Executive Summary

Emma Wang (06648801), wangj422@wfu.edu, 12/09/2022

Problem Statement

This is a report produced for a major financial institution. Our main task is to build machine learning models to predict the "loan status", more specifically, we need to predict which loans are more likely to default. Due to the requirement of this task, our model needs to be both explainable and possess powerful predictability.

Key Findings

- If the most recent month LC pulled credit for this loan is Sep-2016, the possibility of default will be much higher. It is also the most powerful predictor.
- Among all the false negatives that we found, many of them have a very high annual income. Based on the results, these annual income records are more likely to be fake.
- 60-month term loan, a very low last payment amount, and a higher interest rate will potentially result in a higher probability of default.
- Inactive accounts, i.e., the ones with a long time period without payment, will be more likely to be involved in default.

Result of the model

We built and trained three powerful machine learning models and their performance are summarized as follows. Typically, we will investigate the accuracy, AUC, precision, and recall. AUC is the metric that represents how well the model can classify true and false, a higher AUC means this model can distinguish exciting projects better. Accuracy represents that among all records, how many were identified correctly. The recall represents among all the exciting projects, how many of those are successfully found by our model. Finally, precision represents that among all the records that we predicted as true, how many of them are actually true.

Therefore, based on the table below, we will choose the XGBoost model as our final model as it has the highest AUC, accuracy, precision, and recall. It can predict almost 98% of the total records correctly. Of all the loans that it predicted to default,

95% of them will default eventually. Among all default loans, it can successfully identify 91% of them.

	Accuracy	Log loss	AUC	Precision	Recall	F1
Training NN	0.9606	0.3607	0.9826	0.8950	0.8336	0.8632
Testing NN	0.9579	0.3625	0.9790	0.8920	0.8225	0.8575
Training XGB	1.0000	0.0012	1.0000	1.0000	1.0000	1.0000
Testing XGB	0.9786	0.0820	0.9937	0.9497	0.9088	0.9288
Training RF	1.0000	0.0499	1.0000	1.0000	1.0000	1.0000
Testing RF	0.9569	0.1524	0.9842	0.9401	0.7679	0.8453

Recommendations

- Operating at a 5% FNR threshold will bring us $1,083,911 potential savings if we use the XGBoost model to predict loan status. And we will keep the percentage of predicting a defaulted loan as current below 5%.
- Investigate annual income, especially those exceptionally high; investigate the accounts which suddenly implement activities after a long-time silence.
- Launch more financial products, with adopting a relatively lower interest rate and 36-month term, to reduce the probability of default.

Detailed Analysis

Emma Wang (06648801), wangj422@wfu.edu, 12/09/2022

Table of Contents

Files & Field Summary

Files Summary

The imported original file summary is shown below, from which we can get the most basic information about our training file and prediction file:

File Name	Record Count	Column Count	Numeric Columns	Character Columns
Loan_train.csv	29777	52	29	23
Loan_holdout.csv	12761	51	29	22

Fields Summary

The training dataset's categorical fields summary is shown below, from which we can get the number of missing values, the number of distinct values, and the complete rate, which is 1 minus the missing rate:

Name	Feature Type	# Missing	Complete Rate	# Distinct
term	Categorical	3	0.99989925	2
int_rate	Categorical	3	0.99989925	390
grade	Categorical	3	0.99989925	7
sub_grade	Categorical	3	0.99989925	35
emp_title	Categorical	1817	0.93897975	22143
emp_length	Categorical	3	0.99989925	12
home_ownership	Categorical	3	0.99989925	5
verification_status	Categorical	3	0.99989925	3
issue_d	Timestamp	3	0.99989925	55
loan_status	Categorical	0	1.00000000	2
pymnt_plan	Categorical	3	0.99989925	2
url	Categorical	3	0.99989925	29774
desc	Categorical	9432	0.68324546	20310
purpose	Categorical	3	0.99989925	14
title	Categorical	13	0.99956342	15200
zip_code	Categorical	3	0.99989925	819
addr_state	Categorical	3	0.99989925	50
earliest_cr_line	Timestamp	23	0.99922759	516
revol_util	Categorical	67	0.99774994	1094
last_pymnt_d	Timestamp	67	0.99774994	106
next_pymnt_d	Timestamp	27425	0.07898714	96
last_credit_pull_d	Timestamp	5	0.99983209	109
application_type	Categorical	3	0.99989925	1

The training dataset's numeric fields summary is shown below, from which we can get the number of missing values, some statistical results, and the complete rate, which is 1 minus the missing rate:

Name	Feature Type	# Missing	Complete Rate	Mean	Min	Max
id	ID	3	0.99	663006.18	54734	1077501
member_id	ID	3	0.99	823568.14	70473	1314167
loan_amnt	Numeric	3	0.99	11109.43	500	35000
funded_amnt	Numeric	3	0.99	10843.63	500	35000

funded_amnt_inv	Numeric	3	0.99	10149.65	0	35000
installment	Numeric	3	0.99	323.80	15.67	1305.19
annual_inc	Numeric	4	0.99	69201.23	2000	6000000
dti	Numeric	3	0.99	13.38	0	29.99
delinq_2yrs	Numeric	23	0.99	0.15	0	13
fico_range_low	Numeric	3	0.99	713.05	610	825
fico_range_high	Numeric	3	0.99	717.05	614	829
inq_last_6mths	Numeric	23	0.99	1.08	0	33
mths_since_last_delinq	Numeric	18907	0.36	34.71	0	120
mths_since_last_record	Numeric	27208	0.09	59.22	0	129
open_acc	Numeric	23	0.99	9.34	1	47
pub_rec	Numeric	23	0.99	0.05	0	5
revol_bal	Numeric	3	0.99	14310.00	0	1207359
total_acc	Numeric	23	0.99	22.08	1	81
out_prncp	Numeric	3	0.99	11.79	0	3126.61
out_prncp_inv	Numeric	3	0.99	11.76	0	3123.44
total_rec_late_fee	Numeric	3	0.99	1.50	0	180.20
last_pymnt_amnt	Numeric	3	0.99	2615.41	0	36115.20
collections_12_mths_ex_med	Numeric	104	0.99	0.00	0	0
policy_code	Numeric	3	0.99	1.00	1	1
acc_now_delinq	Numeric	23	0.99	0.00013	0	1
chargeoff_within_12_mths	Numeric	104	0.99	0.00	0	0
delinq_amnt	Numeric	23	0.99	0.20	0	6053
pub_rec_bankruptcies	Numeric	966	0.97	0.04	0	2
tax_liens	Numeric	79	0.99	0.00003	0	1

Data Cleaning & Preliminary Transformation

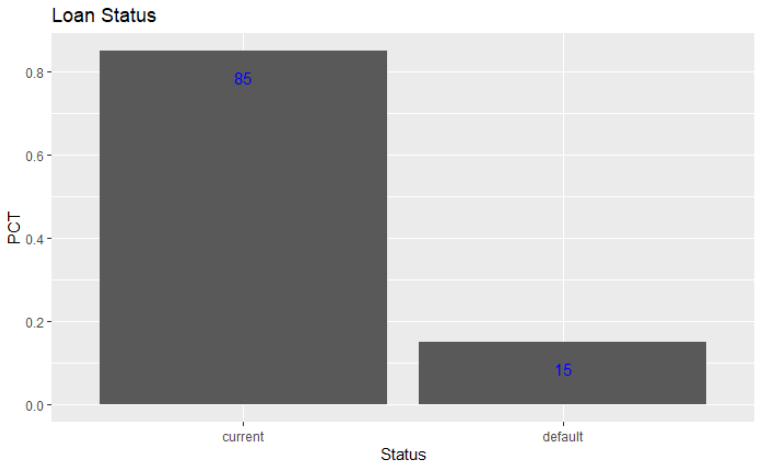
- **Exclude Some Variables**
  - Exclude variables with a missing rate above 30%, i.e., a complete rate below 70%.
  - Exclude variables with high cardinality (over 10,000 unique levels), especially the ones representing IDs.
  - Exclude variables with only one unique value, like application type and policy code.
- **Deal with Date Variables**
  1. Mutate “issue_d”, “earliest_cr_line”, “last_pymnt_d”, “last_credit_pull_d” as date-type variables.
  2. Calculate the time difference in weeks or years between the current date and those dates mentioned above.
  3. Name them as “issue_w”, “earliest_cr_line_y”, “last_pymnt_w”, “last_credit_pull_w”, respectively.
  4. Remove “issue_d”, “earliest_cr_line”, “last_pymnt_d”, “last_credit_pull_d”.
- Mutate some categorical variables into numeric variables because it makes more sense, like interest rate.
- Mutate the response “loan_status” into a factor.
- There are 3 records that lack all attributes except a loan status, remove them.
- After removing, there are 29774 records and 38 variables left.

Exploratory Analysis

Explore the Target

In all 29774 records in the training set, the number of default cases is 4477, which means the default rate is 15%.

Loan Status	N	PCT
Current	25297	0.8496
Default	4477	0.1504



Explore Categorical Variables

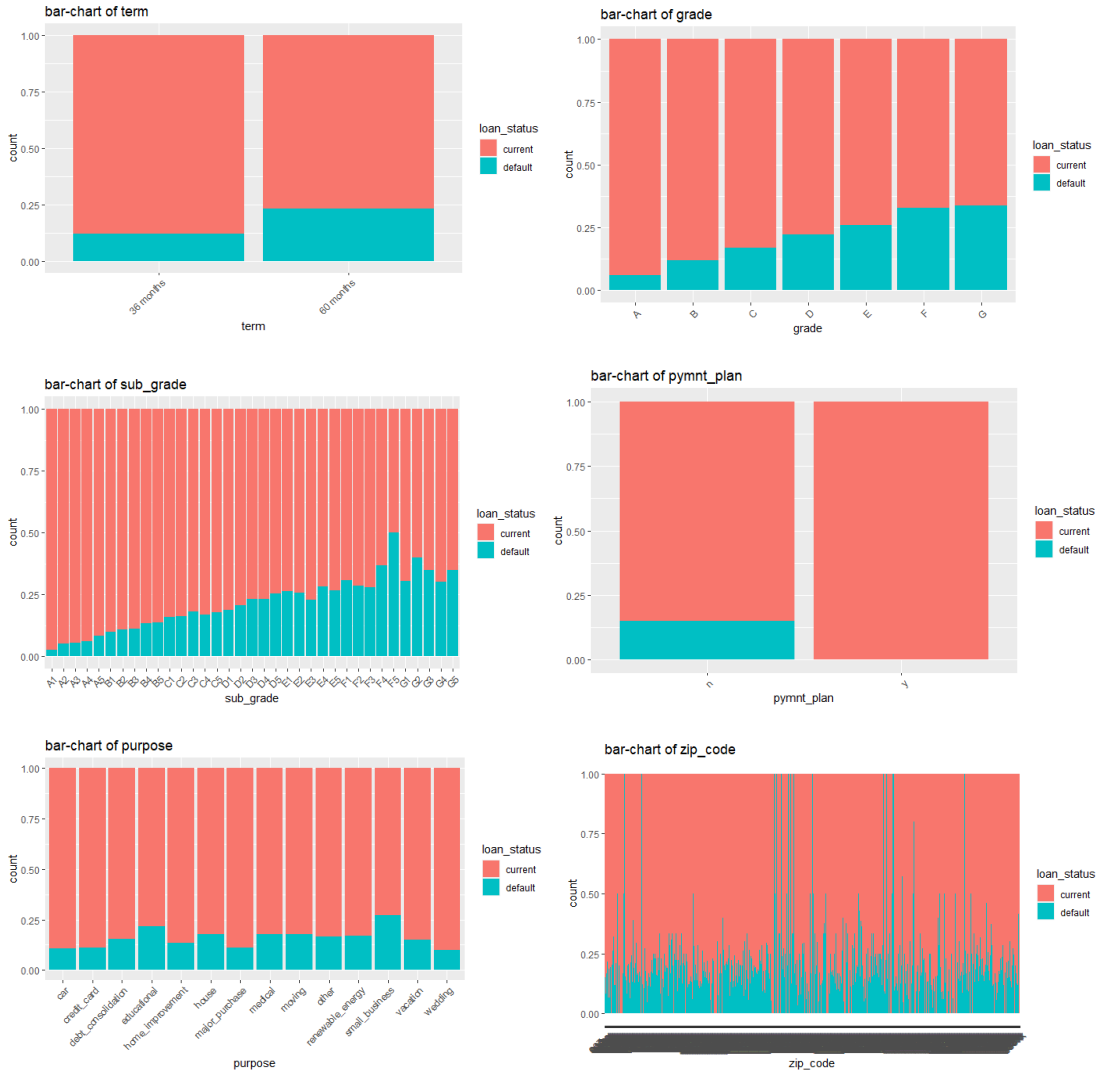
The important statistics of categorical variables are shown below in a table:

Columns	# Missing	Complete Rate	Min	Max	# Distinct
term	0	1.0000	9	9	2
grade	0	1.0000	1	1	7
sub_grade	0	1.0000	2	2	35
emp_length	0	1.0000	3	9	12
home_ownership	0	1.0000	3	8	5
verification_status	0	1.0000	8	15	3
pymnt_plan	0	1.0000	1	1	2
purpose	0	1.0000	3	18	14
zip_code	0	1.0000	5	5	819
addr_state	0	1.0000	2	2	50

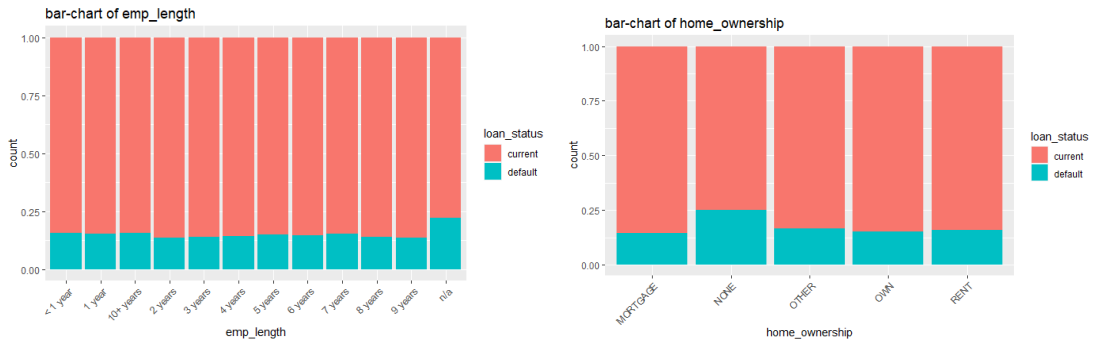
The stacked bar charts of categorical variables are shown below, from which we can see the default rate within each category. Therefore, this is a useful tool to help us decide which variables are useful to detect on default, i.e., the screening process.

