Prediction of 2019 Loan Application Outcomes

Stats 101C
Summer 2023

Team 2

Jun Yu Chen
Anna Deng
Angelina Sun
Eric Xia
Nicole Xu

Introduction

Within the dataset of mortgage loans from Wells Fargo National Bank in 2019, our analysis focuses on understanding the factors influencing the outcomes of loan applications.

According to Trönnberg & Hemlin, “hard information” – including financial details – plays a bigger role in affecting the application decisions (2014). Thus, we hypothesize that variables including property value, income, loan amount are more significant in predicting whether a loan application is approved or denied.

Furthermore, other variables that indicate one’s capacity to repay may be more significant to influence actions taken on the loan application. Therefore, we postulate that variables like housing and loan to value ratio may be important predictors. Lastly, variables like HOEPA status may play a bigger role as a high-cost mortgage a higher risk for lenders, which may affect approval rates.

Through the analysis, we will delve into the predictors affecting loan application outcome, providing valuable insights to lending practices.

Citation:
Trönnberg, C.-C., & Hemlin, S. (2014). Lending decision making in banks: A critical incident study of loan officers. *European Management Journal*, *32*(2), 362–372. https://doi.org/10.1016/j.emj.2013.03.003

Bank of America. Bank of America - Banking, Credit Cards, Loans and Merrill Investing. (n.d.). https://www.bankofamerica.com/smallbusiness/resources/post/factors-that-impact-loan-decisions-and-how-to-increase-your-approval-odds/

Exploratory Data Analysis

Transformation of the Response Variable

In this dataset, the response variable is action_taken, which includes 1, meaning loan originated, and 3, meaning application denied. In order to make clearer visualizations, we transformed the action_taken variable into a categorical variable of True and False. True aligns with the original label “1” and False points to the original label “3.”

