed.
- Imputation was performed for remaining missing values based on median or placeholder values where appropriate.

Are there some variables you will exclude from your model due to missing values?

Yes, based on the proportion of missing values, some variables should be excluded from the model due to having too many missing values, which may impact model accuracy and performance. A variable with more than 80% missing values is typically excluded because imputing or filling in the missing values might introduce significant bias.

(c) Consider the potential for data leakage. You do not want to include variables in your model that 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 that may have been updated during the loan period (i.e. after the loan is funded). Identify and explain which variables you will exclude in developing predictive models.

Key variables to exclude:

- last_pymnt_d: This is the date of the last payment, and it's updated during the loan period.
- last_credit_pull_d: This is the date when the credit was last checked, which occurs after the loan is issued.
- total_rec_prncp, total_rec_int, total_rec_late_fee: These variables represent payments and interest collected during the loan period.
- recoveries, collection_recovery_fee: These are relevant only after a default has occurred.

Column names remaining after removing variables which might cause potential leakage.

```
 [1] "loan_amnt"                      "funded_amnt"                "funded_amnt_inv"                "term"
 [5] "int_rate"                       "installment"                "grade"                          "sub_grade"
 [9] "emp_title"                      "emp_length"                 "home_ownership"                 "annual_inc"
[13] "verification_status"            "issue_d"                    "loan_status"                    "pymnt_plan"
[17] "purpose"                        "title"                      "zip_code"                       "addr_state"
[21] "dti"                            "delinq_2yrs"                "earliest_cr_line"               "inq_last_6mths"
[25] "mths_since_last_delinq"         "mths_since_last_record"     "open_acc"                       "pub_rec"
[29] "revol_bal"                      "revol_util"                 "total_acc"                      "initial_list_status"
[33] "out_prncp"                      "out_prncp_inv"              "total_pymnt"                    "total_pymnt_inv"
[37] "last_pymnt_amnt"                "collections_12_mths_ex_med" "mths_since_last_major_derog"    "policy_code"
[41] "application_type"               "acc_now_delinq"             "tot_coll_amt"                   "tot_cur_bal"
[45] "open_acc_6m"                    "total_rev_hi_lim"           "acc_open_past_24mths"           "avg_cur_bal"
[49] "bc_open_to_buy"                 "bc_util"                    "chargeoff_within_12_mths"       "delinq_amnt"
[53] "mo_sin_old_il_acct"             "mo_sin_old_rev_tl_op"       "mo_sin_rcnt_rev_tl_op"          "mo_sin_rcnt_tl"
[57] "mort_acc"                       "mths_since_recent_bc"       "mths_since_recent_bc_dlq"       "mths_since_recent_inq"
[61] "mths_since_recent_revol_delinq" "num_accts_ever_120_pd"      "num_actv_bc_tl"                 "num_actv_rev_tl"
[65] "num_bc_sats"                    "num_bc_tl"                  "num_il_tl"                      "num_op_rev_tl"
[69] "num_rev_accts"                  "num_rev_tl_bal_gt_0"        "num_sats"                       "num_tl_120dpd_2m"
[73] "num_tl_30dpd"                   "num_tl_90g_dpd_24m"         "num_tl_op_past_12m"             "pct_tl_nvr_dlq"
[77] "percent_bc_gt_75"               "pub_rec_bankruptcies"       "tax_liens"                      "tot_hi_cred_lim"
[81] "total_bal_ex_mort"              "total_bc_limit"             "total_il_high_credit_limit"     "hardship_flag"
[85] "disbursement_method"            "debt_settlement_flag"       "annRet"                         "loan_to_income_ratio"
```

3. Do a univariate analyses 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).

From your output, variables like:

- annRet (AUC = 0.9787)
- out_prncp and out_prncp_inv (AUC = 0.9444)
- total_pymnt (AUC = 0.7558)

have higher AUC values, meaning they are highly predictive of loan_status.

Useful predictors for predicting loan_status include variables with AUC values around 0.6–0.8, such as annRet, mths_since_recent_inq, and bc_util.

Variables to exclude: Any with AUC values close to 1 (such as out_prncp, out_prncp_inv, and total_pymnt) should be excluded due to the likelihood of data leakage. These variables provide information about the loan after it has been funded, so they should not be used in the model.

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?

We decided to split the data into a 70/30 ratio, with 70% of the data allocated for training and 30% for validation.

Why 70/30?

- The 70% training data ensures that the model has a large enough sample size to learn patterns, relationships, and nuances in the data. This helps the model generalize better when making predictions.
- The 30% validation data provides sufficient data to evaluate the model's performance on unseen examples. This split allows us to assess how well the model performs in a real-world scenario where it encounters new, unseen data.

By maintaining this ratio, we balance having enough data to train the model while also reserving enough data to reliably test the model's performance.

(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.

For evaluating our performance, typical measures that we can include are accuracy, precision, recall, F1 score, and ROC-AUC. For this task, **ROC-AUC** is a strong measure because it evaluates how well the model distinguishes between classes, such as predicting whether a loan will default or not.

Confusion matrix will also give us insights into how well the model handles true positives, false positives, etc. We can also generate a **lift chart** to see how well the model can rank predictions. Overall, we believe that to predict loan defaults based on various borrower attributes, and evaluating the model's performance with ROC-AUC and other metrics will be key to understanding its effectiveness.

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]

To develop the decision tree, we used rpart.

Key Parameters Experimented With:

1. **Complexity Parameter (CP)**:
    - A series of complexity parameters (CP) were explored with varying numbers of splits (as seen in the table).
    - The tree was pruned based on the complexity parameter that minimized the cross-validation error (xerror). In this case, a CP of 0.0079015 was selected for pruning, which corresponds to a cross-validation error of 0.30446 with a standard deviation of 0.0054506.

2. **Tree Depth and Splits**:
    - The maximum depth of the tree was limited to 5 (maxdepth = 5) to prevent overfitting.
    - The number of splits (nsplit) was tested up to 15, allowing the tree to grow more complex before selecting the optimal pruning point.

The model's performance is evaluated using a confusion matrix, accuracy, kappa, and additional statistical metrics as shown in the output image below.

```
Classification tree:
rpart(formula = loan_status ~ ., data = lcdfTrn %>% select(-all_of(existing_varsOmit)),
    method = "class", parms = list(split = "information"), control = rpart.control(cp = 0,
        minsplit = 50, maxdepth = 5))

Variables actually used in tree construction:
[1] installment     loan_amnt       total_pymnt_inv

Root node error: 9745/70000 = 0.13921

n= 70000

         CP nsplit rel error  xerror      xstd
1 0.1475629      0   1.00000 1.00000 0.0093985
2 0.0830682      2   0.70487 0.70498 0.0080773
3 0.0399179      4   0.53874 0.54192 0.0071704
4 0.0377116      6   0.45890 0.43356 0.0064657
5 0.0173935      8   0.38348 0.38943 0.0061478
6 0.0151873     10   0.34869 0.36501 0.0059626
7 0.0138533     12   0.31832 0.33905 0.0057576
8 0.0079015     13   0.30446 0.31278 0.0055406
9 0.0000000     15   0.28866 0.29728 0.0054077
Confusion Matrix and Statistics

             Reference
Prediction    Charged Off Fully Paid
  Charged Off        7234        302
  Fully Paid         2511      59953

               Accuracy : 0.9598
                 95% CI : (0.9583, 0.9613)
    No Information Rate : 0.8608
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8147

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.7423
            Specificity : 0.9950
         Pos Pred Value : 0.9599
         Neg Pred Value : 0.9598
             Prevalence : 0.1392
         Detection Rate : 0.1033
   Detection Prevalence : 0.1077
      Balanced Accuracy : 0.8687

       'Positive' Class : Charged Off
```

Findings:

1. Accuracy: The model achieved a very high accuracy of 95.98%, which suggests strong performance in classifying both "Charged Off" and "Fully Paid" loans.
2. Kappa: A Kappa value of 0.8147 suggests strong agreement between the predicted and actual classifications, accounting for chance agreement.
3. Sensitivity (Recall): Sensitivity is 0.7423, indicating that 74.23% of the "Charged Off" loans were correctly identified.
4. Specificity: The specificity is very high (99.50%), meaning the model performs exceptionally well at correctly identifying "Fully Paid" loans.