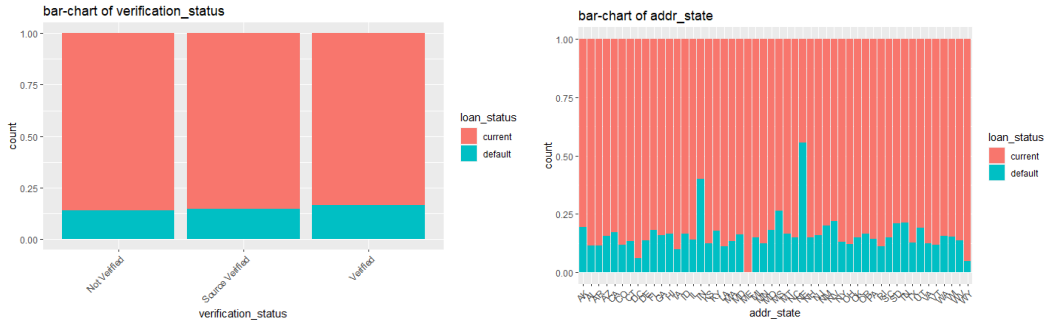
We present the bar charts together with our screening results.

First, from the 6 charts below, we can see the percentage of default within different categories is different. Therefore, these variables can be seen as useful predictors when we build machine learning models.



In the 4 charts below, the default rate within each category is more evenly distributed, i.e., not much different among different categories. They are less likely to be significant when predicting default.





Explore Numeric Variables

The important statistics of numeric variables are shown below in a table:

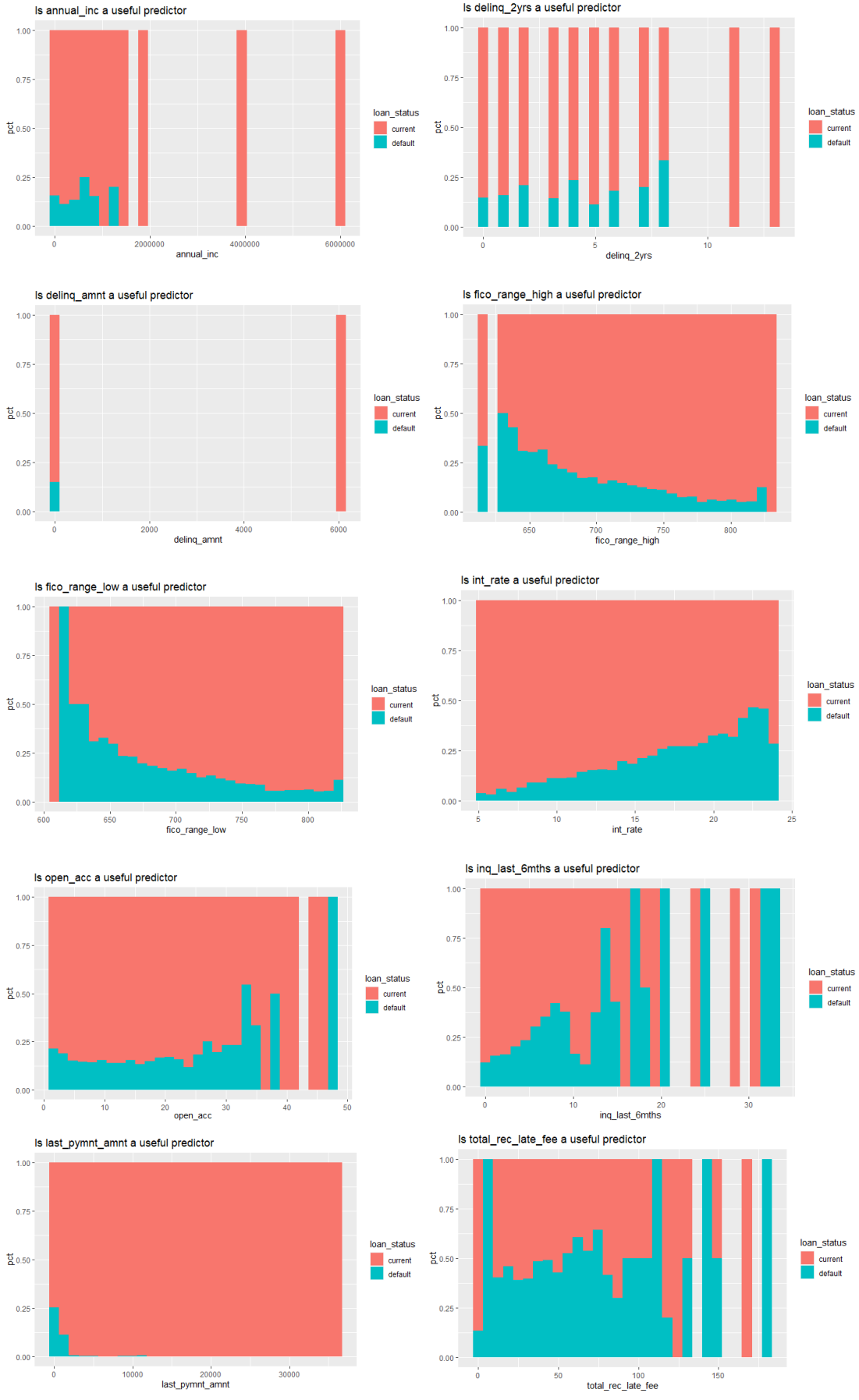
Column	Count	# miss	# dist	Mean	max	min	std
Acc_now_delinq	29774	20	3	0.00013	1.00	0.00	0.01159
Annual_inc	29774	1	4287	69201.23	6000000.0	2000.00	66566.415
Delinq_2yrs	29774	20	12	0.1550	13.00	0.00	0.52414
Delinq_amnt	29774	20	4	0.2043	6053.00	0.00	35.0915
Dti	29774	0	2846	13.384	29.99	0.00	6.7390
Fico_range_high	29774	0	43	717.05	829.00	614.00	36.3101
Fico_range_low	29774	0	43	713.05	825.00	610.00	36.3101
Funded_amnt	29774	0	981	10843.64	35000.00	500.00	7147.0522
Funded_amnt_inv	29774	0	6862	10149.66	35000.00	0.00	7130.8560
Inq_last_6mths	29774	20	28	1.0841	33.00	0.00	1.542828
installment	29774	0	13255	323.808	1305.19	15.67	209.7716
Int_rate	29774	0	390	12.166	24.11	5.42	3.716096
Last_pymnt_amnt	29774	0	26903	2615.405	36115.2	0	4373.69960
Loan_amnt	29774	0	827	11109.43	35000	500	7404.6523
Open_acc	29774	20	45	9.33901	47	1	4.51317
Out_prncp	29774	0	383	11.79629	3126.61	0	123.5276
Out_prncp_inv	29774	0	384	11.76432	3123.44	0	123.2313
Pub_rec	29774	20	7	0.05855	5	0	0.2479
Pub_rec_bankrupcies	29774	963	4	0.04533	2	0	0.2090
Revol_bal	29774	0	18399	14310.00	1207359	0	22696.5458
Revol_util	29774	64	1095	49.0854	119	0	28.327
Tax_liens	29774	76	3	0.000034	1	0	0.00580
Total_acc	29774	20	79	22.08278	81	1	11.58967
Total_rec_late_fee	29774	0	1604	1.50478	180.2	0	7.7227
Earliest_cr_line_y	29774	20	53	25.19789	53	-46	7.41998
Issue_w	29774	0	55	633.8805	807.0195	572.162	52.748
Last_credit_pull_w	29774	2	110	406.285	811.4481	324.162	97.410
Last_pymnt_w	29774	64	107	507.134	780.8767	324.162	84.399

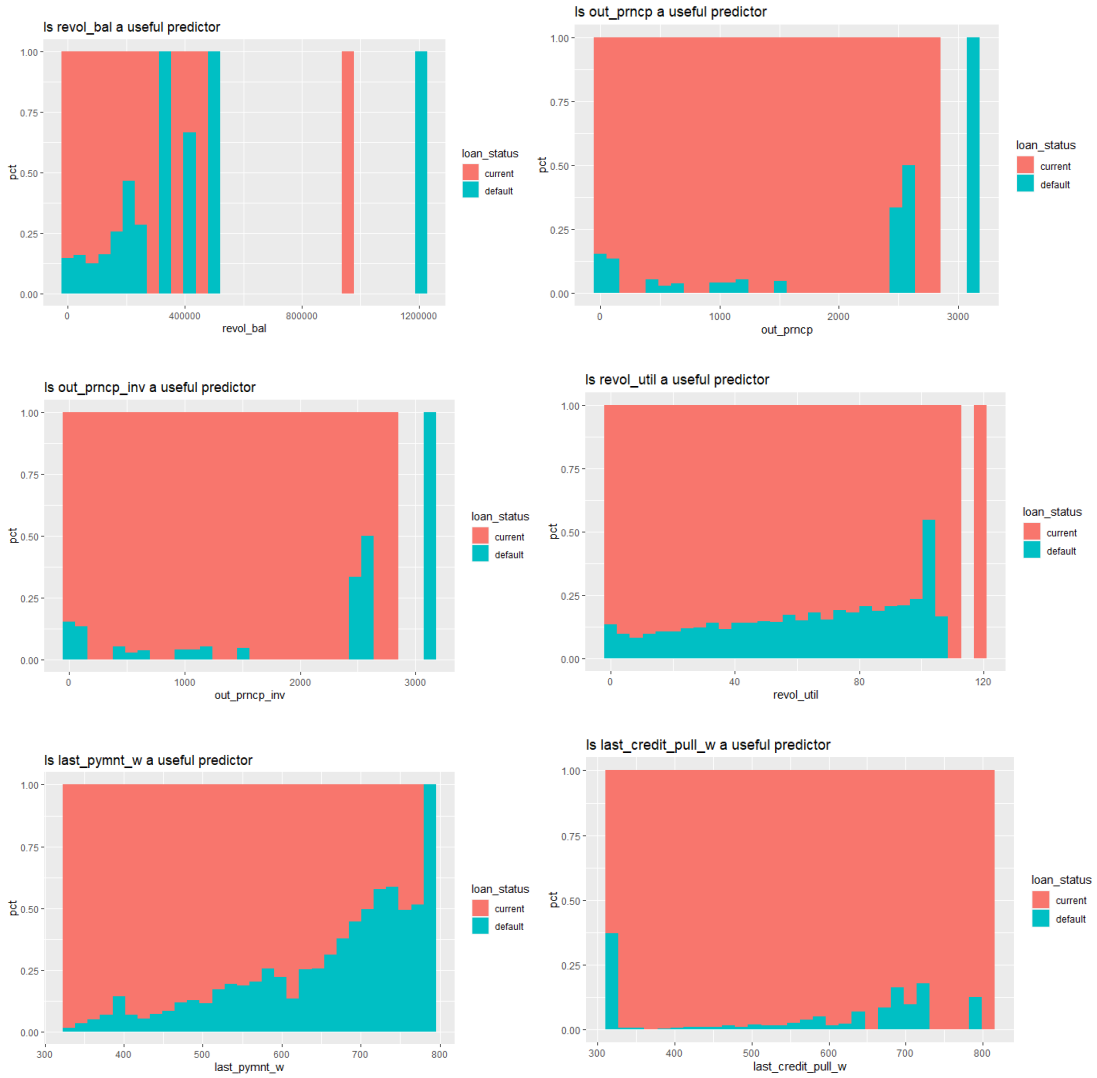
The stacked histogram charts of numeric variables are shown below, from which we can see the distribution of default rate and how it changes as the distribution of variables changes. Therefore, this is a useful tool to help us decide which variables are useful to detect on default.

We present the histogram charts together with our screening results.

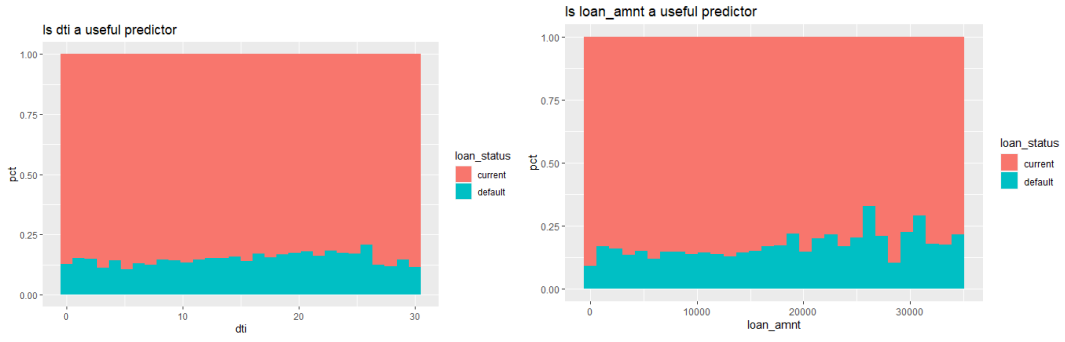
First, from the 6 charts below, we can see the percentage of default within different categories is different. Therefore, these variables can be seen as useful

predictors when we build machine learning models.





From the charts below, as the numbers on the x-axis increase, the percentage of default doesn't appear a clear pattern, instead, it stays relatively stable. Therefore, these variables are less likely useful.