Exploration of Potential Relationship between Variables

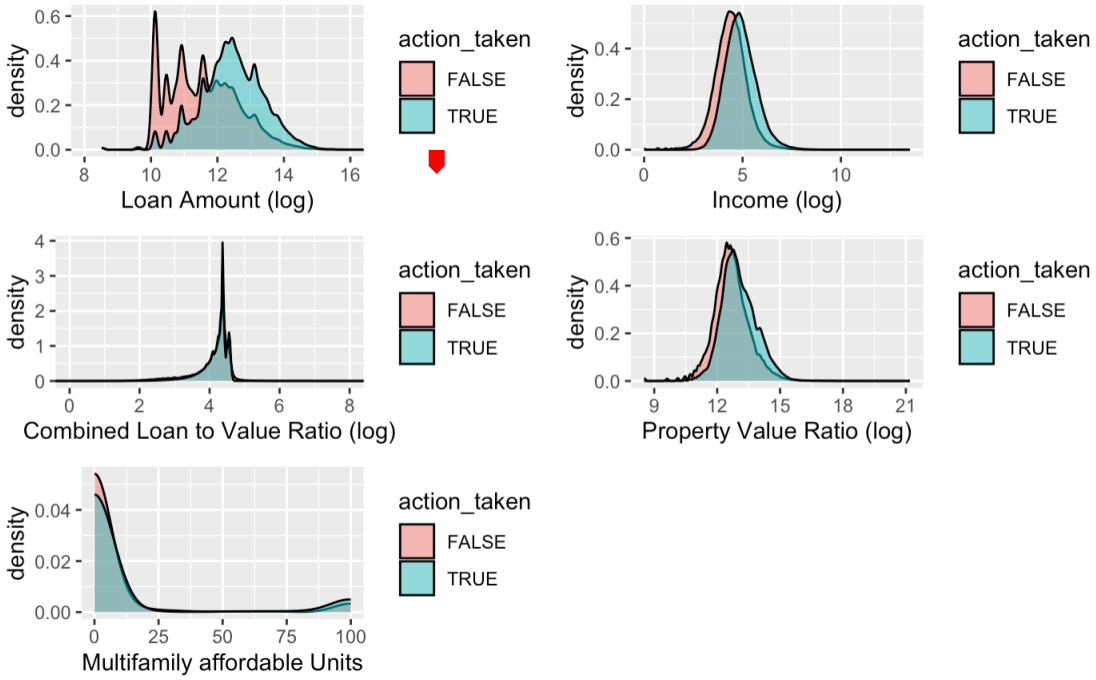


Figure 1: Density Plots of Numeric Variable Distributions by Action Taken

The above graphs provide insights into how actions on loan application differs in regards to the distribution of numeric variables. For the first graph of loan amount distribution by action taken, it is evident that the distribution of loan amount differs with different actions. The distribution of denied applications appears more right skewed and scattered but with less loan amounts on average, compared to the distribution of originated applications. This suggests that approved applications tend to cluster around a certain range of loan amounts centered at a larger value (around 12 after log transformation), while denied applications exhibit greater variations.

For the graphs of Income and Property Value Ratio (after log transformations), the approved loans exhibit higher income and property value ratios on average. These graphs exhibit similar distributions while centered at different values for different actions taken.

For the other graphs, the distribution of approved applications and denied applications show similar patterns, suggesting that these variables may not play a significant role in differentiating between approval and denial decisions. Further analysis may be necessary to uncover factors contributing to action outcomes.

Figure 2: Bar Chart of Loan Term by Action Taken

We cut loan terms into different categories to better explore and identify potential trends. Each bar represents a specific duration of loan term. For example, in the 6 months to 1 year, there are more approved applications, while for longer-term loans, there are more denials. This suggests that loan terms may influence the application outcomes. It does seem to us that with a shorter loan term, the loan is more likely to be approved.

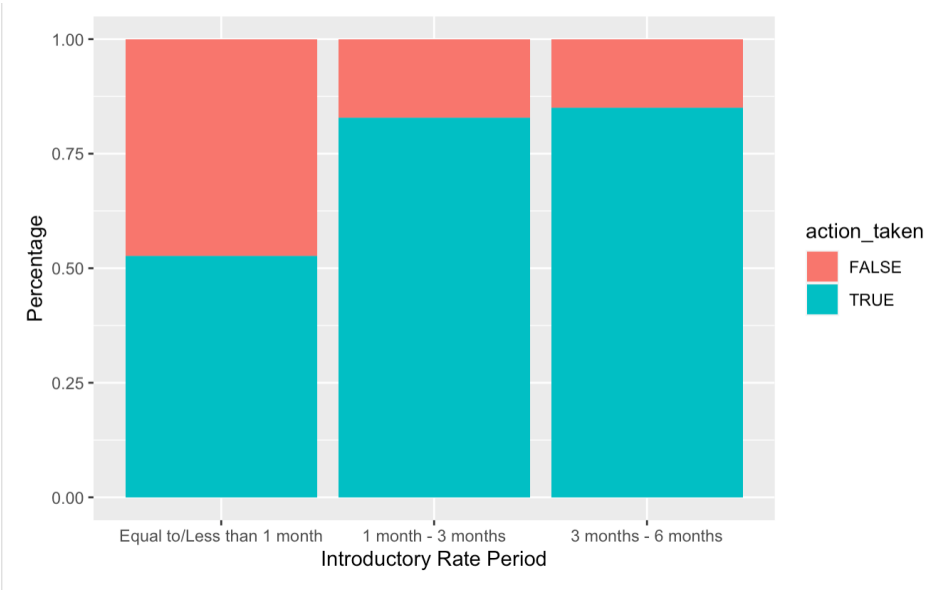
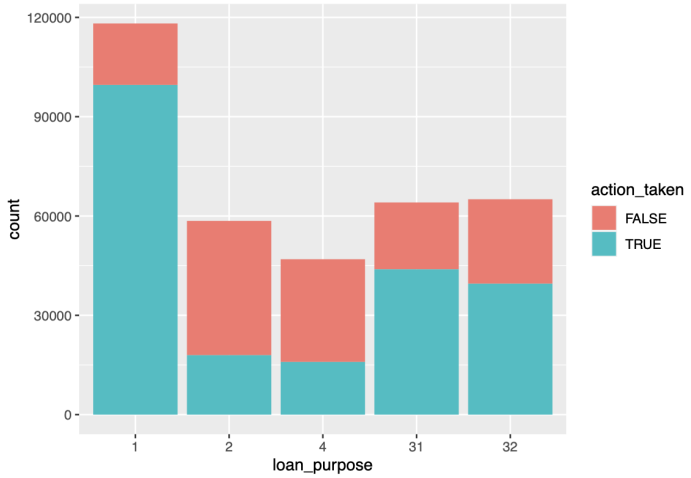