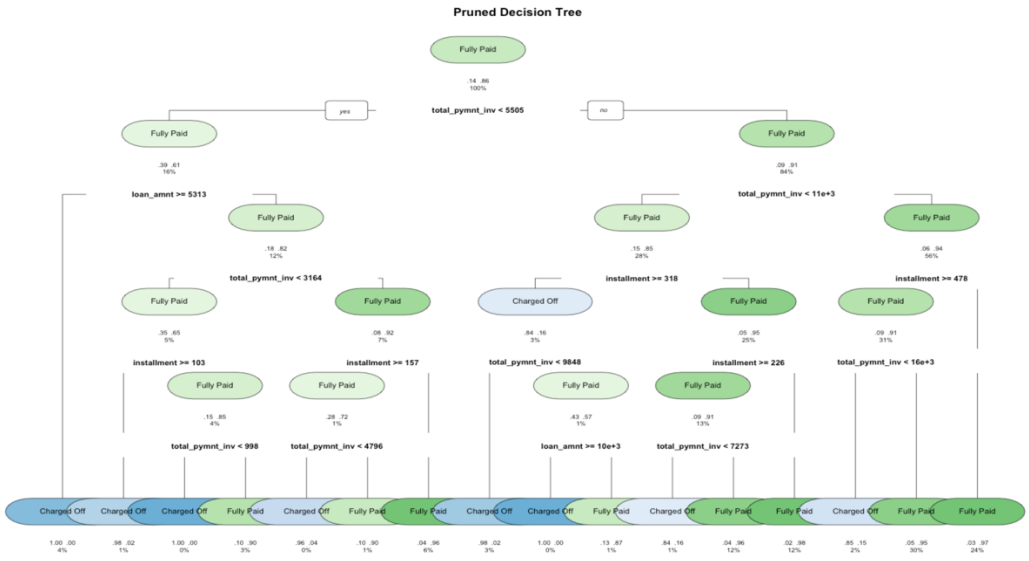
5. Balanced Accuracy: The balanced accuracy of 86.87% reflects the average of sensitivity and specificity, indicating good overall performance in both classes.

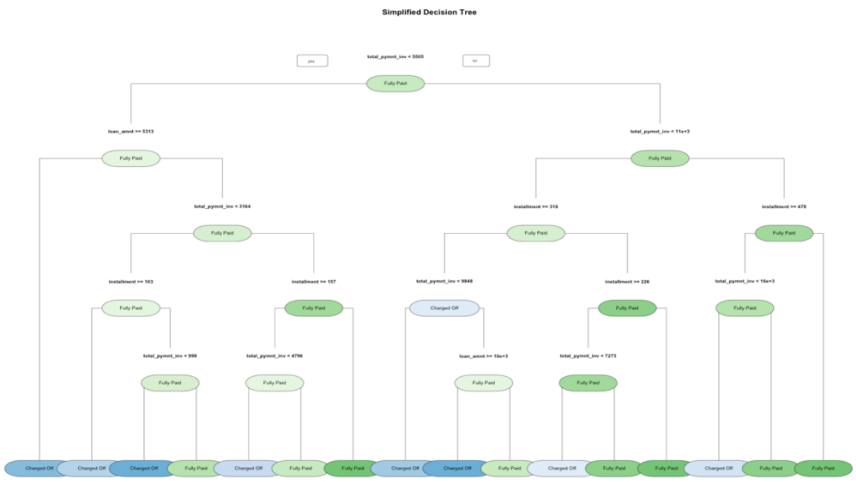
Addressing Potential Leakage:

The model shows a high accuracy and specificity, especially for "Fully Paid" loans. Such high performance could indicate potential data leakage. Specifically, total_pymnt_inv (total payment) could potentially introduce leakage because it might contain post-loan information that wouldn't be available at the time of loan approval.

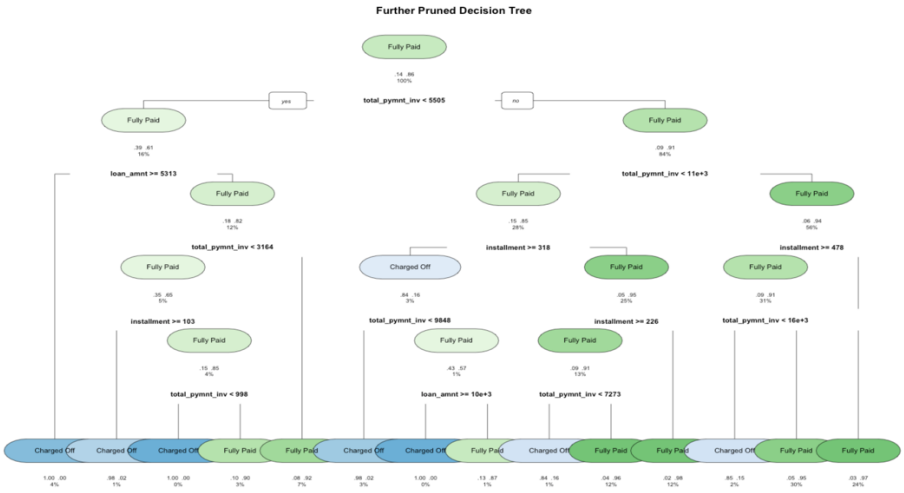
To address this potential leakage one thing, we can do is to ensure to review the feature set being used in the future data (such as payments made after loan issuance)being used to predict loan default.

Identify the best tree model. Why do you consider it best?





Simplified Decision Tree:
This model further simplifies the decision-making process. The tree still accurately classifies loans but with even fewer decision nodes, making it more interpretable while still focusing on major contributing factors like loan_amnt, installment, and total_pymnt_inv.



Further Pruned Decision Tree:
This tree shows even more simplifications compared to the initial pruned tree, but it may be overly simplified, possibly losing some prediction power compared to the previous models. Still, it provides a highly interpretable model with key features such as installment, loan_amnt, and total_pymnt_inv.

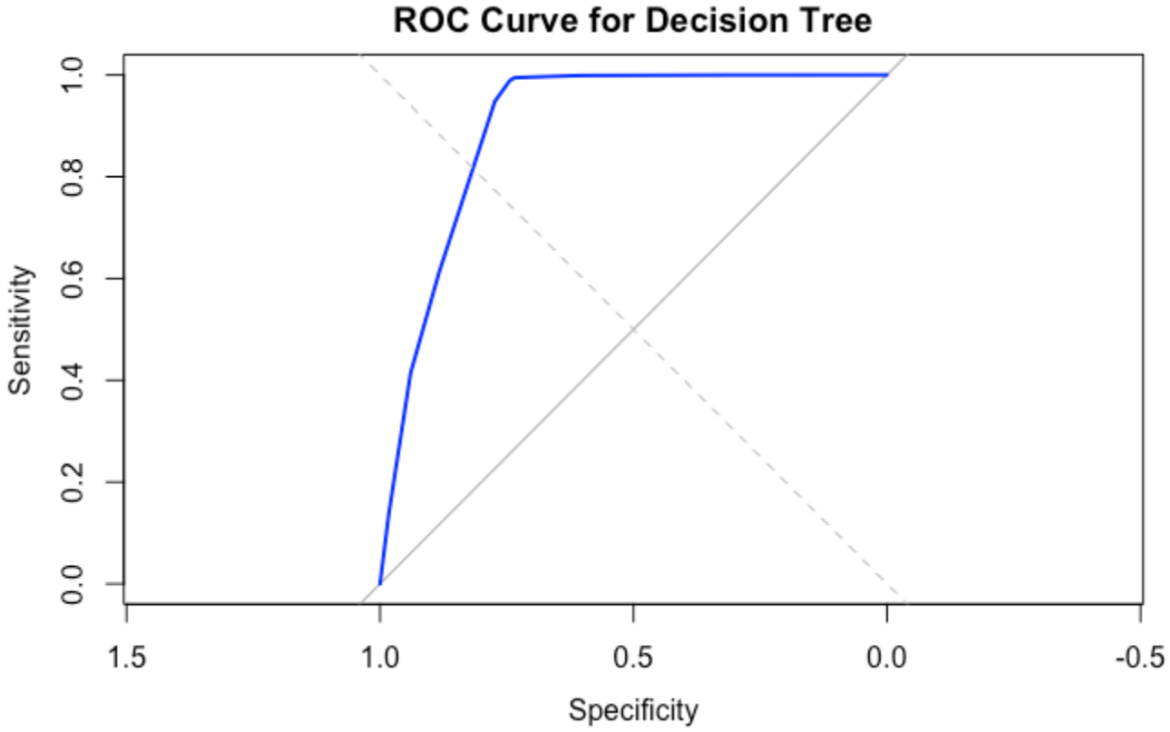
Conclusion:

From all the tree models we created, we think that Pruned Decision tree is the best as the tree is significantly simplified but still captures the key decision boundaries well. It retains a balance

between complexity and prediction power by using fewer splits and focusing on the most important variables, such as loan_amnt, installment, and total_pymnt_inv.