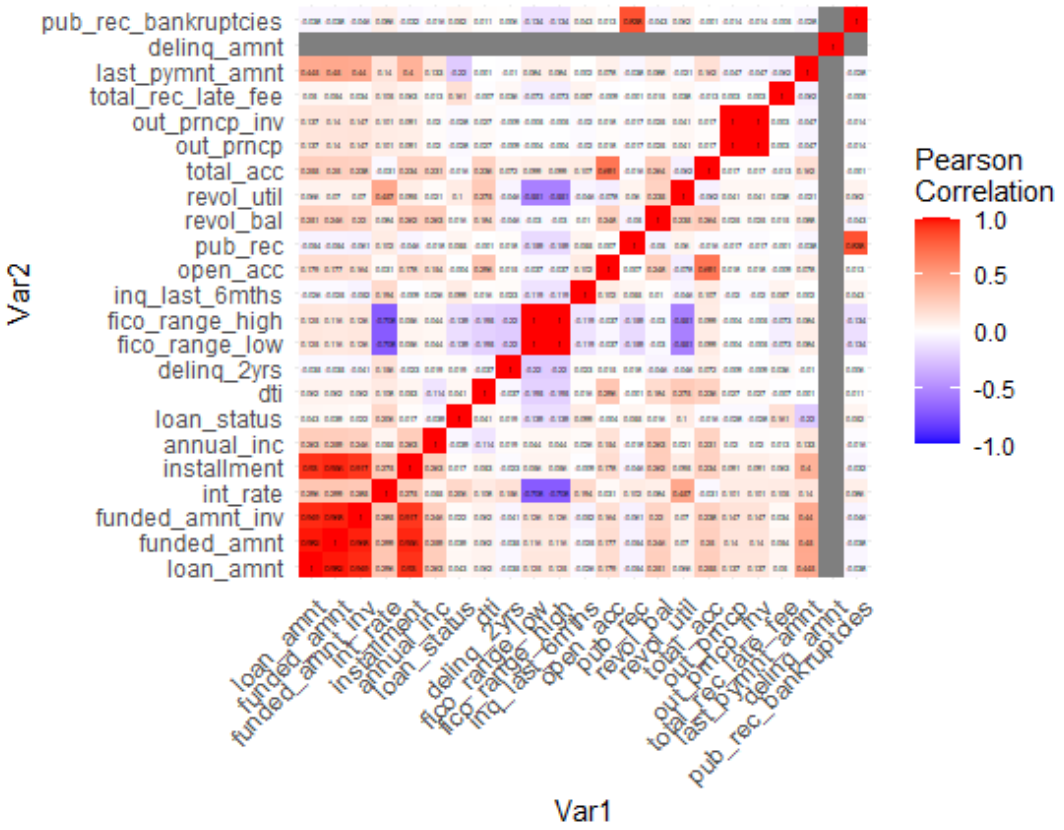
Explore Correlations

The correlation matrix of numeric variables and the target is shown below.

Clearly, "out_prncp" and "out_prncp_inv", "funded_amnt" and "funded_amnt_inv" are highly correlated, since the latter ones are the transformations of the former ones. "loan_amnt", "funded_amnt", and "installment" are highly correlated. The two boundaries range the borrower's FICO at loan origination belongs to are negatively correlated with the interest rate.

As for the target variable, "int_rate", and "total_rec_late" are positively related to the status of the loan, while "fico_range_high" and "fico_range_low" are negatively

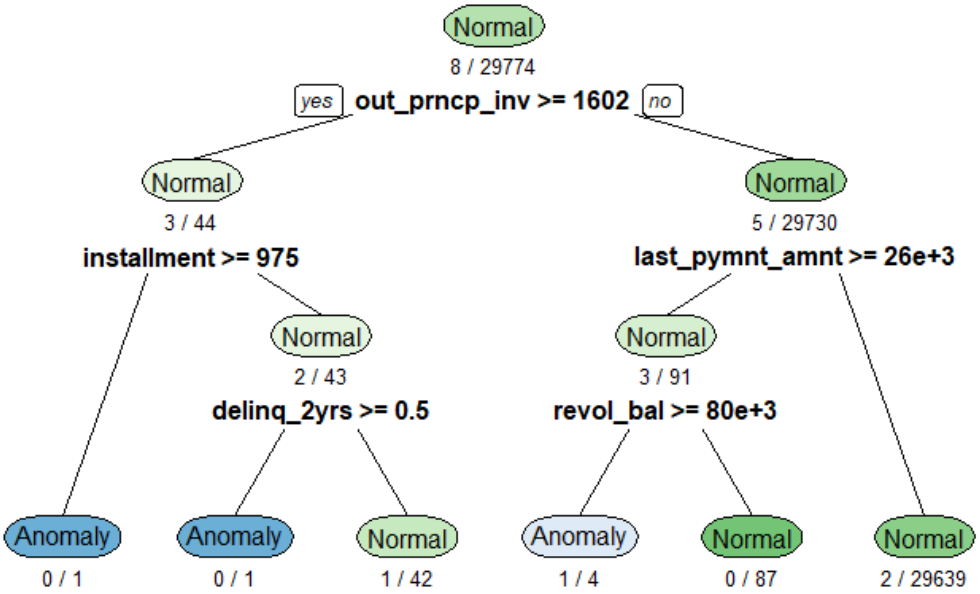
related to it.



Detect Anomaly

Global Anomaly Rules

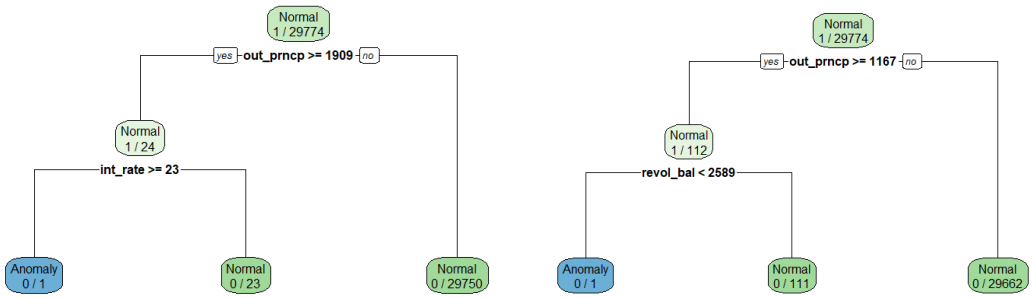
This is a tree visualization that can be used to detect anomalies and study their attributes. From what we can see, there are mainly 3 nodes containing the majority of all detected anomalies.

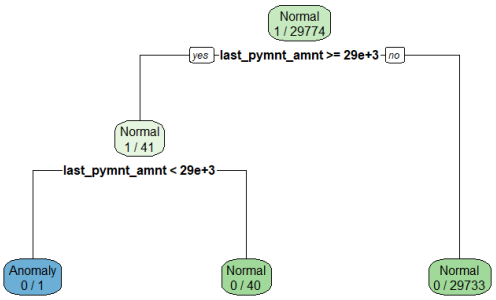


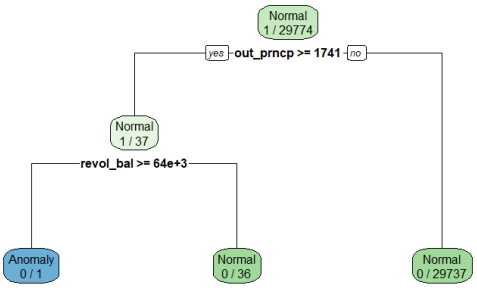
IF out_prncp_inv >= 1602 & installment >= 975 THEN Anomaly coverage 0%
IF out_prncp_inv >= 1602 & installment < 975 & delinq_2yrs >= 0.5 THEN Anomaly coverage 0%
IF out_prncp_inv < 1602 & last_pymnt_amnt >= 25714 & revol_bal >= 79777 THEN Anomaly coverage 0%

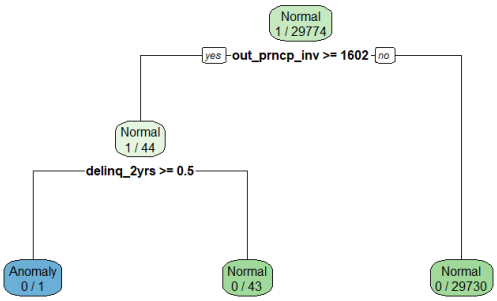
Local Anomaly Rules

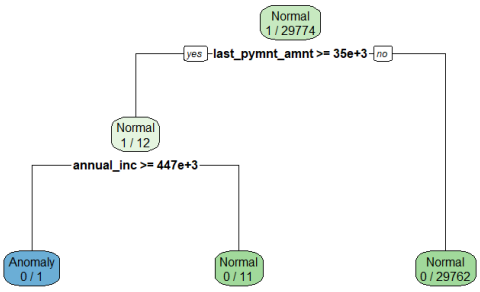
Below is 8 smaller trees that can explain how 8 anomalies were found separately.

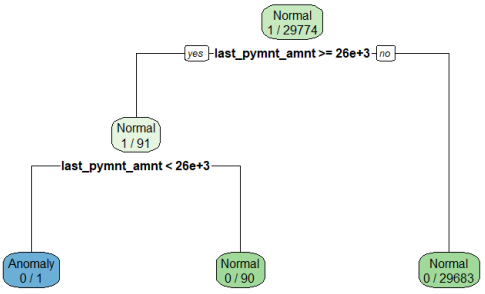


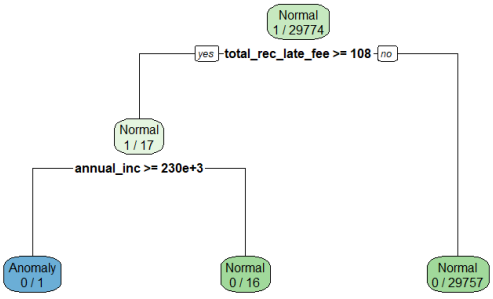












Rule	Cover
IF out_prncp >= 1909 & int_rate >= 23	0%
IF out_prncp >= 1167 & revol_bal < 2589	0%
IF last_pymnt_amnt is 29024 to 29153	0%
IF out_prncp >= 1741 & revol_bal >= 64267	0%
IF out_prncp_inv >= 1602 & delinq_2yrs >= 0.5	0%
IF last_pymnt_amnt >= 34537 & annual_inc >= 446500	0%
IF last_pymnt_amnt is 25714 to 25765	0%
IF total_rec_late_fee >= 108 & annual_inc >= 229750	0%

Model Building & Training

Data Preparation & Transformation

- Randomly split data into 70/30, as training set and testing set, respectively.
- Split the training data into 10 folds.
- Mutate response into a factor.
- Feature Engines Setting:
    - Impute missing values with the median (numeric) or assign them to a new category called "unknown" (categorical).
    - Scale numeric variables.
    - Dummy code categorical variables.
    - Create a new level for categorical variables.
    - Order, and assign an integer to each unique level to high cardinality variables

Derive new variables

No extra derivations except what was mentioned before (repeat here again):

1. Mutate "issue_d", "earliest_cr_line", "last_pymnt_d", "last_credit_pull_d" as date-type variables.
2. Calculate the time difference in weeks or years between the current date and those dates mentioned above.
3. Name them as "issue_w", "earliest_cr_line_y", "last_pymnt_w", "last_credit_pull_w", respectively.
4. Remove "issue_d", "earliest_cr_line", "last_pymnt_d", "last_credit_pull_d".

- Mutate some categorical variables into numeric variables because it makes more sense, like interest rate.

What is worth noticing is that the original data file has a mistaken format, which caused the mdy() function to mutate some of the years of "earliest_cr_line" incorrectly (mistake 1968 as 2068). Again, this is because of the format of original data, instead of coding techniques. However, this variable will be mutated the same way when making predictions, and luckily, this variable contributes almost nothing both before and after the mutation, according to my trials.

Model Training

Variable	**Role**
Loan_status	Response
term	Categorical predictor
grade	Categorical predictor
sub_grade	Categorical predictor
emp_length	Categorical predictor
home_ownership	Categorical predictor
verification_status	Categorical predictor
pymnt_plan	Categorical predictor
purpose	Categorical predictor
zip_code	Categorical predictor
addr_state	Categorical predictor
Acc_now_delinq	Numeric predictor

Annual_inc	Numeric predictor
Delinq_2yrs	Numeric predictor
Delinq_amnt	Numeric predictor
Dti	Numeric predictor
Fico_range_high	Numeric predictor
Fico_range_low	Numeric predictor
Funded_amnt	Numeric predictor
Funded_amnt_inv	Numeric predictor
Inq_last_6mths	Numeric predictor
installment	Numeric predictor
Int_rate	Numeric predictor
Last_pymnt_amnt	Numeric predictor
Loan_amnt	Numeric predictor
Open_acc	Numeric predictor
Out_prncp	Numeric predictor
Out_prncp_inv	Numeric predictor
Pub_rec	Numeric predictor
Pub_rec_bankrupcies	Numeric predictor
Revol_bal	Numeric predictor
Revol_util	Numeric predictor
Tax_liens	Numeric predictor
Total_acc	Numeric predictor
Total_rec_late_fee	Numeric predictor
Earliest_cr_line_y	Numeric predictor
Issue_w	Numeric predictor
Last_credit_pull_w	Numeric predictor
Last_pymnt_w	Numeric predictor

Recipe

```
the_recipe <- recipe(loan_status ~ ., data = train )%>%
  step_impute_median(all_numeric_predictors()) %>%
  step_scale(all_numeric_predictors()) %>%
  step_novel(all_nominal_predictors()) %>%
  step_unknown(all_nominal_predictors()) %>%
  step_integer(sub_grade, zip_code, addr_state) %>%
  step_dummy(all_nominal_predictors())

the_bake <- bake(the_recipe %>% prep(), train )

skim(the_bake)
```

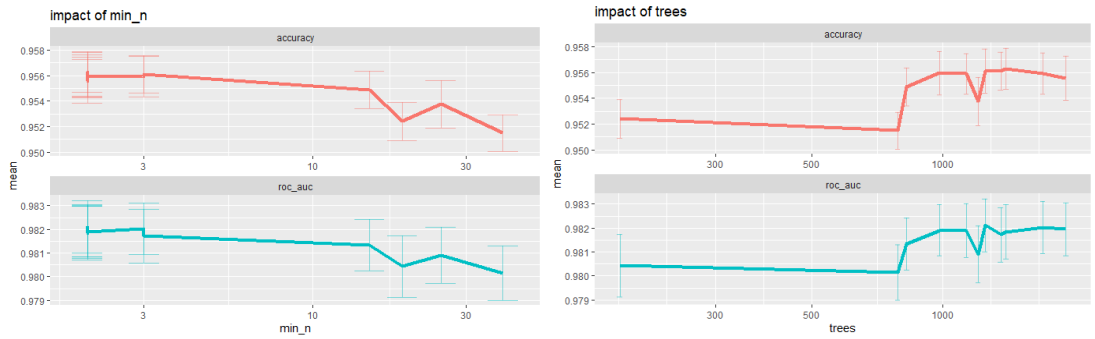
Random Forest

Hyperparameter Tuning

We used K-fold Validation and split the training data into 10 folds, and we use the Bayes tuning method to tune hyperparameters.