Figure 3: Bar Chart of Introductory Rate Period by Action Taken

Similarly, we cut the introductory rate period into different categories and found that loan applications with longer introductory rate periods tend to have more approved outcomes compared to introductory rate periods with less than/equal to 1 month that have mixed outcomes. This suggests that the duration of introductory periods may affect the application outcomes.

After transforming most categorical variables from the numeric form, we also completed a thorough analysis of the relationship between all categorical variables and the response variable. Among all the relationships, there are a few variables that show strong patterns with the action-taken variable.

For example, different loan purposes may lead to different possibilities of a successful loan. Here we take the loan_purpose variable as an example, where 1, 2, 4, 31, and 32 indicate home purchase, home improvement, other purpose, refinancing, and Cash-out refinancing respectively. In the following graph, we can see that it is more likely to initiate an approved loan if the purpose is home purchase (1) than the case when the purpose is refinancing (31), and the loan is most likely to be denied if the purpose is home improvement (2) or other (4). Therefore, this variable may have a strong predictive power when we want to predict whether a loan is accepted when we are given a new dataset.

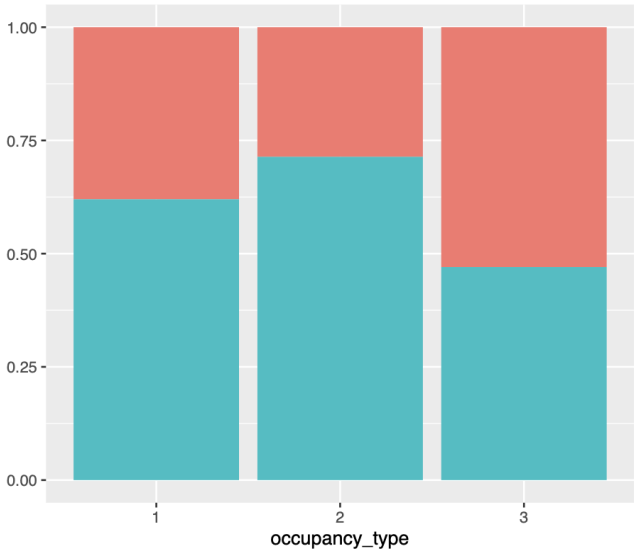



```
##   loan_purpose pct_action_take
##   <fct>                  <dbl>
## 1 1                      0.843
## 2 2                      0.307
## 3 4                      0.339
## 4 31                     0.685
## 5 32                     0.608
```

Figure 4: Bar Chart of loan_purpose by action taken and table of loan_purpose and percentage action taken

Different occupancy types may also lead to different success rates of loaning. For the occupancy_type variable, 1 means principal residence, 2 means second residence, and 3 means investment property. Since the counts differ between different occupancy types to a great extent, we used percentage as the y variable for the following graph for visual purpose. In the following charts, we can see that it is more likely to successfully be approved for the loan if the occupancy type is second residence (2), indicating that financial institutions may perceive them as lower-risk investments. An occupancy type of principal residence (2) has a lower rate of

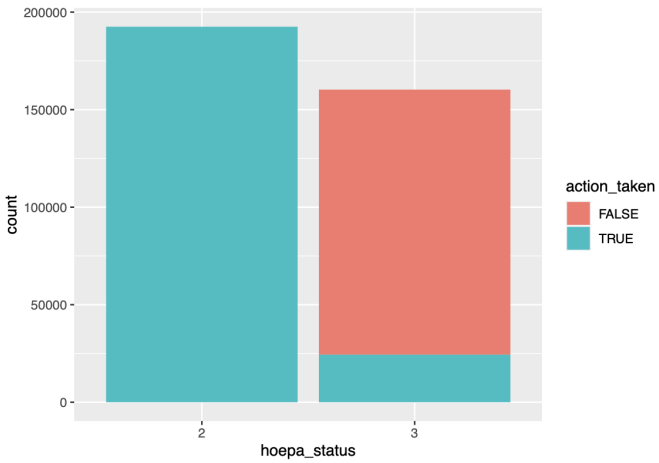
success, and it is most difficult to be approved for the loan if the occupancy type is investment property (3), possibly due to the greater financial risks associated with them. Hence, this variable might exhibit strong predictive capability when we attempt to predict loan approval within a fresh dataset.




```
##   occupancy_type pct_action_take
##   <fct>                    <dbl>
## 1 1                        0.620
## 2 2                        0.714
## 3 3                        0.470
```