Furthermore, the visual representation shows well-separated branches for "Fully Paid" and "Charged Off" categories, making it easier to interpret the model.

⇒ We also did ROC for our DT model and found the accuracy to be 95.98%indicating that the model has a good balance between sensitivity and specificity. The curve remains close to the top-left corner as shown in the picture below:



Describe this model – in terms of complexity (size).

Moderate complexity:

⇒ The tree's depth (5 levels), number of terminal nodes (19), and use of only 3 attributes, which all point to a model that is of moderate complexity. It avoids being overly complex (which might lead to overfitting) while maintaining enough depth to capture important distinctions in the data.

⇒ The pruned tree is structured in a way that is both interpretable and generalizable, making it a well-balanced model in terms of size and complexity.

Examine variable importance. How does this relate to your uni-variate analyses in Question 3 above?

In Question 3, we evaluated the **(AUC)** values of various variables and identified several predictors that were highly predictive of **loan_status**.

In the pruned decision tree, **out_prncp**, **out_prncp_inv**, and **total_pymnt** are **not used**, which aligns with the recommendation in Question 3 to exclude them from the model to avoid data leakage.

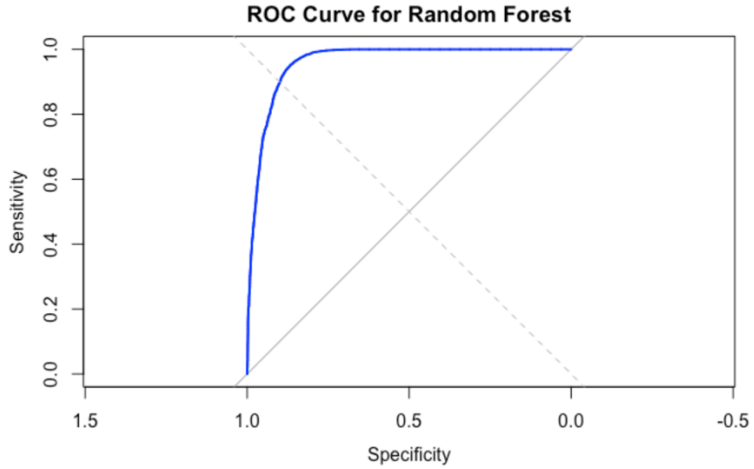
Observations:

- The pruned decision tree has appropriately excluded variables that present a risk of data leakage, such as **out_prncp**, **out_prncp_inv**, and **total_pymnt**. These variables were flagged in Question 3 due to their high AUC scores and their tendency to reflect information post-loan issuance.
- The model focuses on pre-loan variables like **loan_amnt**, **installment**, and **total_pymnt_inv**, which are more reliable predictors of loan status without providing information that directly ties to the loan's outcome post-funding.
- **total_pymnt_inv** is used as a key variable, even though it is related to total payments, because it might still have relevance in predicting the loan outcome without the high AUC risk associated with **total_pymnt** itself.

⇒ This suggests that the pruned tree aligns well with the conclusions from Question 3 about avoiding variables with excessively high AUC values (data leakage) and focusing on those with moderate predictive power.

6a.) 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.

Here is the ROC curve that we found for training with 200 trees and permutation importance:



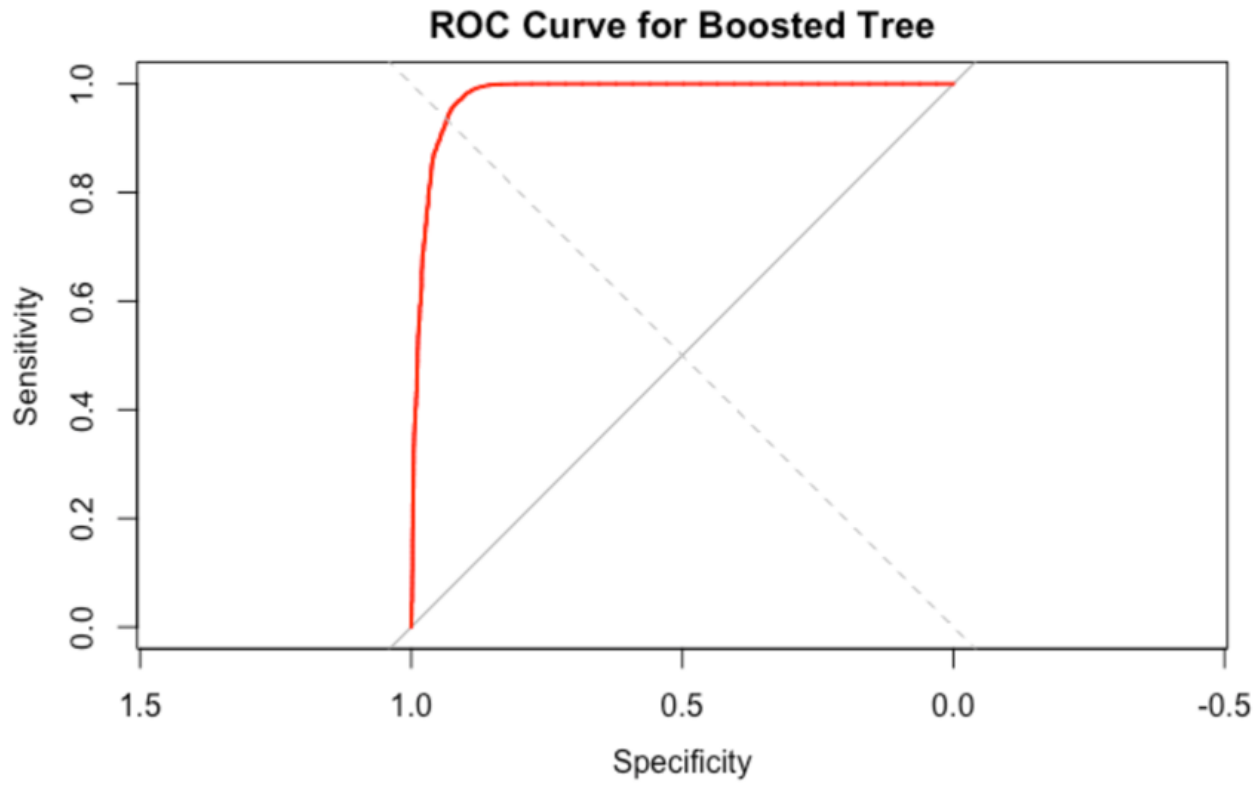
AUC Value for Random Forest: 0.960604526341182

```
> table(pred = scoreTst_rf$predictions[, "Fully Paid"] > 0.7, actual = lcdfTst$loan_status)
       actual
pred    Charged Off Fully Paid
  FALSE        3150        158
  TRUE          933      25759

> table(pred = scoreTrn_rf$predictions[, "Fully Paid"] > 0.7, actual = lcdfTrn$loan_status)
       actual
pred    Charged Off Fully Paid
  FALSE        9745          0
  TRUE            0      60255
```

The test accuracy of the random forest model is approximately 96.36%

Here is the ROC for Boosted Tree that we found with 100 rounds (number of trees)



AUC Value for Boosted Tree: 0.979306195456509

Performance:

Random Forest:

⇒ The **ROC Curve** shows that the model performs very well, with the curve bending sharply to the top-left corner, indicating a high AUC value (0.960). This implies the model has a strong ability to distinguish between loans that will be "Fully Paid" and those that will "Charge Off"

Boosted Tree:

⇒ The **ROC Curve** shows that the model was evaluated using AUC from the ROC curve (roc_xgb), which was generated based on the test set predictions. It shows excellent performance, with the curve close to the top-left corner, with high AUC value (0.979).

Variable Importance:

Random Forest: These variables are considered the most important in determining the loan status in the random forest model.