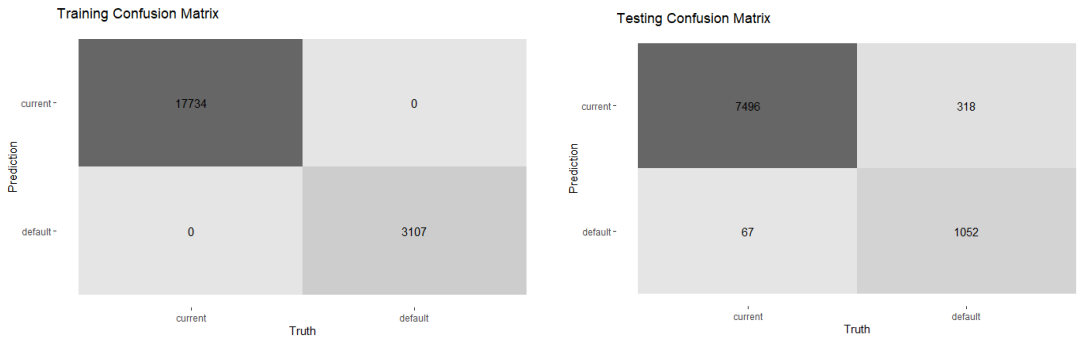
- Tuning Performance

From the chart below, we can see that the optimal min_n is around 3 while the optimal number of trees is above 1000, as the accuracy and AUC both reached the highest in these charts.



- Bayes tune result:
  - Trees = 1251
  - Min_n = 2

Model Performance



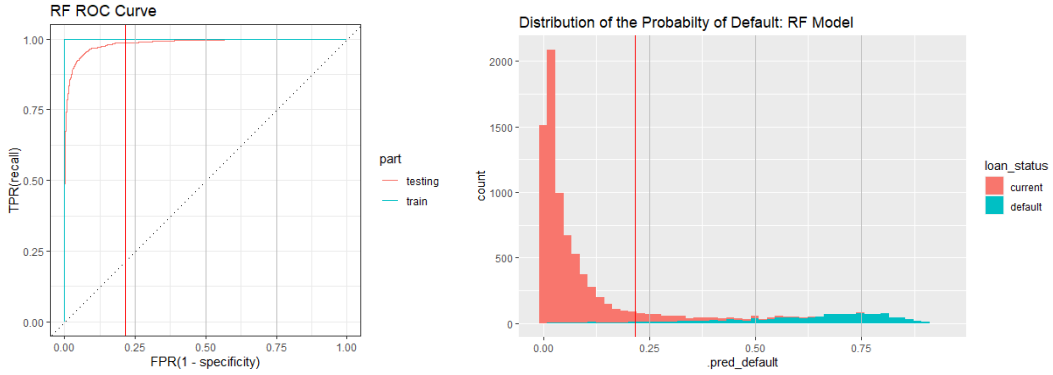
	Accuracy	Log loss	AUC	Precision	Recall	F1
Training	1.0000	0.0499	1.0000	1.0000	1.0000	1.0000
Testing	0.9569	0.1524	0.9842	0.9401	0.7679	0.8453

Operating Range

ROC and Distribution of the probability of default

The desired operating level is at **5% FNR**. At this level, we will incorrectly define 5% positives as negatives, i.e., 5% default as current. The desired operating level

is marked as a red v-line, while other common thresholds are marked as gray v-lines.



Operation Chart (precision/recall/FPR/FNR/Threshold)

FNR	Threshold	FPR	Recall (TPR)	Precision
0.01	0.070	0.302	0.981	0.380
0.02	0.121	0.159	0.979	0.530
0.03	0.154	0.116	0.970	0.605
0.04	0.193	0.086	0.961	0.666
0.05	**0.216**	**0.074**	**0.950**	**0.699**
0.06	0.241	0.063	0.940	0.730
0.07	0.263	0.054	0.931	0.758
0.08	0.287	0.045	0.921	0.787
0.09	0.311	0.040	0.910	0.812
0.10	0.329	0.033	0.900	0.829

- Given the context, we choose an FNR at 5%. It means we will accept 5% of the loans which will default as current.
- To achieve this FNR rate, when the predicted default rate is higher than 0.216, we will classify it as default; and we will detect it as current when the predicted default rate is below 0.216.
- At this FNR rate, the FPR is 0.074, which means we will define 7.4% of the current loan as default.
- The recall/TPR is 0.950, which means we will successfully identify 95% of the total default loans.
- The precision will be 0.699, which means among all the loans we defined as the default, 69.9% of them will actually default.

Potential Savings (in Every 1000 Loan Applications):

Accept all loan applications (nowadays): ***1000 * 0.15 * mean (loan amount)***

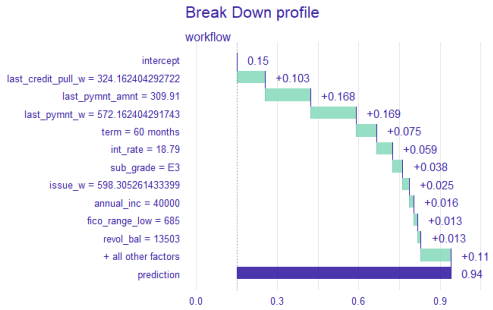
Operating at 5% FNR: (5% will be the default, 7.4% current loan will be rejected and lose interest income): ***1000*0.05*mean (loan amount) + 1000 * 0.074 * mean (interest rate) * mean (loan amount)***

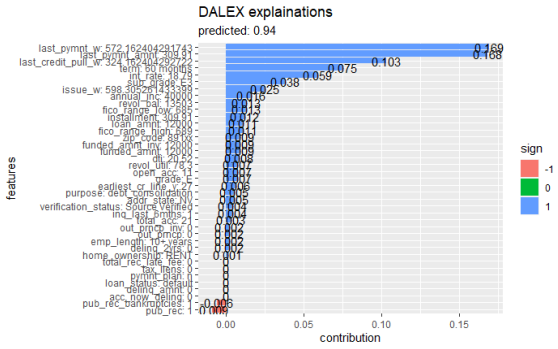
Potential Savings = $1,010,923

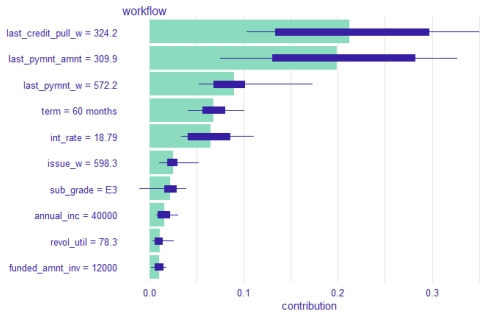
DALEX, SHAPLEY, and VIP

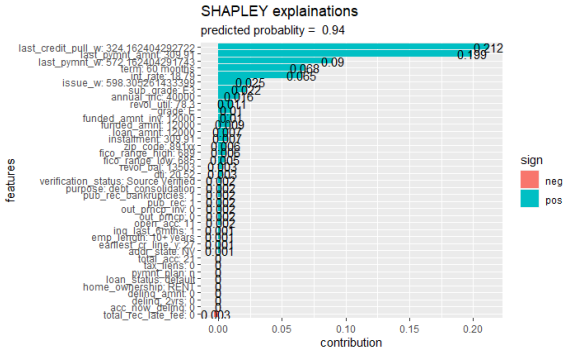
We used 3 methods to interpret the model from a global perspective, VIP, DALEX breakdown, and SHAPLEY, and we used the highest prediction probability to build DALEX and SHAPLEY chart.

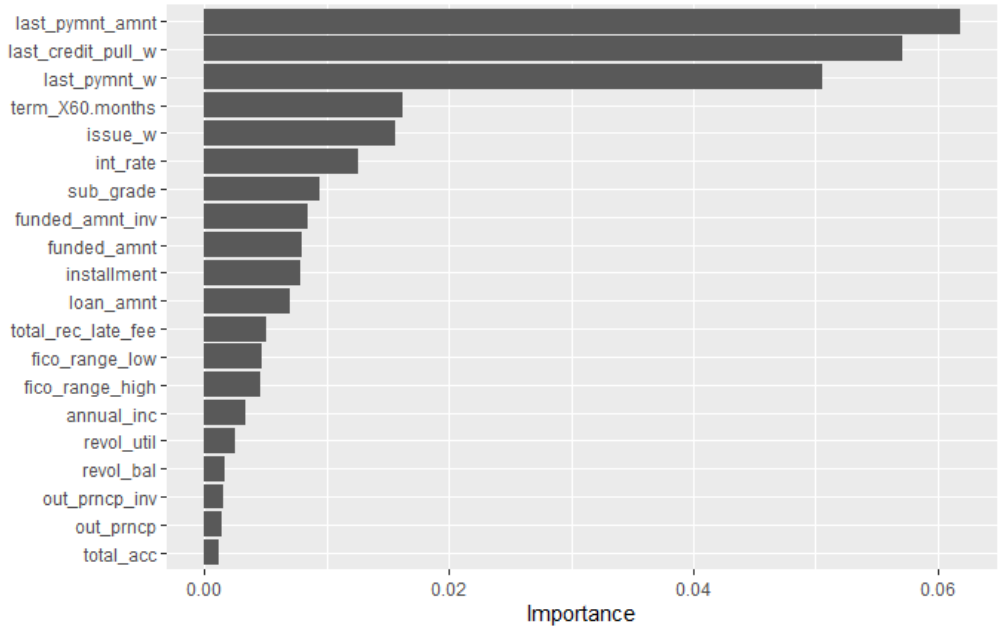
According to the charts below, “last_pymnt_w”, “last_pymnt_amnt”, “last_credit_pull_w”, “term”, and “int_rate” are important predictors to detect default.









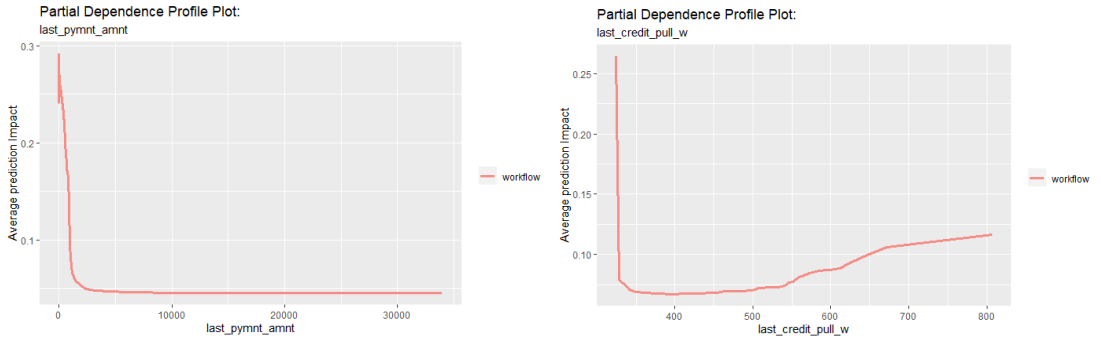


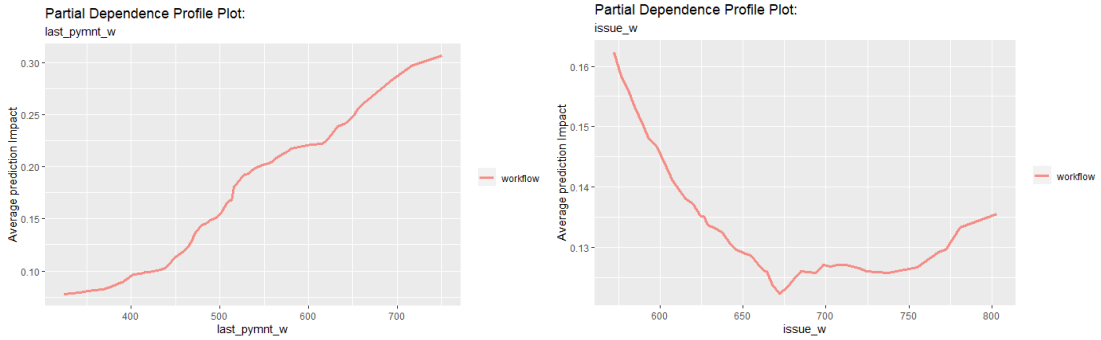
Partial Dependence Plots

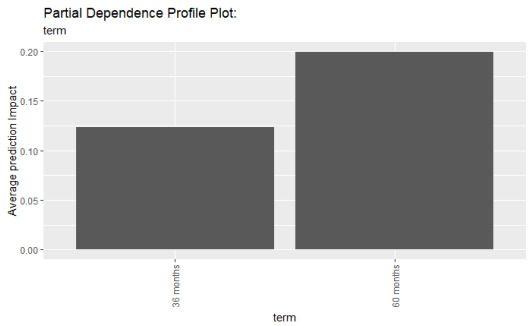
We selected top variables from the global explanations above to draw partial dependence plots, in order to study the interaction between one specific predictor and the response.