Figure 5: Bar Chart of occupancy_type by action taken and table of occupancy_type and percentage action taken

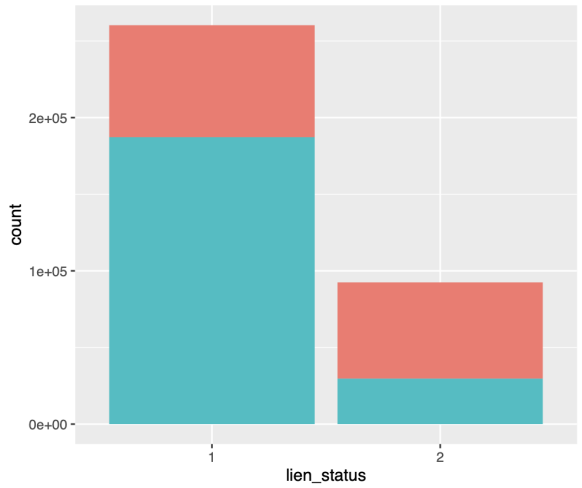
Among all the categorical variables, hoepa_status seems to possess the strongest predictive power and lien_status also seems to possess a very strong predictive power. For hoepa_status, 2 indicates not a high-cost mortgage and 3 indicates not applicable. According to the following graphs, we can see that within the training data, all the loan applications that are not a high-cost mortgage are approved while nearly 85% of the loan applications with an unclear hoepa status are denied. This indicates a very strong predictive power of hoepa_status to forecast whether a loan will be successful or not.




```
##   hoepa_status pct_action_take
##   <fct>                   <dbl>
## 1 2                       1
## 2 3                       0.152
```

Figure 6: Bar Chart of hoepa_status by action taken and table of hoepa_status and percentage action taken

For lien_status, 1 indicates first lien loan while 2 indicates subordinate lien loan. In the following graphs, we can tell that if the loan is secured by a first lien, about 70% of the loans are approved. However, if the loan is secured by a subordinate lien, only about 30% of the loans are approved. Therefore, although there is high uncertainty in the testing data, it is likely that the loans secured by a first lien will be more likely to be approved compared to the loans that are secured by a subordinate lien. This pattern will facilitate our future prediction of whether a loan will be approved.




```
##   lien_status pct_action_take
##   <fct>                 <dbl>
## 1 1                     0.719
## 2 2                     0.321
```

Figure 7: Bar Chart of lien_status by action taken and table of lien_status and percentage action taken

Similarly, we explored the relationship between all predictor variables and the response variables and developed a clearer understanding of the internal relationships within the dataset, which will facilitate further steps including preprocessing and model building.

Transformation of Predictor Variables

A tibble: 34 × 2

variable <chr>	missing_percentage <dbl>
ethnicity_of_applicant_or_borrower_1	0.69
ethnicity_of_applicant_or_borrower_2	95.46
ethnicity_of_applicant_or_borrower_3	99.92
ethnicity_of_applicant_or_borrower_4	100.00
ethnicity_of_applicant_or_borrower_5	100.00
ethnicity_of_co_applicant_or_co_borrower_1	0.23
ethnicity_of_co_applicant_or_co_borrower_2	98.15
ethnicity_of_co_applicant_or_co_borrower_3	100.00
ethnicity_of_co_applicant_or_co_borrower_4	100.00
ethnicity_of_co_applicant_or_co_borrower_5	100.00
race_of_applicant_or_borrower_1	0.43
race_of_applicant_or_borrower_2	94.36
race_of_applicant_or_borrower_3	99.67
race_of_applicant_or_borrower_4	100.00
race_of_applicant_or_borrower_5	100.00
race_of_co_applicant_or_co_borrower_1	0.14
race_of_co_applicant_or_co_borrower_2	97.26
race_of_co_applicant_or_co_borrower_3	99.86
race_of_co_applicant_or_co_borrower_4	100.00
race_of_co_applicant_or_co_borrower_5	100.00
age_of_applicant_62	0.24
age_of_co_applicant_62	52.91
income	5.73
total_points_and_fees	100.00
prepayment_penalty_term	94.43
combined_loan_to_value_ratio	3.42
loan_term	0.10
introductory_rate_period	93.28
property_value	1.39
multifamily_affordable_units	99.73
automated_underwriting_system_2	84.52
automated_underwriting_system_3	87.08
automated_underwriting_system_4	100.00
automated_underwriting_system_5	100.00