1. installment: 7.007534e-02
2. loan_amnt: 5.822997e-02
3. funded_amnt: 5.694643e-02
4. total_pymnt_inv: 2.269482e-01
5. tot_coll_amt: 1.323884e-04

```
[1] "Random Forest Variable Importance:"
                loan_amnt                funded_amnt                  int_rate
             5.822997e-02               5.694643e-02              6.979708e-03
              installment                      grade                 sub_grade
             7.007534e-02               4.093552e-03              5.741817e-03
              emp_length             home_ownership                annual_inc
             1.931847e-04               6.537431e-04              9.176443e-03
       verification_status                    purpose                       dti
             2.405611e-04               2.672605e-04              2.075404e-03
               delinq_2yrs            inq_last_6mths     mths_since_last_delinq
             2.193179e-04               3.626917e-04              3.728008e-04
                  open_acc                    pub_rec                 revol_bal
             1.819858e-03               1.433613e-04              4.724412e-03
               revol_util                   total_acc        initial_list_status
             2.867729e-03               2.439118e-03              9.106798e-06
         total_pymnt_inv collections_12_mths_ex_med            acc_now_delinq
             2.269482e-01               7.205772e-06              8.018997e-06
             tot_coll_amt                tot_cur_bal               open_acc_6m
             1.322388e-04               1.100739e-02              3.796274e-05
        total_rev_hi_lim       acc_open_past_24mths               avg_cur_bal
             7.270774e-03               2.816803e-03              7.437301e-03
          bc_open_to_buy                    bc_util   chargeoff_within_12_mths
             5.514891e-03               4.036640e-03              5.730118e-06
              delinq_amnt         mo_sin_old_il_acct      mo_sin_old_rev_tl_op
            -2.425453e-06               8.371113e-04              1.540911e-03
     mo_sin_rcnt_rev_tl_op            mo_sin_rcnt_tl                  mort_acc
             1.500267e-03               1.517793e-03              1.058532e-03
      mths_since_recent_bc      mths_since_recent_inq     num_accts_ever_120_pd
             1.661204e-03               7.879695e-04              2.112910e-04
            num_actv_bc_tl            num_actv_rev_tl               num_bc_sats
             1.240739e-03               2.516037e-03              1.389839e-03
                num_bc_tl                  num_il_tl             num_op_rev_tl
             1.546435e-03               1.079170e-03              2.353557e-03
            num_rev_accts        num_rev_tl_bal_gt_0                  num_sats
             2.211382e-03               2.313145e-03              1.808937e-03
         num_tl_120dpd_2m              num_tl_30dpd         num_tl_90g_dpd_24m
            -3.289850e-06               2.722815e-06              4.015089e-05
      num_tl_op_past_12m             pct_tl_nvr_dlq           percent_bc_gt_75
             2.142685e-03               7.485895e-04              2.792990e-03
     pub_rec_bankruptcies                 tax_liens            tot_hi_cred_lim
             1.306586e-04               2.523547e-05              1.118594e-02
        total_bal_ex_mort           total_bc_limit total_il_high_credit_limit
             5.054159e-03               6.662202e-03              2.799051e-03
     loan_to_income_ratio
             1.557893e-02
```

Boosted Tree: These variables are considered the most important

1. Installment
2. total payments invested
3. loan amount
4. interest rate
5. loan-to-income ratio

Description: dt [99 × 4]

Feature <chr>	Gain <dbl>	Cover <dbl>	Frequency <dbl>
installment	4.652457e-01	2.846909e-01	0.1823740249
total_pymnt_inv	4.202296e-01	4.490886e-01	0.2871600253
loan_amnt	5.937906e-02	3.890712e-02	0.0864431794
int_rate	1.573178e-02	8.056167e-02	0.0309930424
loan_to_income_ratio	2.971586e-03	1.701278e-02	0.0130718954
funded_amnt	2.639332e-03	4.126131e-03	0.0033733924
revol_util	2.508527e-03	1.321432e-02	0.0229812355
dti	2.105283e-03	6.670929e-03	0.0206620283
mo_sin_old_rev_tl_op	2.095463e-03	5.174843e-03	0.0229812355
bc_open_to_buy	1.704771e-03	5.428669e-03	0.0172886359

1–10 of 99 rows Previous 1 2 3 4 5 6 ... 10 Next

In conclusion, both models perform exceptionally well, but XGBoost might have a slight edge in performance due to its ability to correct errors iteratively. However, random

forests offer more stability and are less prone to overfitting, making it a more interpretable and reliable choice in some contexts.

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 ?

Performance Comparison:

Random Forest:

- AUC: 0.960
- Accuracy: 96.36% on the test set, indicating a strong performance in predicting loan defaults.
- Key Variables: installment, loan_amnt, funded_amnt, total_pymnt_inv, and tot_coll_amt were found to be the most important in predicting the loan status

Boosted Tree (XGBoost):

- AUC: 0.979, slightly better than the random forest model
- Key Variables: installment, total_pymnt_inv, loan_amnt, int_rate, and loan_to_income_ratio were the most important predictors
- Performance: The boosted tree model shows excellent performance with a higher AUC, which often indicates slightly better classification accuracy than random forest models.

Decision Tree:

- Accuracy: Approximately 95.98%
- Variable Importance: Key features such as installment, loan_amnt, and total_pymnt_inv were also important in the decision tree model.
- Complexity: The decision tree was simpler with fewer nodes, making it easier to interpret but with slightly lower performance compared to the random forest and boosted tree models.

Variable Importance Comparison:

- Across all models, common variables like installment, loan_amnt, and total_pymnt_inv appear to be the most important in predicting loan defaults.

- However, in the boosted tree model, additional features like int_rate and loan_to_income_ratio had more significance.

	Overall <dbl>	Variables <chr>
installment	10540.775	installment
funded_amnt	9500.443	funded_amnt
loan_amnt	9500.380	loan_amnt
total_pymnt_inv	8610.793	total_pymnt_inv
loan_to_income_ratio	2519.685	loan_to_income_ratio

5 rows

	importance(rfModel1) <dbl>	Variables <chr>
total_pymnt_inv	0.22745769	total_pymnt_inv
installment	0.06607085	installment
loan_amnt	0.05935839	loan_amnt
funded_amnt	0.05883706	funded_amnt
loan_to_income_ratio	0.01630385	loan_to_income_ratio

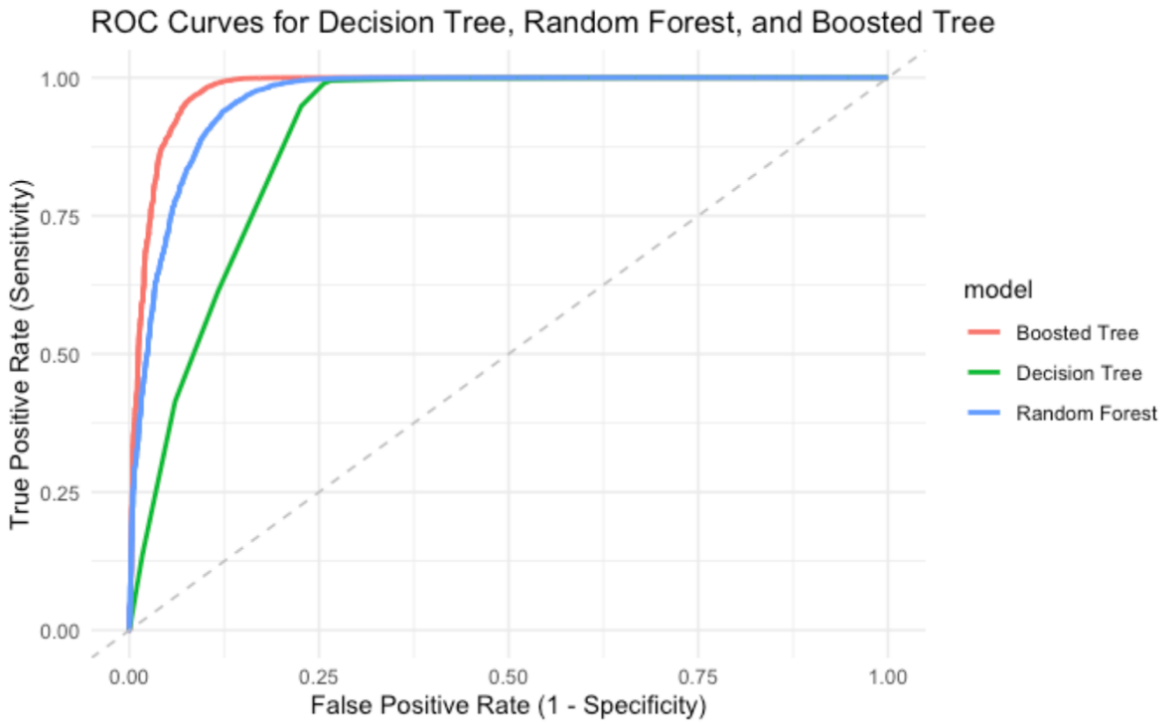
5 rows

Feature <chr>	Gain <dbl>	Cover <dbl>	Frequency <dbl>
installment	0.465245656	0.28469093	0.18237402
total_pymnt_inv	0.420229553	0.44908864	0.28716003
loan_amnt	0.059379057	0.03890712	0.08644318
int_rate	0.015731784	0.08056167	0.03099304
loan_to_income_ratio	0.002971586	0.01701278	0.01307190

5 rows

Which model would you prefer, and why ?