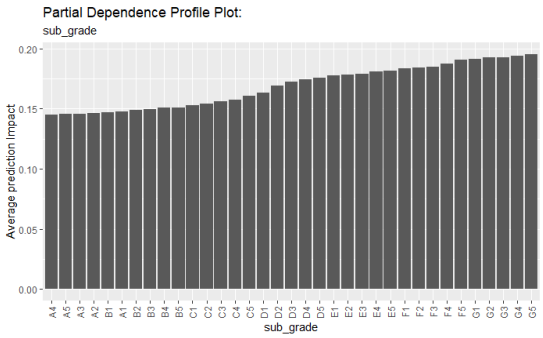
From the charts below we can see that, when “last_pymnt_amnt” or “issue_w” is very low, when “last_credit_pull_w” is very small or big, the probability that this loan will be default is higher. Besides, a higher “last_pymnt_w” will result a higher probability.

60-months term and a lower grade (F or G) will also result a higher default probability.







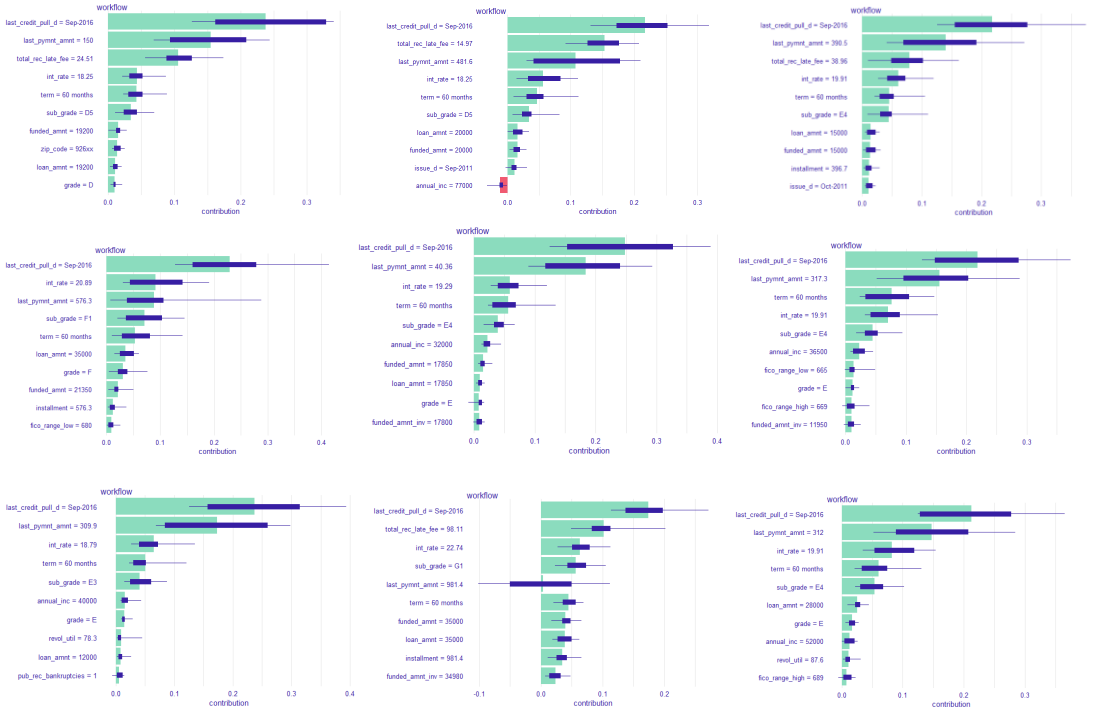


Local Explanation: Top TP FP FN Records

Top 9 TP Records

We have the results of the Top 10 true positive records but we only display 9 of them for a better viewing.

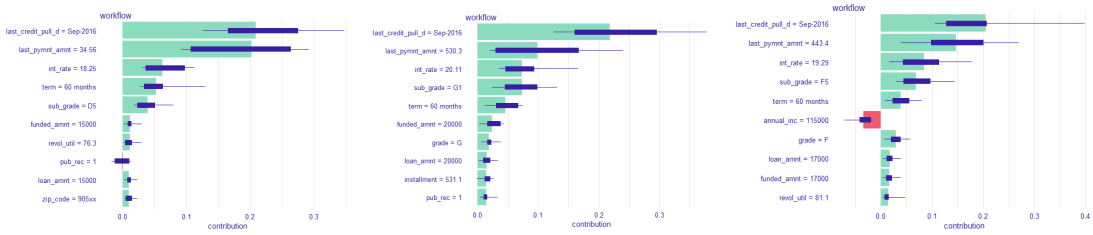
Among those records, there are some similarities: “last_credit_pull_d” contributes the most, and the value of it is all “Sep-2016”. Besides, “total_rec_late_fee” and “last_pymnt_amnt” contributes a lot as well. Most of them has a lower “last_pymnt_amnt”, usually higher than $300 but lower than $500.

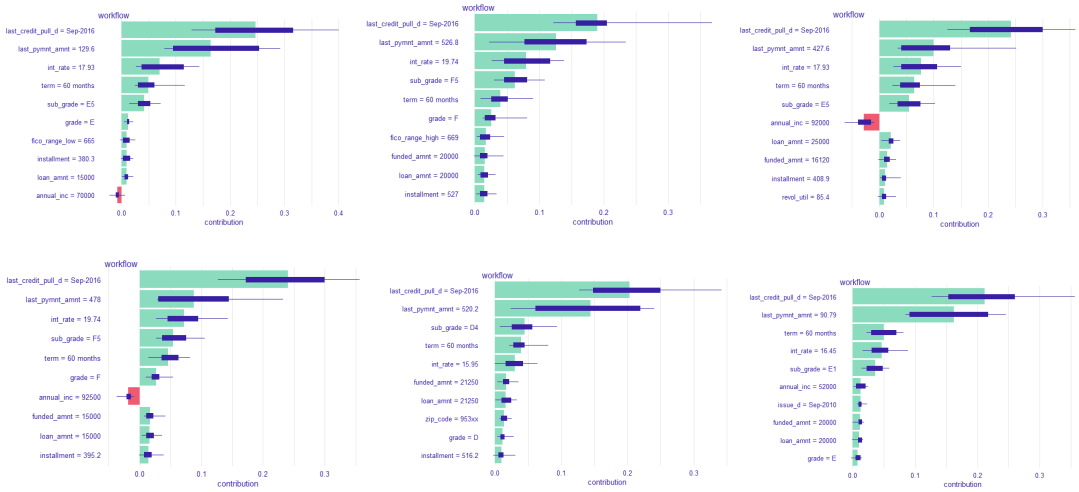


Top 9 FR Records

We have the results of the Top 10 false positive records but we only display 9 of them for a better viewing.

Among those records, there are some similarities: “last_credit_pull_d” contributes the most, and all these records have the same value “Sep-2016”. Besides, “int_rate” and “last_pymnt_amnt” contributes a lot as well. The “int_rate” of all those records are around 20%, and the “last_pymnt_amnt” is usually relatively low, around $500.



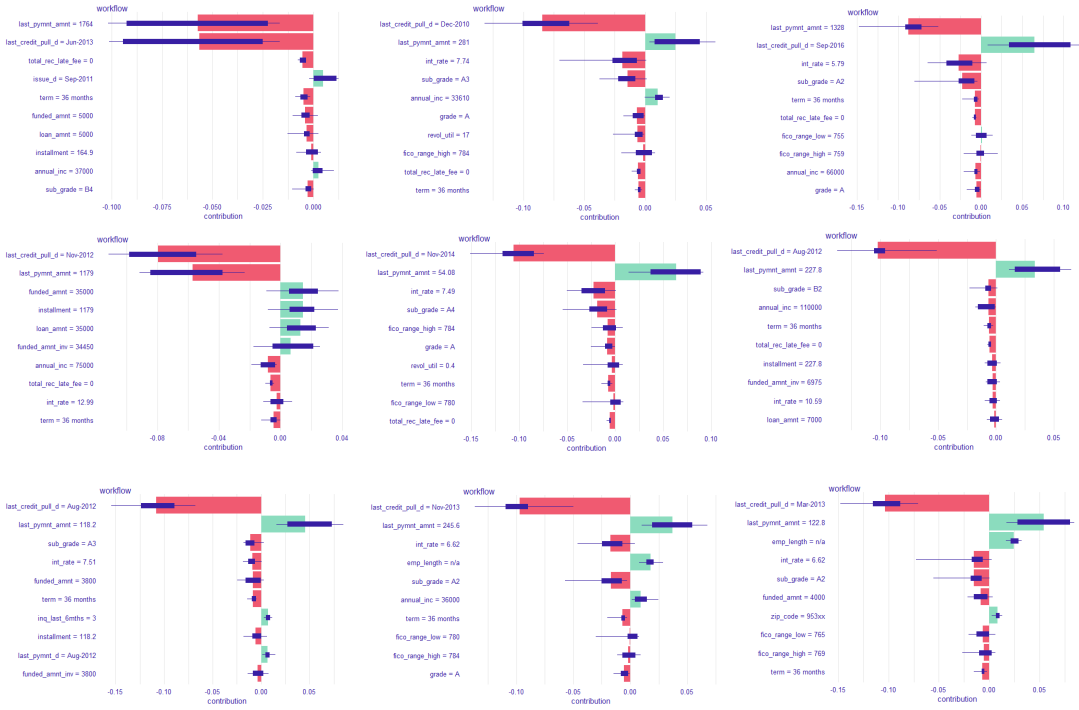


Top 9 FN Records

We have the results of the Top 10 false negative records, but we only display 9 of them for a better viewing.

Among those records, there are some similarities: “last_credit_pull_d” and “last_pymnt_amnt” contributes the most. The “last_pymnt_amnt” is either very high (above $1000), or every low (below $300).

and the value of it is all “Sep-2016”. Besides, “total_rec_late_fee” and “last_pymnt_amnt” contributes a lot as well. Most of them has a lower “last_pymnt_amnt”, usually higher than $300 but lower than $500. Their “int_rate” are all lower than other records, usually below 10%.



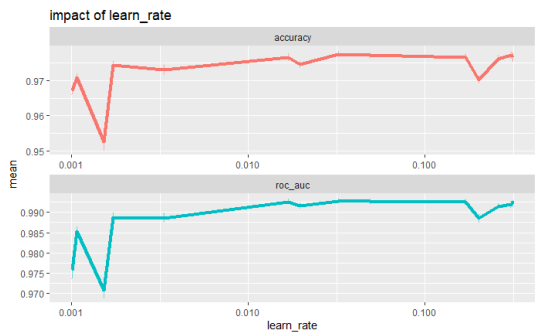
XGBoost

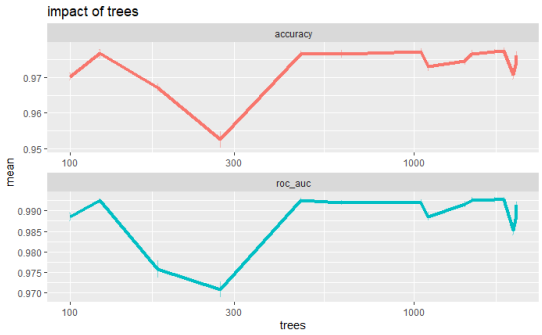
Hyperparameter Tuning

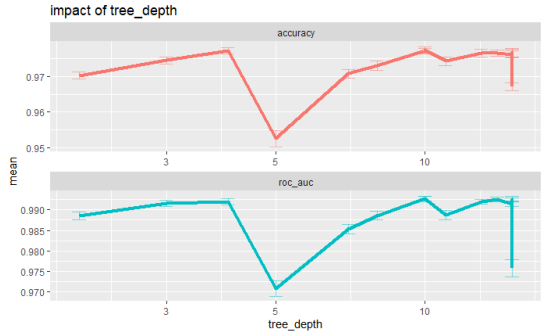
We used K-fold Validation and split the training data into 10 folds, and we use the Bayes tuning method to tune hyperparameters.

- Tuning Performance

From the chart below, we can see that the optimal learn_rate is between 0.01 to 0.1, while the optimal number of trees is above 500, the optimal tree_depth is about 10, as the accuracy and AUC both reached the highest in these charts.

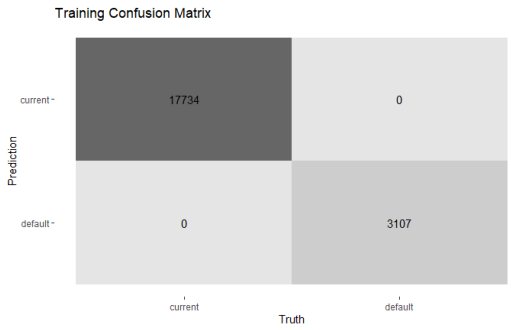


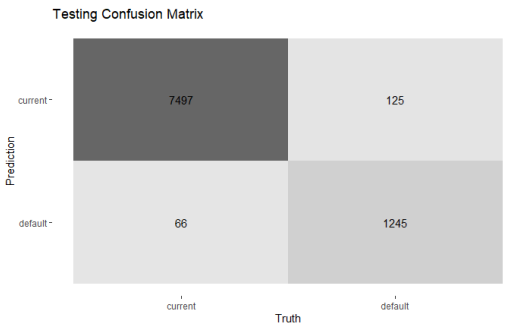




- Bayes tune result:
  - Trees = 1821
  - Tree_depth = 10
  - Learn_rate = 0.03225007

Model Performance



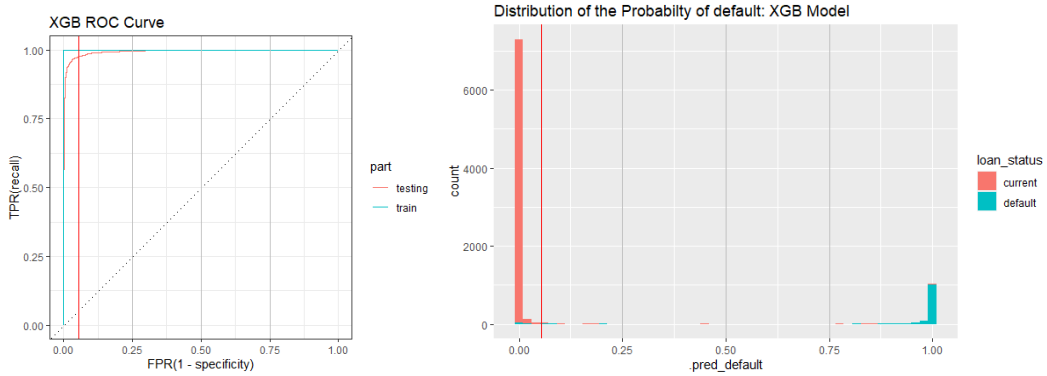


	Accuracy	Log loss	AUC	Precision	Recall	F1
Training	1.0000	0.0012	1.0000	1.0000	1.0000	1.0000
Testing	0.9786	0.0820	0.9937	0.9497	0.9088	0.9288

Operating Range

ROC and Distribution of the probability of default

The desired operating level is at **5% FNR**. At this level, we will incorrectly define 5% positives as negatives, i.e., 5% default as current. The desired operating level is marked as a red v-line, while other common thresholds are marked as gray v-lines.