34 rows

Figure 8: Missing Data Percentages

According to the table above, there are significant missing percentages in certain variables. Variables with more than 80% missing data are considered for removal, for the reason that they are unlikely to provide meaningful or reliable insights for statistical analysis or predictive modeling. The lack of sufficient data points can lead to biased or inaccurate conclusions, and may also add unnecessary complexity to the data preprocessing or analysis stages. Additionally, imputing such a high percentage of missing values could introduce substantial error or bias into the dataset.

Apart from the missing values, we noticed that ethnicity and race each has 5 different columns and the options are layered within each one. For example, broader categorizations like 'hispanic_or_latino' and 'not_hispanic_or_latina' coexist with more specific distinctions such as 'Mexican' and 'Puerto Rican'. Thus, we've created new variables to simplify and categorize ethnicity information.

For instance, "Is_hispanic_applicant" and "is_hispanic_co_applicant" denotes Hispanic ethnicity status, which contains 0, 1 to represent whether the applicant/ co-applicant is hispanic, and an ethnicity_applicant column that contains the numbers 0-3 to represent the specific ethnicity groupings.

For race, we again removed the original columns, and created a new race column that only contains the broader race groupings (White, Asian, Black, etc.), excluding those that are too specific for only certain races (Asian Indian, Chinese, Filipino, etc.). This process was applied to both applicant and co-applicant race columns.

Ultimately, we consolidated the 20 columns related to ethnicity and race down to just 6. These transformations help to maintain essential information while effectively managing missing data.

Preprocessing

From our exploratory data analysis, we found that contrary to the election dataset used for the regression competition, which predominantly consists of numeric predictors, the loan register dataset presents a unique challenge with its numerous categorical variables. Our team has attempted several feature engineering, encoding, and selection techniques to overcome this challenge which are outlined below.

Encoding categorical variables

Dummy encoding

Given the numerous categorical variables, particularly those with high cardinality, traditional encoding methods like one-hot or dummy encoding prove suboptimal, as they would drastically increase the number of features. However, for low cardinality features such as occupancy_type and loan_purpose, using dummy encoding might potentially be advantageous. Thus, in one of our logistic regression models, we used an alternative recipe (explained below) with dummy encoding practice as well.

Label encoding

Label encoding transforms each category of a categorical variable into a distinct integer, preserving the original number of features; this is achieved by first converting the variable into a

factor and subsequently into an integer: for a categorical variable x, we used step_mutate(x=as.integer(as.factor(x))).

A limitation of this method is that for categorical variables like ethnicity and race, which lack inherent ordinality, the assigned integers can not only be arbitrary, but mislead certain models into picking up nonexistent patterns. Despite its inherent drawbacks, we settled on label encoding due to its straightforward implementation, ability to preserve feature numbers, and solid performance during our tests. We compared the performance of logistic regression using label encoding and dummy encoding, and it appears that its performance (measured by F1) didn't differ significantly. Thus, for the sake of computational efficiency and convenient implementation, we opted for label encoding for most of our categorical variables.

Other encoding methods

We also explored various other encoding strategies such as target encoding, count encoding, and glm encoding. Target encoding represents each category using the mean of the target variable for that category. Count encoding signifies each category based on its occurrence count. Glm encoding fits a generalized linear model to each predictor and target and uses the coefficient as the encoding for that predictor. However, there wasn't a direct way to implement target and count encoding in the recipe, and even though glm encoding could be achieved through step_lencode_glm(), it did not outperform label encoding and potentially generated bugs during model training. Therefore, we did not employ these methods in our final recipes.