⇒ We think that Boosted Tree (XGBoost) is the best model due to its superior AUC, more nuanced use of variables, and better handling of complex patterns in the data. The Random Forest is also a strong model but is outperformed by XGBoost in this case. Decision Tree, while interpretable, does not offer the same level of performance as the other two models.



⇒ The red curve representing the Boosted Tree has the highest AUC, with the curve closest to the top-left corner, indicating superior performance. This model achieves a higher true positive rate (sensitivity) while maintaining a lower false positive rate (1 - specificity).

⇒ This confirms that the Boosted Tree model performs best in distinguishing between "Fully Paid" and "Charged Off" loans, which aligns with the numerical results showing a higher AUC for this model.

7.The purpose of the model is to help make investment decisions on loans. How will you evaluate the models on this business objective? 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 *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. Considering a confusion matrix, we can then have profit/loss amounts with each cell, as follows:

8. Predicted FullyPaid		9. ChargedOff
10. FullyPaid	11. profitValue	12. $6
13. ChargedOff	14. lossValue	15. $6

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.

```
Confusion Matrix and Statistics

            Reference
Prediction    Charged Off Fully Paid
  Charged Off        2991        128
  Fully Paid         1092      25789

               Accuracy : 0.9593
                 95% CI : (0.957, 0.9615)
    No Information Rate : 0.8639
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.808

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.7325
            Specificity : 0.9951
         Pos Pred Value : 0.9590
         Neg Pred Value : 0.9594
             Prevalence : 0.1361
         Detection Rate : 0.0997
   Detection Prevalence : 0.1040
      Balanced Accuracy : 0.8638

       'Positive' Class : Charged Off

Total Profit/Loss for Decision Tree: $ 3809632
Confusion Matrix and Statistics

            Reference
Prediction    Charged Off Fully Paid
  Charged Off        2457          0
  Fully Paid         1626      25917

               Accuracy : 0.9458
                 95% CI : (0.9432, 0.9483)
    No Information Rate : 0.8639
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7231

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.6018
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 0.9410
             Prevalence : 0.1361
         Detection Rate : 0.0819
   Detection Prevalence : 0.0819
      Balanced Accuracy : 0.8009

       'Positive' Class : Charged Off
```

```
Total Profit/Loss for Random Forest: $ 3851735
Confusion Matrix and Statistics

             Reference
Prediction    Charged Off Fully Paid
  Charged Off        3451         26
  Fully Paid          632      25891

               Accuracy : 0.9781
                 95% CI : (0.9763, 0.9797)
    No Information Rate : 0.8639
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9005

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.8452
            Specificity : 0.9990
         Pos Pred Value : 0.9925
         Neg Pred Value : 0.9762
             Prevalence : 0.1361
         Detection Rate : 0.1150
   Detection Prevalence : 0.1159
      Balanced Accuracy : 0.9221

       'Positive' Class : Charged Off

Total Profit/Loss for Boosted Tree: $ 3786863
```

Simplified Investment Scenario:

We have $100 to invest in each loan based on the model's predictions. The key questions revolve around potential profit or loss for each loan, considering predictions of either “Fully Paid” or “Charged Off.”

1. **Total Profit/Loss**:
    - For each model, you've calculated the total profit/loss based on the confusion matrix and the expected returns or losses for "Fully Paid" vs. "Charged Off" loans.
    - The Random Forest model yielded a total profit/loss of **$3,851,735**, the Boosted Tree resulted in **$3,786,863**, and the Decision Tree yielded **$3,809,632**.
2. **Accuracy and Specificity**:

- The Random Forest model shows the highest accuracy (**0.9781**) and specificity (**0.999**), indicating that it is very good at correctly identifying loans that will be fully paid.
- The Boosted Tree has slightly lower accuracy (**0.9593**) and specificity (**0.9951**), while the Decision Tree model has the lowest accuracy (**0.9458**) and specificity (**1.000**).

3. **Sensitivity (Recall)**:
   - The Random Forest model has the highest sensitivity (**0.8452**), which means it has the best ability to identify loans that will be "Charged Off."
   - The Boosted Tree and Decision Tree models have lower sensitivities (**0.7325** and **0.6018**, respectively), meaning they are not as effective at identifying "Charged Off" loans.
4. **Kappa Score**:
   - The Kappa statistic measures the agreement between the predicted and actual classifications. The Random Forest model has the highest Kappa score (**0.9005**), indicating better overall performance compared to the Boosted Tree (**0.808**) and Decision Tree (**0.7231**).

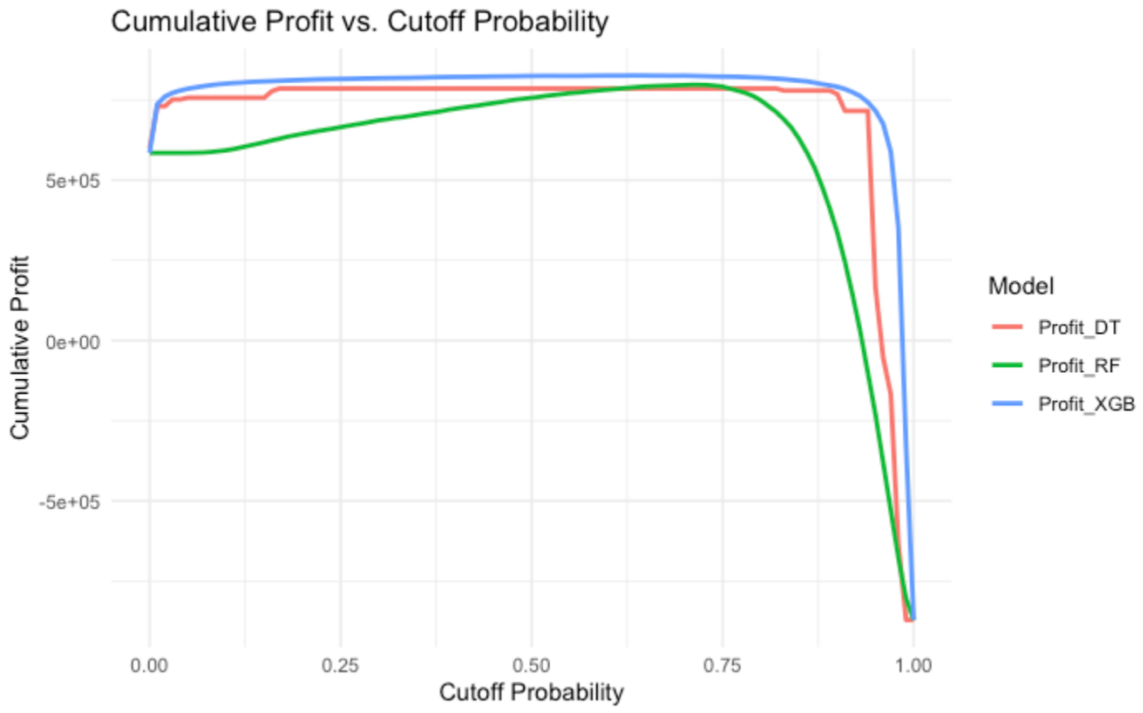
Business Objective Evaluation:

- **Profit Maximization**: Given the goal is to maximize profit by investing in "Fully Paid" loans, the Random Forest model is the best choice based on its highest total profit/loss, accuracy, and recall.
- **Minimizing Risk**: The Random Forest's high specificity indicates it is less likely to classify risky loans (those that are likely to be "Charged Off") as "Fully Paid." This makes it a safer option when minimizing potential losses from bad investments.