Operation Chart (precision/recall/FPR/FNR/Threshold)

FNR	Threshold	FPR	Recall (TPR)	Precision
0.01	0.001	0.145	0.991	0.590
0.02	0.004	0.068	0.979	0.729
0.03	0.010	0.042	0.970	0.811
0.04	0.026	0.029	0.959	0.859
0.05	**0.054**	**0.020**	**0.951**	**0.890**
0.06	0.119	0.017	0.939	0.915
0.07	0.232	0.010	0.929	0.934
0.08	0.346	0.010	0.921	0.941
0.09	0.494	0.010	0.909	0.949
0.10	0.652	0.010	0.900	0.957

- Given the context, we choose an FNR at 5%. It means we will accept 5% of the loans which will default as current.
- To achieve this FNR rate, when the predicted default rate is higher than 0.054, we will classify it as default; and we will detect it as current when the predicted default rate is below 0.054.
- At this FNR rate, the FPR is 0.020, which means we will define 2% of the current loan as default.
- The recall/TPR is 0.951, which means we will successfully identify 95.1% of the total default loans.

- The precision will be 0.890, which means among all the loans we defined as the default, 89% of them will actually default.

Potential Savings (in Every 1000 Loan Applications):

Accept all loan applications (nowadays): ***1000 * 0.15 * mean (loan amount)***

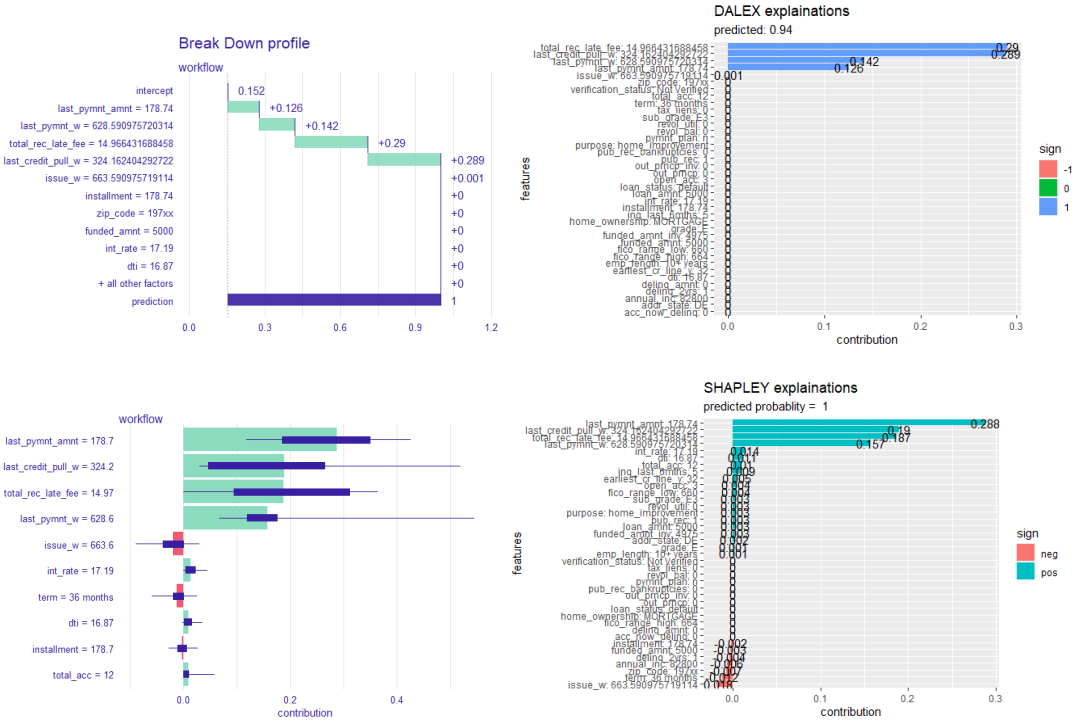
Operating at 5% FNR: (5% will be the default, 7.4% current loan will be rejected and lose interest income): ***1000*0.05*mean (loan amount) + 1000 * 0.02 * mean (interest rate) * mean (loan amount)***

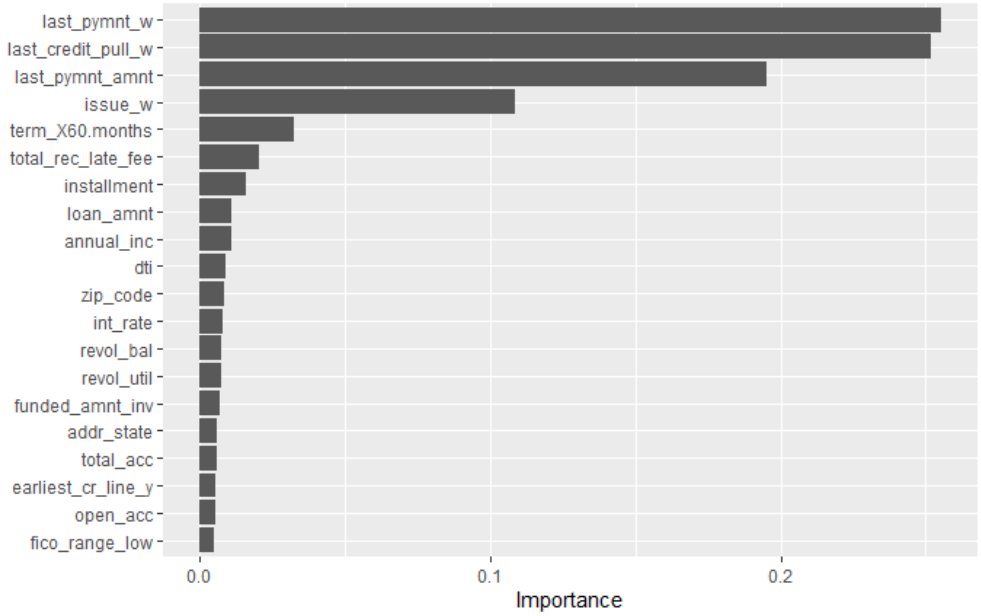
Potential Savings = $1,083,911

DALEX, SHAPLEY, and VIP

We used 3 methods to interpret the model from a global perspective, VIP, DALEX breakdown, and SHAPLEY, and we used the highest prediction probability to build DALEX and SHAPLEY chart.

According to the charts below, "last_pymnt_w", "last_pymnt_amnt", "last_credit_pull_w", "term", and "total_rec_late_fee" are important predictors to detect default.

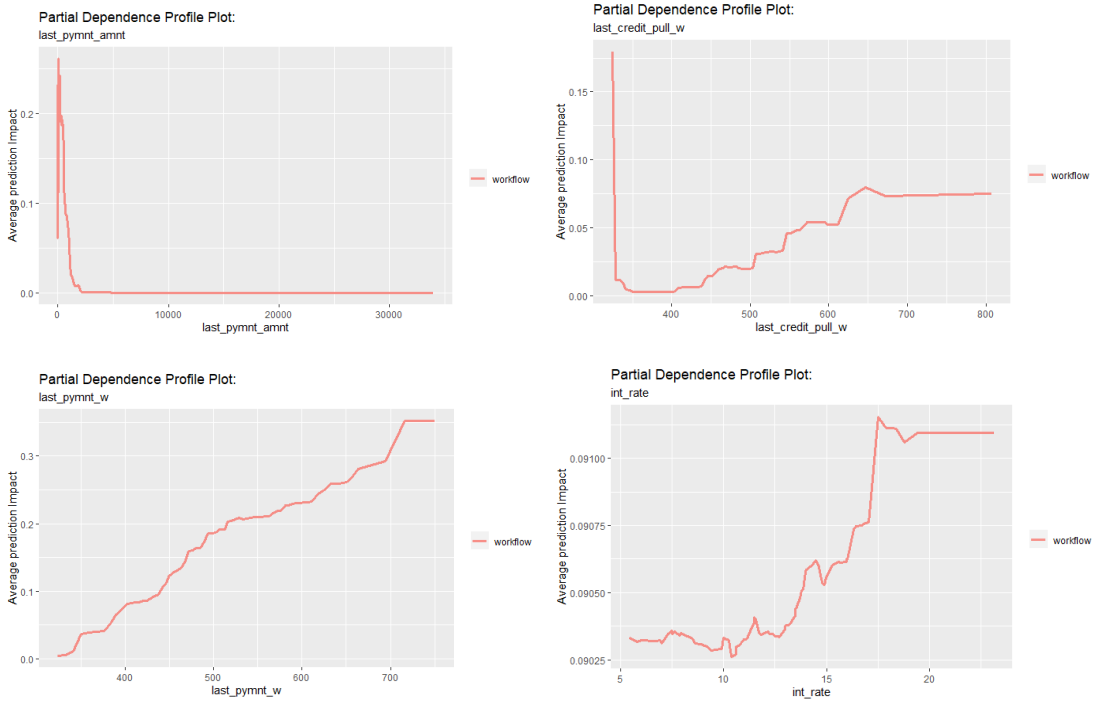


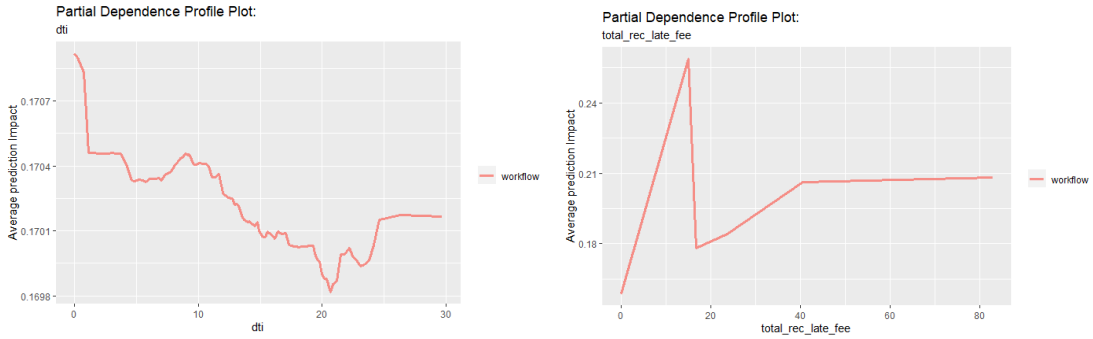


Partial Dependence Plots

We selected top variables from the global explanations above to draw partial dependence plots, in order to study the interaction between one specific predictor and the response.

From the charts below we can see that, when “last_pymnt_amnt” is very low, when “last_credit_pull_w” is very small or big, the probability that this loan will be default is higher. Besides, a higher “last_pymnt_w” or “int_rate” will result a higher probability. What’s more, a lower “dti” will also result in a higher probability of default.



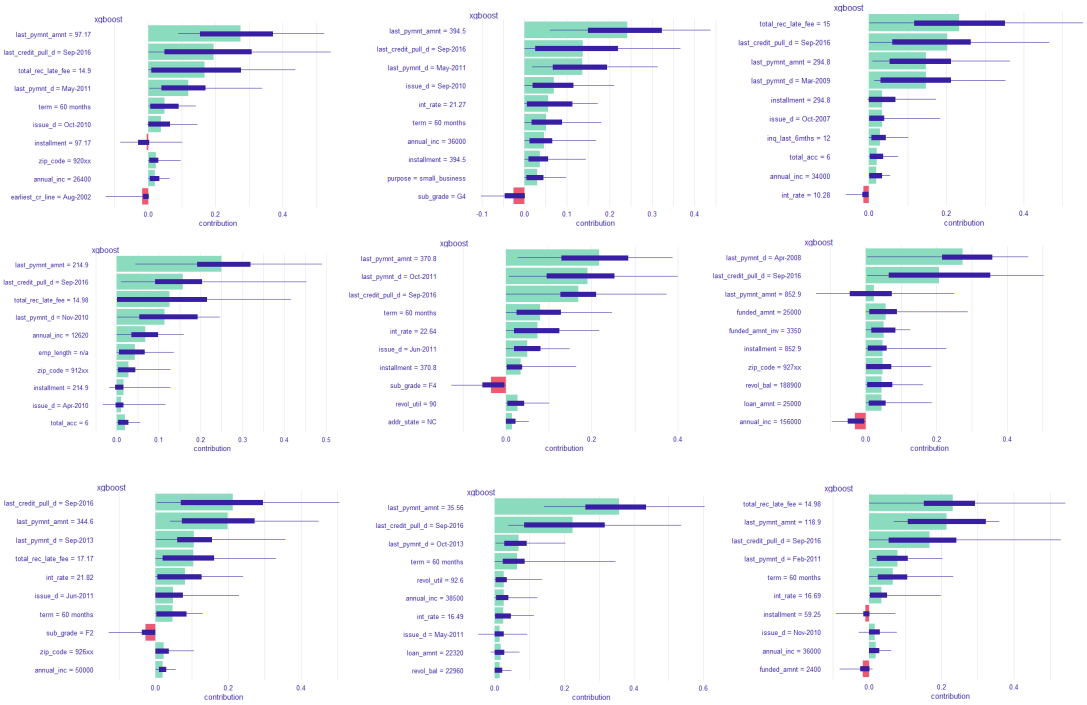


Local Explanation: Top TP FP FN Records

Top 9 TP Records

We have the results of the Top 10 true positive records but we only display 9 of them for a better viewing.