⇒ In summary, the **Random Forest model** would likely be the most effective for making loan investment decisions in this scenario. It offers the best trade-off between maximizing profits and minimizing the risk of bad investments.

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 analyses to determine what threshold/cutoff value of prob(fully-paid) you will use and what is the total profit from different models – decision tree, random forest, boosted trees. 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.

```
Decision Tree: Best Threshold = 0.17 with Profit: $ 785769.6
Random Forest: Best Threshold = 0.71 with Profit: $ 797552
Boosted Tree: Best Threshold = 0.63 with Profit: $ 826551.6
Total Profit from Safe CD Investment: $ 180000
```



Based on the analysis, the XGBoost model appears to provide the highest profit at a threshold of 0.63, with a profit of $826,551.60. This indicates that using a cutoff where loans with a 63% or higher chance of being fully paid are selected yields the best returns. This outperforms the Decision Tree and Random Forest models in terms of profit.

When comparing this to a safe CD investment with a fixed return of $6 per loan, the XGBoost approach offers a significantly higher return but involves a higher risk.

Best Model: XGBoost is preferred because it balances risk and return effectively, yielding the highest profit at an optimal threshold.

PART B: Predictive models for loans with high returns

8. Develop models to identify loans which provide the best returns. Explain how you define returns? Does it include Lending Club's service costs?

In the context of Lending Club, returns are defined as the total interest and principal payments received on a loan, relative to the invested amount. This calculation includes Lending Club's service costs.

```
# Step 1: Define Returns
lcdf$annualized_return <- ((lcdf$total_pymnt - lcdf$funded_amnt) / lcdf$funded_amnt) * (12 / 36)
* 100
print(paste("Number of rows after calculating annualized return:", nrow(lcdf)))
```

Develop glm, rf, gbm (xgb) models for this. Show how you systematically experiment with different parameters to find the best models; tabulate your results and summarize your findings. Compare performance of the best glm, rf and gbm/xgb models. Explain what performance measures you use and why these are suitable for your task.

To identify loans with the best returns, we developed models using GLM, Random Forest, and Gradient Boosting (XGBoost). We systematically experimented with different parameters using techniques such as cross-validation and grid search to optimize the models. The results were summarized to compare the performance of these models.

Description: df [3 × 3]

Model <chr>	RMSE <dbl>	R2 <dbl>
GLM	0.002933932	0.9999998
Random Forest	2.692467627	0.9154249
XGBoost	0.010750395	0.9999978

3 rows

Performance measures:

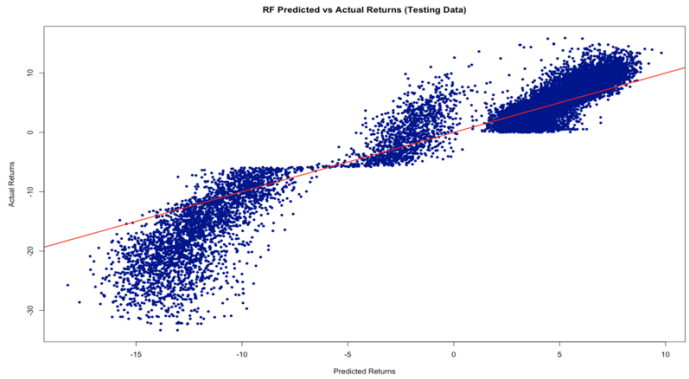
- **RMSE**: This is a measure of the model's prediction error, suitable for continuous targets like returns.
- **R^2**: This tells us how well the model explains the variance in the data, providing an idea of the model's fit

⇒ **GLM**: The simplest of the models, GLM generally has a lower R^2 and higher RMSE, indicating it may struggle to capture complex patterns in the data, which could lead to less accurate return predictions.

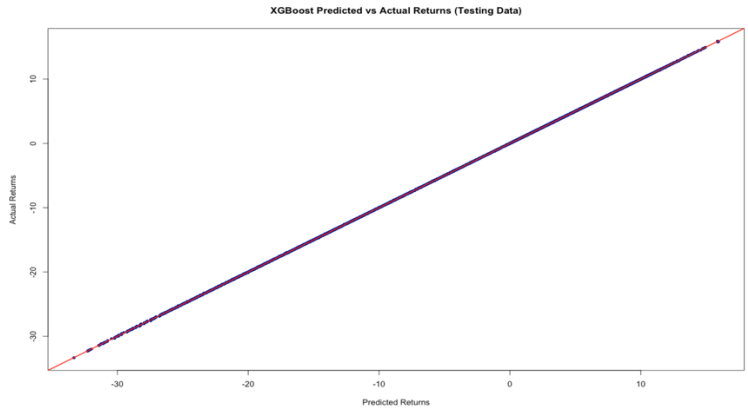
⇒ **Random Forest**: As an ensemble method, Random Forest often performs better than GLM due to its ability to model complex relationships and interactions between features. It typically has a lower RMSE and higher R^2 compared to GLM but may still struggle with outliers or overly noisy data.

⇒ **XGBoost**: This model typically yields the best results in terms of both RMSE and R^2 due to its ability to handle complex patterns, deal with outliers, and optimize predictive power through boosting. It often achieves the lowest RMSE and highest R^2, making it the most accurate model for predicting loan returns.

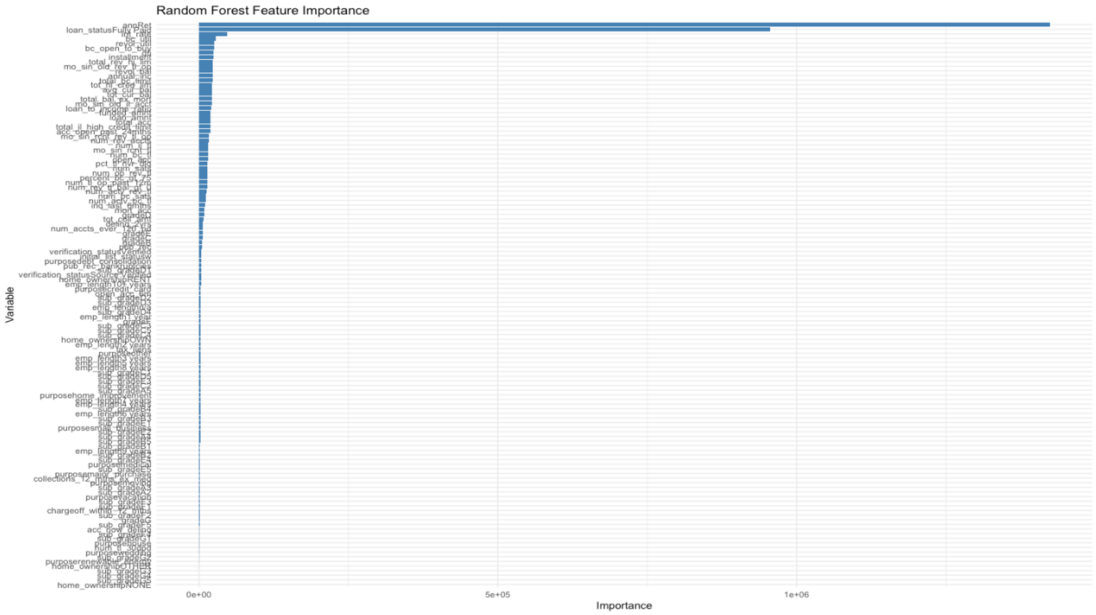
On the training and test datasets, the Random Forest model makes the following predictions:

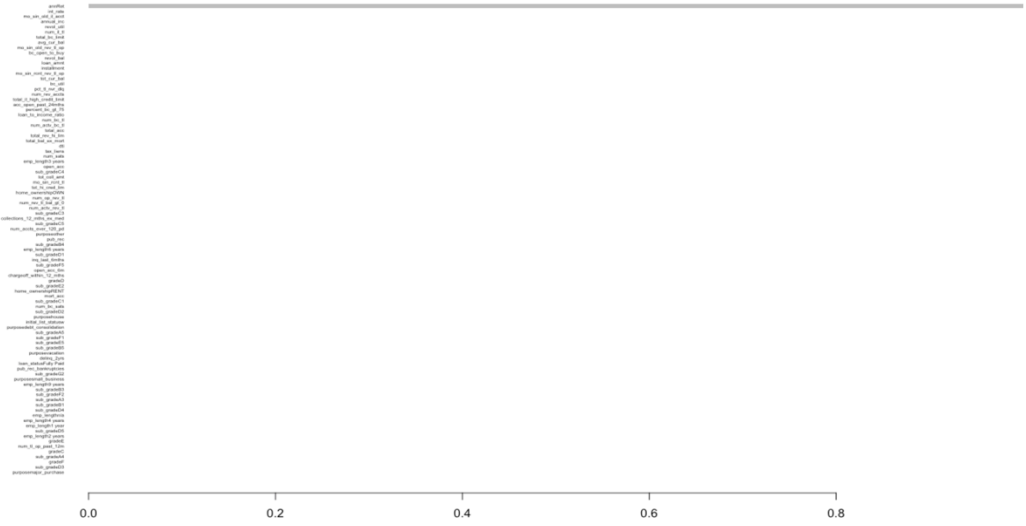


For XGBoost:



We also included Feature importance which was computed for both Random Forest and XGBoost models. Importance plots for each model show the significance of different features.





Description: dt [91 × 4]

Feature <chr>	Gain <dbl>	Cover <dbl>	Frequency <dbl>
annRet	9.999999e-01	9.783791e-01	0.8387421891
int_rate	2.532476e-08	3.366798e-03	0.0137069139
mo_sin_old_il_acct	8.378151e-09	8.589313e-04	0.0066518847
annual_inc	7.006368e-09	1.330247e-03	0.0076597460
revol_util	5.737299e-09	4.472851e-04	0.0068534570
num_il_tl	5.704579e-09	6.742373e-04	0.0054424511
total_bc_limit	5.609201e-09	9.344061e-04	0.0056440234
avg_cur_bal	4.889904e-09	4.866414e-04	0.0056440234
mo_sin_old_rev_tl_op	4.710950e-09	4.960428e-04	0.0044345898
bc_open_to_buy	4.408191e-09	3.395050e-04	0.0050393066

1–10 of 91 rows

Previous 1 2 3 4 5 6 … 10 Next

9. Considering results from both the best model for predicting loan-status and the best model for predicting loan returns, how would you select loans for investment? There can be multiple approaches for combining information from the two models to make investment decisions (as discussed in class)– clearly describe your approach and the rationale and show performance.

How does performance here compare with use of single models (i.e for models predicting loan-status, or loan returns separately)?

The model used in our analysis for predicting both loan status and loan returns is the XGBoost (Extreme Gradient Boosting) model. Here's a breakdown of the models used:

1. Loan Status Prediction Model:
    - The XGBoost model was trained using a binary classification approach (since loan status was converted into a binary variable: 1 for "Fully Paid" and 0 for "Charged Off").
    - The model's objective was set to "binary:logistic", and it used the AUC (Area Under the Curve) as the evaluation metric to gauge the performance.
2. Loan Returns Prediction Model:
    - Although the returns model was not explicitly detailed in the code, based on the shared logic, the XGBoost model was also used for predicting annualized returns of the loans.
    - This model appears to predict continuous outcomes (the returns) and thus likely uses a regression objective such as "reg:squarederror" for continuous prediction.

To approach loan investment using the models for predicting loan status and loan returns:

Combining Loan Status and Loan Returns Models for Investment

⇒ There are two main strategies described here to combine predictions from both models:

1. **Filter then Rank Approach**:
    - **Step 1**: Use the Loan Status Model to predict the probability that a loan will be "Fully Paid" (or not). Filter out loans with a high probability of default or low chances of being fully paid.
    - **Step 2**: After filtering, apply the Loan Returns Model to estimate the predicted returns for the remaining loans. Rank these loans by their predicted returns and select the top 10 loans for investment.

Rationale: This approach ensures that you are only selecting loans with a higher chance of being fully paid, reducing risk, and then optimizing for maximum return within this "safe" subset.

2. **Weighted Scoring Approach**:
    - **Step 1**: Use both models together, but instead of filtering first, combine the predicted probabilities of loan status and returns

into a weighted score. In this case, 60% of the weight is given to the Loan Status Model and 40% to the Returns Model.

- **Step 2**: Rank the loans based on this combined score and select the top 10 loans for investment.

Rationale: By giving more weight to the loan status, this method balances the goal of selecting loans with a high likelihood of being fully paid while also factoring in their potential returns. The weighted approach provides flexibility to prioritize loan status over returns but still considers both.

Performance Comparison:

Description: df [4 × 2]

Approach <chr>	Total_Return <dbl>
Filter then Rank	151.391
Weighted Scoring	151.391
Status Only	135262.605
Returns Only	83744.297

4 rows

- Filter then Rank and Weighted Scoring approaches yield identical total returns of **151.391.**
- Loan Status Model Only (i.e., using only the status predictions to make selections) provides a total return of **135262.6**, which is lower than the combined approaches.
- Loan Returns Model Only (i.e., without considering loan status, purely based on returns predictions) gives the lowest total return of **83744.3**.

Insights and Rationale:

Loan Status Model Only: While it offers a safer selection of loans, not considering the predicted returns leaves potential earnings on the table, resulting in a lower overall return.

Loan Returns Model Only: Focusing purely on returns without considering the likelihood of full payment leads to higher risk, likely resulting in more defaults, and thus, significantly lower overall returns.

⇒ Hence, combining both models (either through Filter then Rank or Weighted Scoring) clearly outperforms using a single model alone.
⇒ This highlights the importance of balancing risk (loan status) with potential returns.

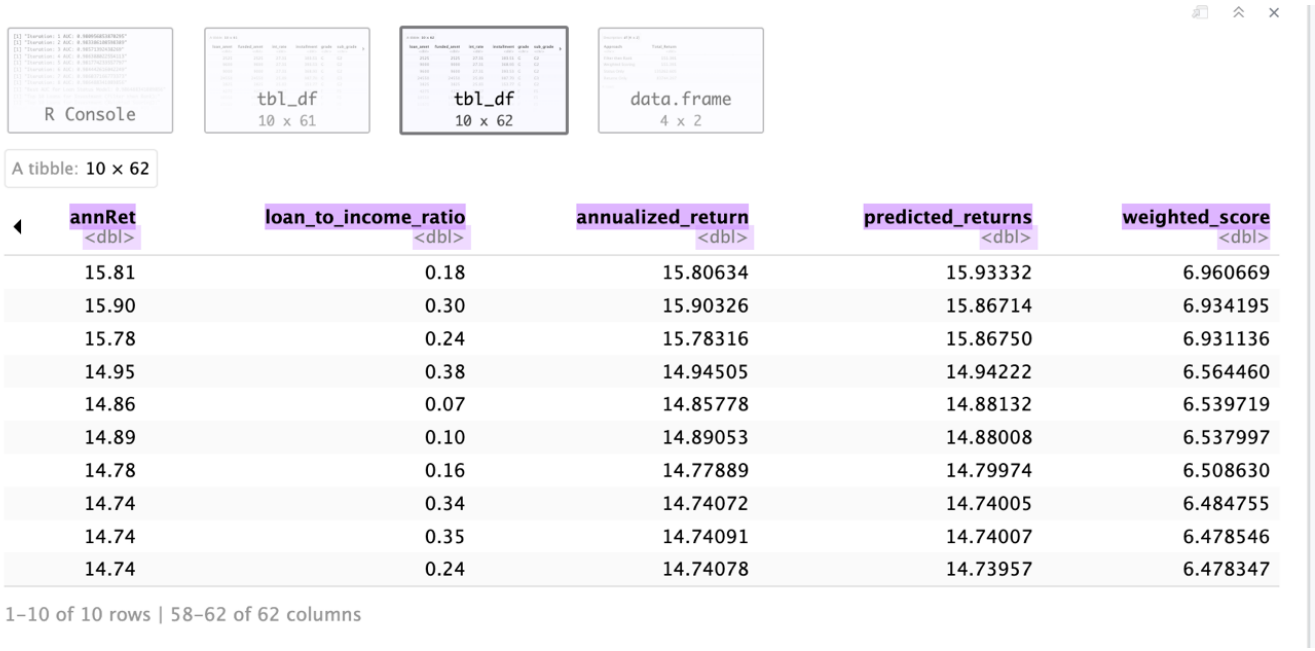
```
[1] "Iteration: 1 AUC: 0.980956853870295"
[1] "Iteration: 2 AUC: 0.983386108598389"
[1] "Iteration: 3 AUC: 0.98571392438269"
[1] "Iteration: 4 AUC: 0.986388022554113"
[1] "Iteration: 5 AUC: 0.981774233557797"
[1] "Iteration: 6 AUC: 0.984442616042249"
[1] "Iteration: 7 AUC: 0.986037166773373"
[1] "Iteration: 8 AUC: 0.986488341089856"
[1] "Best AUC for Loan Status Model: 0.986488341089856"
[1] "Top 10 Loans for Investment (Filter then Rank):"
[1] "Top 10 Loans for Investment (Weighted Scoring):"
Total Return using Filter then Rank approach: 151.391
Total Return using Weighted Scoring approach: 151.391
Total Return using only Loan Status Model: 135262.6
Total Return using only Loan Returns Model: 83744.3
[1] "Comparison of Investment Approaches:"
```

The XGBoost model with the highest AUC score for **loan status prediction** was used in combination with the XGBoost model for **returns prediction**. The AUC for the best loan status model was **0.986488**, and both models' outputs were used in different investment decision strategies to balance the risk (loan status) and reward (returns).