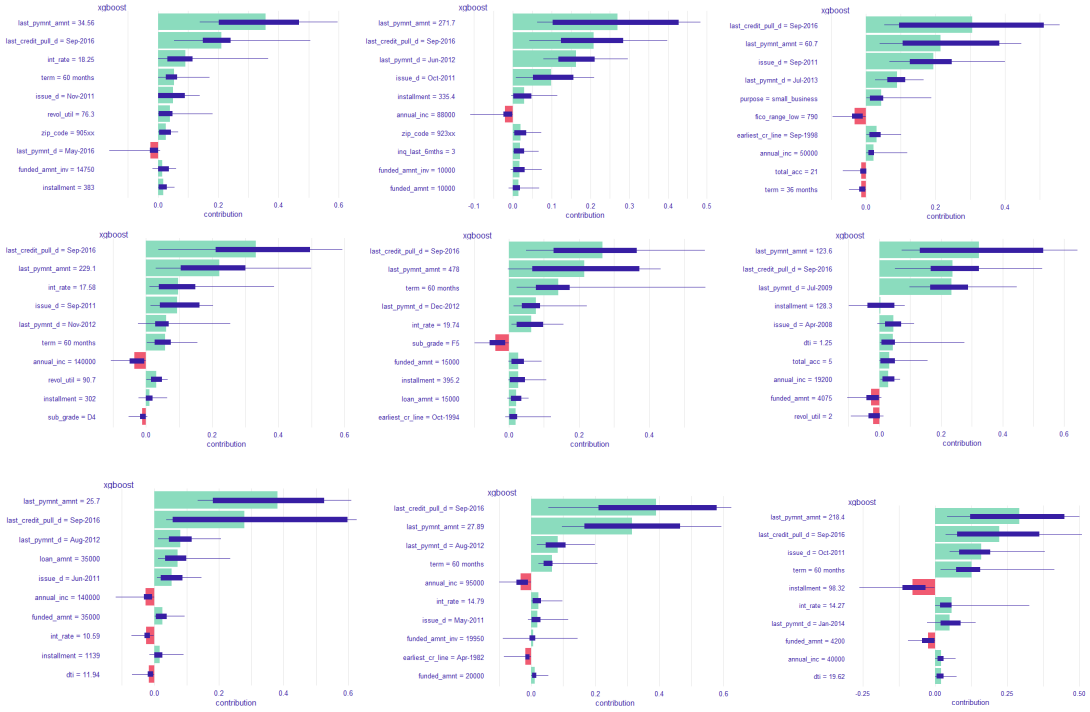
Among those records, lots of them have the same "last_credit_pull_d" which is "Sep-2016". When "term" is on the board, it's usually 60-months term. Their "last_pymnt_amnt" is below $500.



Top 9 FP Records

We have the results of the Top 10 false positive records but we only display 9 of them for a better viewing.

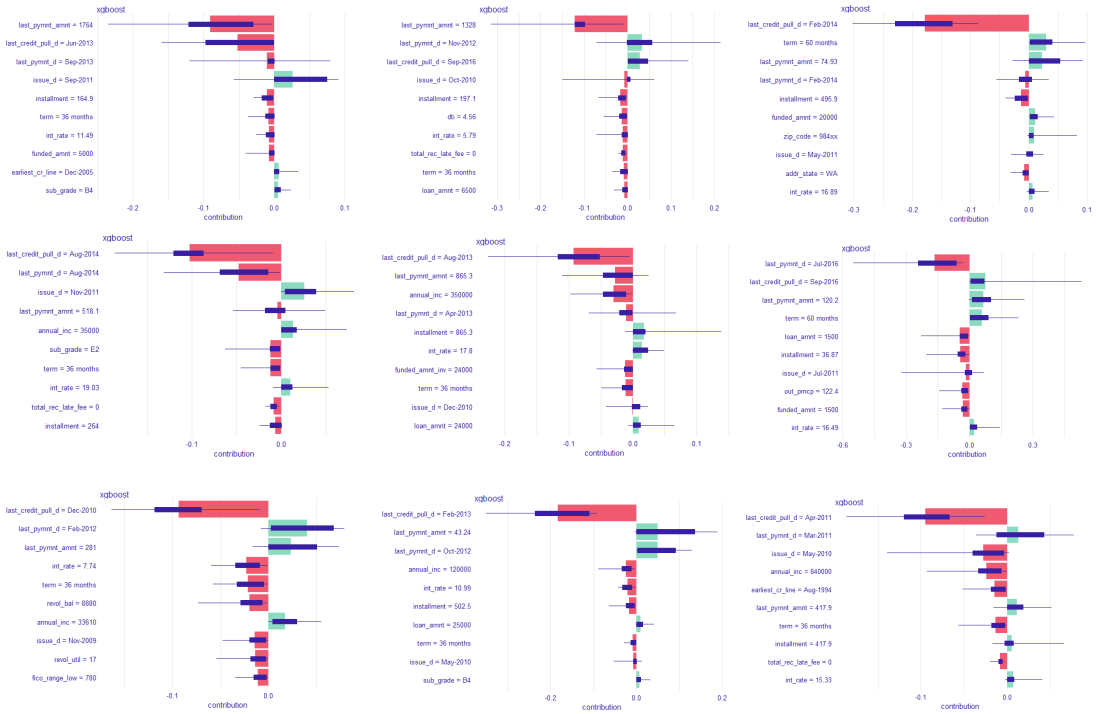
Among those records, lots of them have the same "last_credit_pull_d" which is "Sep-2016". When "term" is on the board, it's usually 60-months term. Their "last_pymnt_amnt" is below $300.



Top 9 FN Records

We have the results of the Top 10 false negative records but we only display 9 of them for a better viewing.

Among those records, lots of them have a higher “int_rate” which is between 15% and 20%. If the “term” appears, it’s usually 36-months.



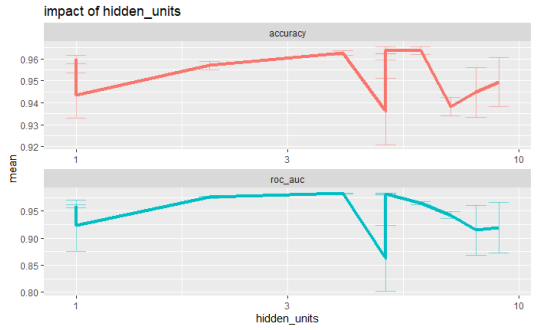
Neural Network

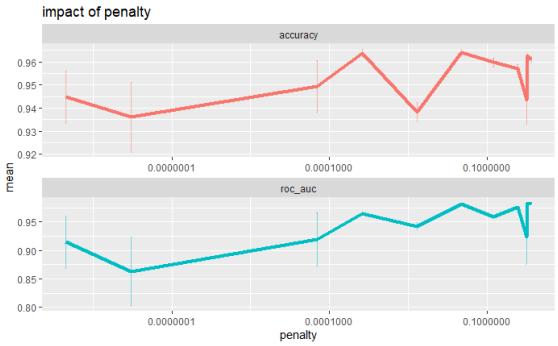
Hyperparameter Tuning

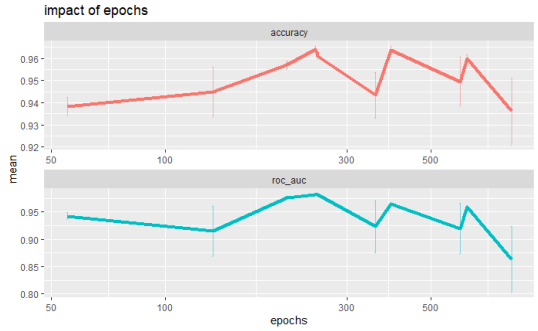
We used K-fold Validation and split the training data into 10 folds, and we use the Bayes tuning method to tune hyperparameters.

- Tuning Performance

From the chart below, we can see that the optimal hidden_unit is around 4, while the optimal penalty is between 0.0001 and 0.1, and the optimal epochs is around 250, as the accuracy and AUC both reached the highest in these charts.

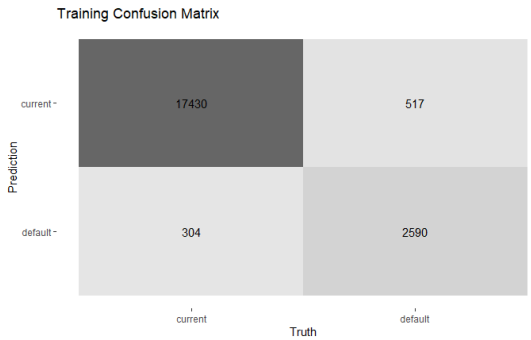


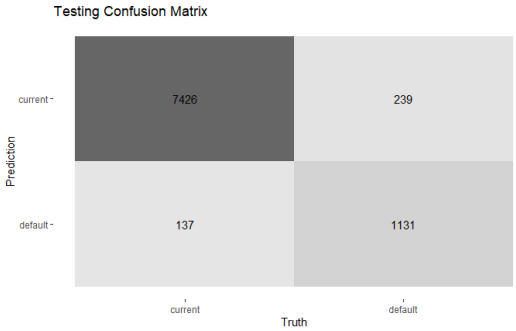




- Bayes tune result:
  - Hidden_units = 4
  - Penalty = 0.592135
  - Epochs = 250

Model Performance



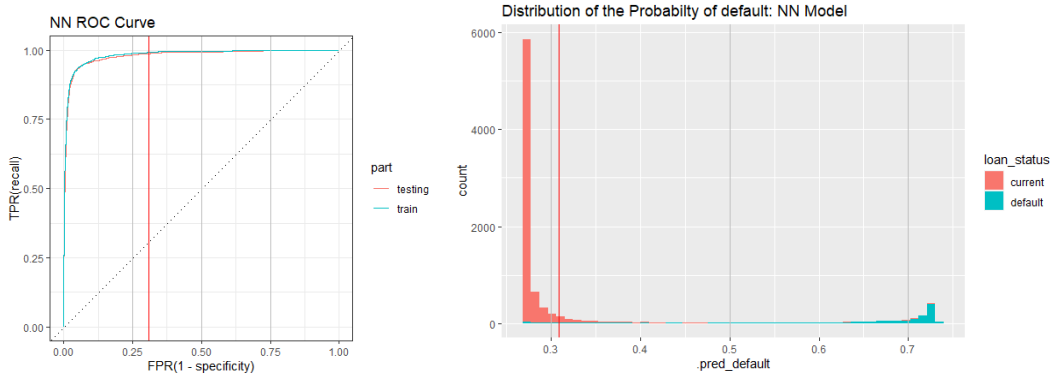


	Accuracy	Log loss	AUC	Precision	Recall	F1
Training	0.9606	0.3607	0.9826	0.8950	0.8336	0.8632
Testing	0.9579	0.3625	0.9790	0.8920	0.8225	0.8575

Operating Range

ROC and Distribution of the probability of default

The desired operating level is at **5% FNR**. At this level, we will incorrectly define 5% positives as negatives, i.e., 5% default as current. The desired operating level is marked as a red v-line, while other common thresholds are marked as gray v-lines.



Operation Chart (precision/recall/FPR/FNR/Threshold)

FNR	Threshold	FPR	Recall (TPR)	Precision
0.01	0.271	0.454	0.992	0.286
0.02	0.278	0.232	0.980	0.438
0.03	0.285	0.166	0.971	0.522
0.04	0.294	0.116	0.961	0.595
0.05	**0.309**	**0.078**	**0.950**	**0.694**
0.06	0.322	0.059	0.940	0.745
0.07	0.334	0.047	0.931	0.782
0.08	0.348	0.040	0.920	0.811
0.09	0.358	0.036	0.908	0.824
0.10	0.371	0.030	0.900	0.833

- Given the context, we choose an FNR at 5%. It means we will accept 5% of the loans which will default as current.
- To achieve this FNR rate, when the predicted default rate is higher than 0.309, we will classify it as default; and we will detect it as current when the predicted default rate is below 0.309
- At this FNR rate, the FPR is 0.078, which means we will define 7.8% of the current loan as default.
- The recall/TPR is 0.950, which means we will successfully identify 95% of the total default loans.

- The precision will be 0.694, which means among all the loans we defined as the default, 69.4% of them will default.