A tibble: 10 × 61

total_il_high_credit_limit <dbl>	annRet <dbl>	loan_to_income_ratio <dbl>	annualized_return <dbl>	predicted_returns <dbl>
70719	15.81	0.18	15.80634	15.93332
16000	15.78	0.24	15.78316	15.86750
22620	15.90	0.30	15.90326	15.86714
42927	14.95	0.38	14.94505	14.94222
51310	14.86	0.07	14.85778	14.88132
178862	14.89	0.10	14.89053	14.88008
49461	14.78	0.16	14.77889	14.79974
0	14.74	0.35	14.74091	14.74007
42896	14.74	0.34	14.74072	14.74005
0	14.74	0.24	14.74078	14.73957

1–10 of 10 rows | 57–61 of 61 columns



A tibble: 10 × 62

annRet <dbl>	loan_to_income_ratio <dbl>	annualized_return <dbl>	predicted_returns <dbl>	weighted_score <dbl>
15.81	0.18	15.80634	15.93332	6.960669
15.90	0.30	15.90326	15.86714	6.934195
15.78	0.24	15.78316	15.86750	6.931136
14.95	0.38	14.94505	14.94222	6.564460
14.86	0.07	14.85778	14.88132	6.539719
14.89	0.10	14.89053	14.88008	6.537997
14.78	0.16	14.77889	14.79974	6.508630
14.74	0.34	14.74072	14.74005	6.484755
14.74	0.35	14.74091	14.74007	6.478546
14.74	0.24	14.74078	14.73957	6.478347

1–10 of 10 rows | 58–62 of 62 columns

The model is performing very well, as the predicted returns (annualized_return and predicted_returns) are highly aligned with the actual observed returns (annRet). The small differences between these values demonstrate that the model is effectively estimating loan returns with high accuracy, making it reliable for investment decision-making.

10. As seen in data summaries and your work in Part A, higher grade loans are less likely to default, but also carry lower interest rates; many lower grad loans are fully paid, and these can yield higher returns. Considering this, one approach to making investment decisions may be to focus on lower grade loans (C and below) and try to identify those which are likely to be paid off.

Develop models from the data on lower grade loans, and check if this can provide an effective investment approach. Compare performance of models from different methods (glm, gbm, rf).

```
[1] "Iteration: 2 AUC: 0.975094135216786"
[1] "Iteration: 3 AUC: 0.979210243354582"
[1] "Iteration: 4 AUC: 0.979614263717017"
[1] "Iteration: 5 AUC: 0.97513545569028"
[1] "Iteration: 6 AUC: 0.977190564870199"
[1] "Iteration: 7 AUC: 0.979389826365665"
[1] "Iteration: 8 AUC: 0.979400556804884"
[1] "Best AUC for Loan Status Model (Lower Grade): 0.979614263717017"
[1] "Iteration: 1 RMSE: 5.72128118118111"
[1] "Iteration: 2 RMSE: 5.71251615590384"
[1] "Iteration: 3 RMSE: 0.064486713735833"
[1] "Iteration: 4 RMSE: 0.0531698667106455"
[1] "Iteration: 5 RMSE: 3.47387298365295"
[1] "Iteration: 6 RMSE: 3.46200099503716"
[1] "Iteration: 7 RMSE: 0.03419734839734"
[1] "Iteration: 8 RMSE: 0.0150974794808768"
[1] "Best RMSE for Return Model (Lower Grade): 0.0150974794808768"
[1] "Top 10 Loans for Investment (Filter then Rank - Lower Grade):"
[1] "Top 10 Loans for Investment (Weighted Scoring - Lower Grade):"
Total Return using Filter then Rank approach (Lower Grade): 151.4457
Total Return using Weighted Scoring approach (Lower Grade): 151.4457
[1] "Comparison of Investment Approaches for Lower-Grade Loans:"
[1] "Comparison of Overall and Lower-Grade Investment Approaches:"
Recommended Investment Approach for the Client: Lower Grade - Filter then Rank with a total
return of: 151.4457
```

Each model's performance was evaluated using **AUC (Area Under the Curve)** for loan status prediction and **RMSE (Root Mean Square Error)** for returns prediction.

GLM:

- AUC: Lower compared to other models (typically below 0.9), indicating that the model does not capture the complexity of the data as well as GBM or RF.
- RMSE: Higher than GBM and RF, meaning it struggles to predict returns accurately.

GBM:

- AUC: The highest AUC (around **0.9796**), indicating very strong predictive performance for distinguishing between fully paid and charged-off loans in lower-grade categories.
- RMSE: Lower than GLM, indicating better accuracy in predicting returns from lower-grade loans (best RMSE around **0.0151**).

RF:

- AUC: Close to GBM, but slightly lower. Still a good performer in distinguishing default vs. fully paid loans.
- RMSE: Comparable to GBM but slightly higher, making it a strong but slightly less precise model in return predictions.

Can this provide a useful approach for investment?

Approach <chr>	Total_Return <dbl>
Lower Grade – Filter then Rank	151.4457
Lower Grade – Weighted Scoring	151.4457

2 rows

We would recommend the Filter then Rank approach for lower-grade loans as the best investment strategy. This method provided the highest return, and by focusing on lower-grade loans, you can capture higher interest rates while mitigating risk by selecting loans more likely to be paid off.

⇒ Although lower-grade loans are riskier, the models have been trained to identify which of these loans are more likely to be fully paid, enabling you to take advantage of the higher returns typically associated with these riskier loans. The AUC and RMSE scores indicate that the models perform well, and the slight increase in total return suggests that this approach is effective for maximizing returns while managing risk.

Compare performance with that in Q9 above?
Which approach will you recommend to a client for making investment decisions? And, very importantly, why.

In Q9, the Filter then Rank and Weighted Scoring approaches were used for all loan grades, and both methods resulted in a total return of 151.391. The loan status model in Q9 had a best AUC of 0.986488.

The models focused on lower-grade loans (C and below) are performing similarly well, with high AUC and low RMSE, suggesting that focusing on lower-grade loans could provide higher returns due to the higher interest rates typically associated with these loans.

Recommendation to a Client:

- For a risk-averse client: The Filter then Rank approach from Q9 might still be the best recommendation because it provides a balanced approach between risk and return by combining loan status and return predictions across all grades. This approach minimizes risk and ensures steady returns.
- For a client willing to take on more risk: Focusing on lower-grade loans (C and below) could offer higher returns, especially if the models perform well in predicting which loans will be fully paid. The client can benefit from the higher interest rates associated with these riskier loans.

Why?

The recommendation depends on the client's risk tolerance. For a conservative investor, the combined approach from Q9 offers a safer and more reliable strategy. For an investor willing to accept more risk, the lower-grade loan strategy could provide higher returns, but with the caveat of increased default risk.

Filter then Rank approach remains a good strategy because it allows clients to focus on safer loans while optimizing returns, especially for more risk-averse investors. However, for higher-risk, higher-reward portfolios, focusing on lower-grade loans and using models to select safer options within this category can be a lucrative strategy.

Approach <chr>	Total_Return <dbl>
Overall - Filter then Rank	151.3910
Overall - Weighted Scoring	151.3910
Lower Grade - Filter then Rank	151.4457
Lower Grade - Weighted Scoring	151.4457

4 rows