Potential Savings (in Every 1000 Loan Applications):

Accept all loan applications (nowadays): ***1000 * 0.15 * mean (loan amount)***

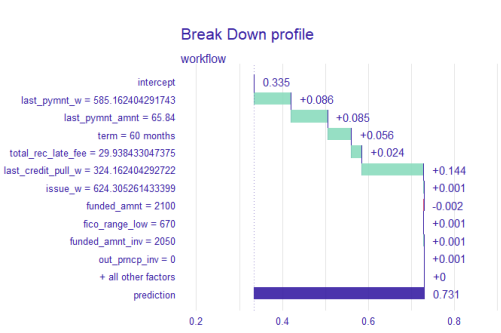
Operating at 5% FNR: (5% will be the default, 7.4% current loan will be rejected and lose interest income): ***1000*0.05*mean (loan amount) + 1000 * 0.078 * mean (interest rate) * mean (loan amount)***

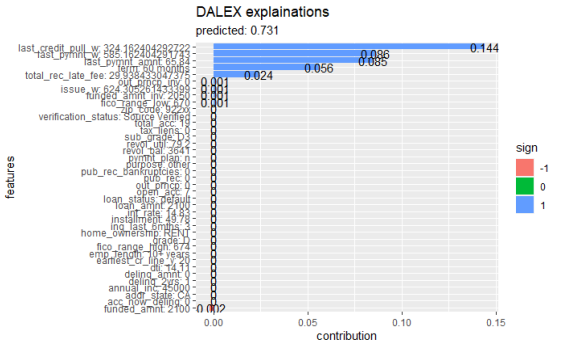
Potential Savings = $1,005,517

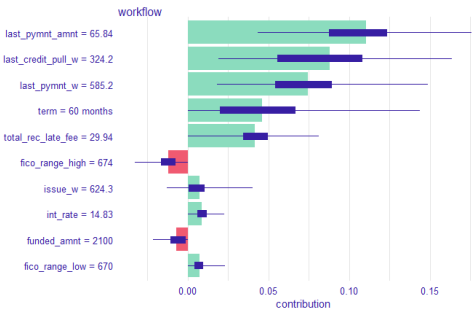
DALEX, SHAPLEY, and VIP

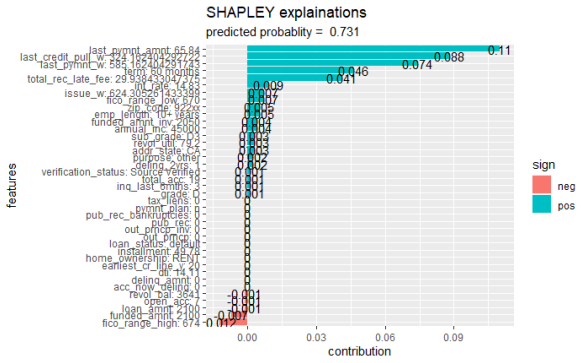
We used 3 methods to interpret the model from a global perspective, VIP, DALEX breakdown, and SHAPLEY, and we used the highest prediction probability to build DALEX and SHAPLEY chart.

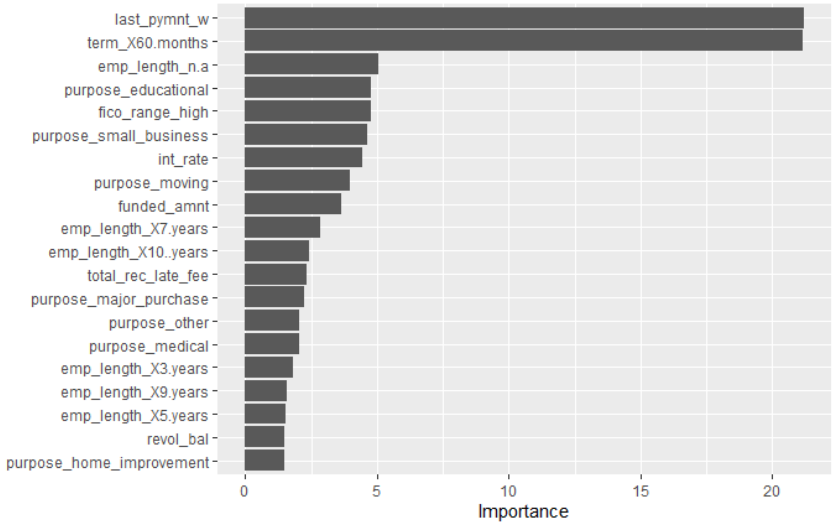
According to the charts below, "last_pymnt_w", "last_pymnt_amnt", "last_credit_pull_w", "term", and "total_rec_late_fee" are important predictors to detect default.







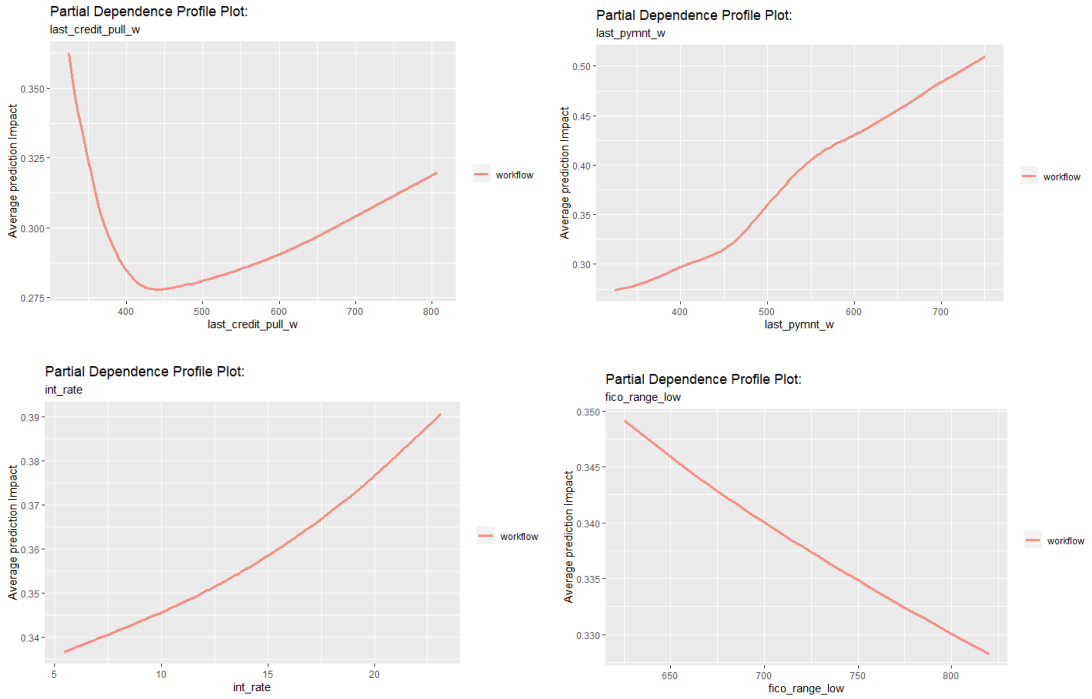


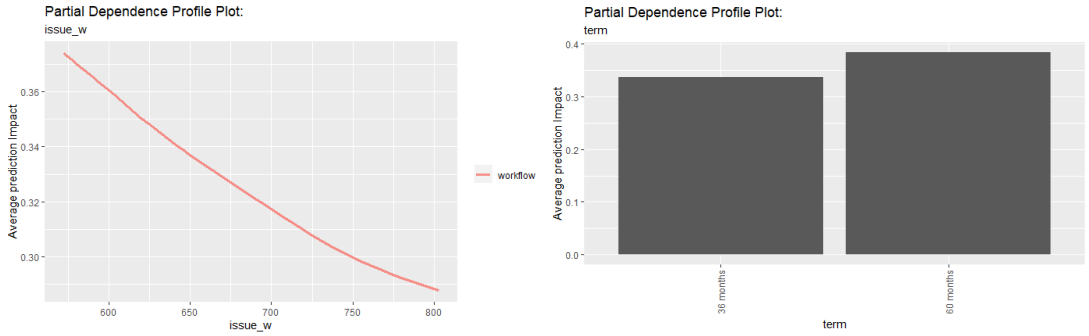


Partial Dependence Plots

We selected top variables from the global explanations above to draw partial dependence plots, in order to study the interaction between one specific predictor and the response.

From the charts below we can see that, when “last_pymnt_w”, “int_rate” affect the response in a positive way, while “fico_range_low” and “issue_w” affect the response in a negative way. “last_credit_pull_w” affect the response in a “U-shape” way. Same as the result from random forest, 60-months term tends to generate a higher possibility of default.



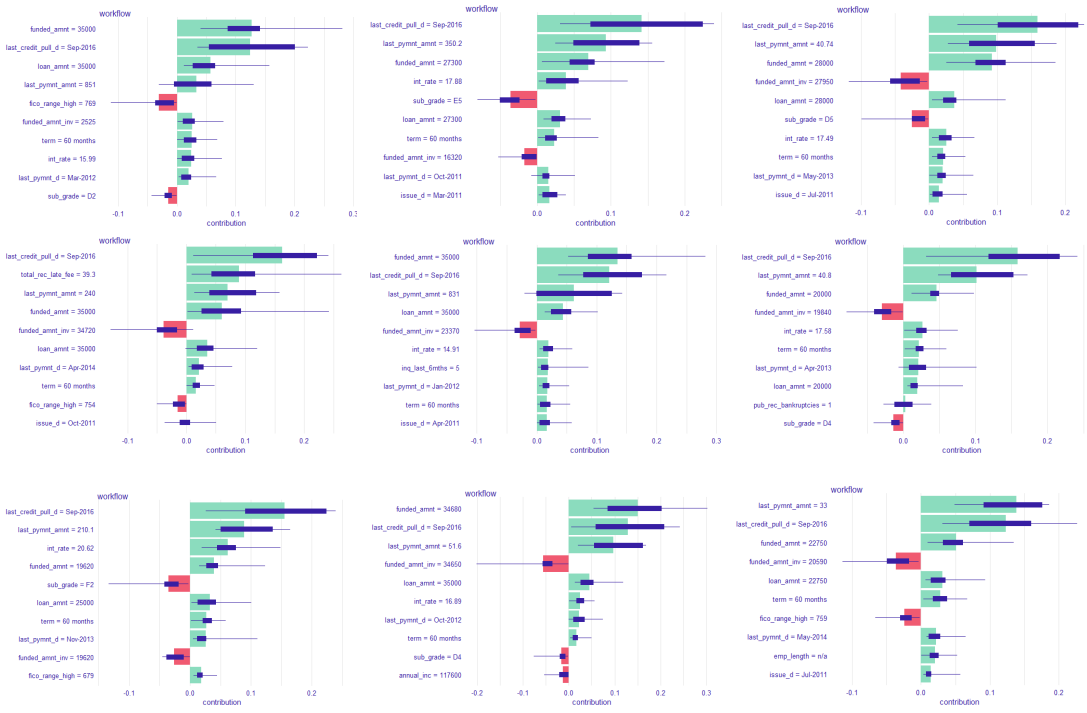


Local Explanation: Top 10 TP FP FN Records

Top 9 TP Records

We have the results of the Top 10 true positive records but we only display 9 of them for a better viewing.

Among those records, lots of them have the same “last_credit_pull_d” which is “Sep-2016”. When “term” is on the board, it’s usually 60-months term. Their “last_pymnt_amnt” is below $300.



Top 10 FR Records

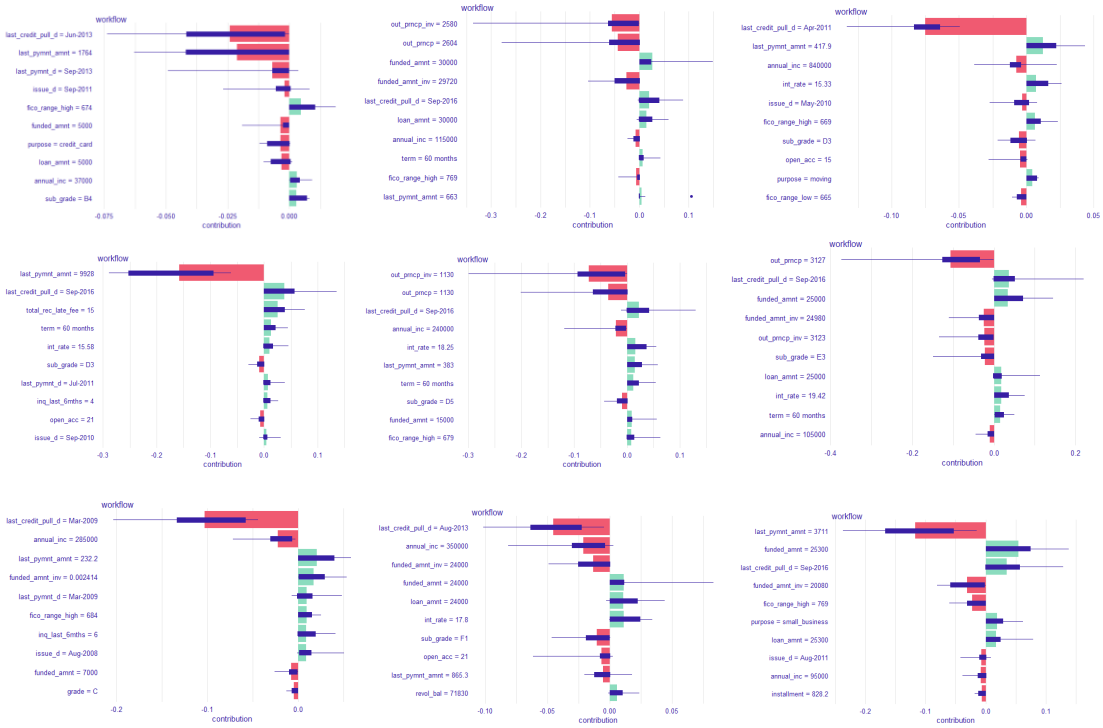
We have the results of the Top 10 false positive records but we only display 9 of them for a better viewing.

Among those records, lots of them have the same “last_credit_pull_d” which is “Sep-2016”. They all have a low “last_pymnt_amnt”.

Top 9 FN Records

We have the results of the Top 10 false negative records but we only display 9 of them for a better viewing.

Among those records, some of them have a higher “annual_inc”, above $84,000, or $200,000. Lots of them have an interest rate between 15% to 20%.



Model Comparing

Metrics and Generating Predictions

	Accuracy	Log loss	AUC	Precision	Recall	F1
Training NN	0.9606	0.3607	0.9826	0.8950	0.8336	0.8632
Testing NN	0.9579	0.3625	0.9790	0.8920	0.8225	0.8575
Training XGB	1.0000	0.0012	1.0000	1.0000	1.0000	1.0000
Testing XGB	0.9786	0.0820	0.9937	0.9497	0.9088	0.9288
Training RF	1.0000	0.0499	1.0000	1.0000	1.0000	1.0000
Testing RF	0.9569	0.1524	0.9842	0.9401	0.7679	0.8453

Basing on the metrics above, we choose XGB as our final model and generate predictions.

```
holdout_score <- predict(xgb_wflow, holdout, type = "prob") %>%
  bind_cols(predict(xgb_wflow, holdout, type = "class")) %>%
  bind_cols(., holdout) %>%
  select(id, .pred_class, .pred_default) %>%
  write_csv("Prediction_EmmaWang.csv")

head(holdout_score)
```