In the end, most of our recipes used label encoding as the encoding method, while one recipe for logistic regression used a mixture of dummy encoding and label encoding.

Feature selection

Baseline Model Architecture

Our baseline model incorporates an extensive range of engineered features, including aggregated variables for ethnicity and race, label-encoded and dummy-encoded variables, as well as existing numerical attributes. We removed variables that contain more than 80% missing values (NAs), as their inclusion may lead the model to infer spurious patterns. Further, the step_zv(all_predictors()) filter is applied to eliminate variables that contain only a single unique value, streamlining the feature set for subsequent analysis.

Variable Selection

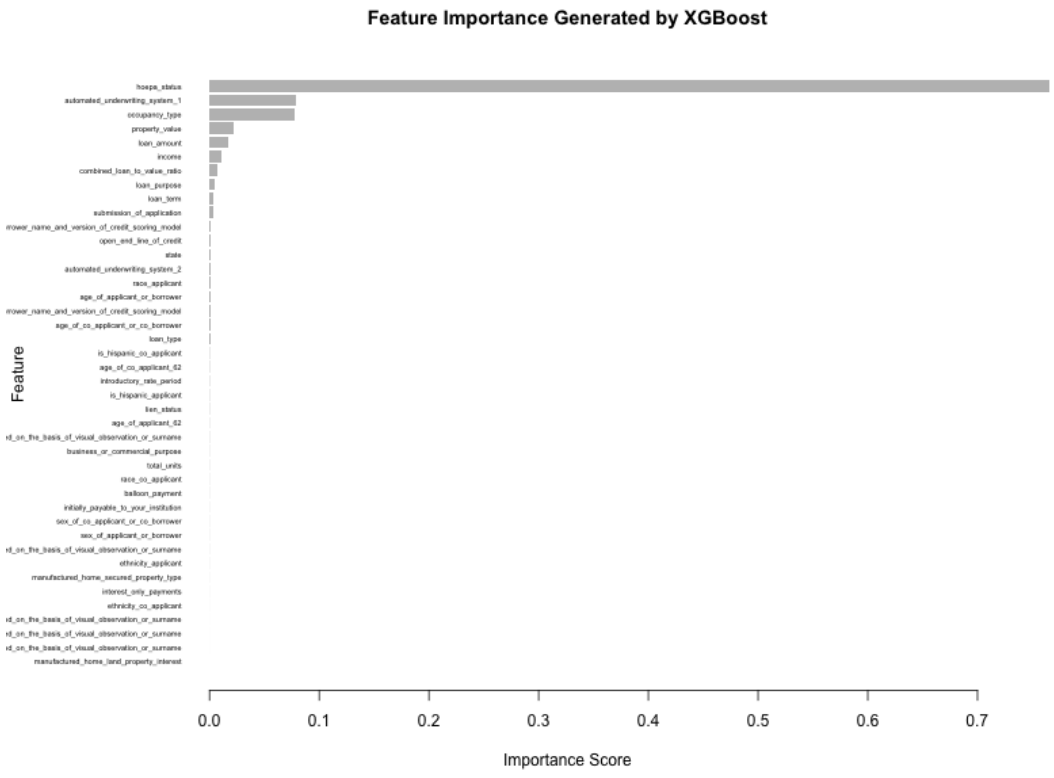
Given that our baseline model comprises 53 features, it becomes imperative to analyze which variables play more influence on prediction performance. To selectively identify these key features while minimizing model noise, we deploy the Gradient Boosting Machine (GBM) algorithm to rank features based on their contribution to predictive accuracy.

Mechanics of the Gradient Boosting Machine (GBM)

The Gradient Boosting Machine employs ensemble learning, building on the principle of boosting. It strategically combines weak learners—commonly decision trees—to create a robust predictive model. The algorithm operates iteratively, each time adding a new tree that aims to correct the inaccuracies made by its predecessors. Through this iterative refinement, GBM rapidly and effectively identifies the features most instrumental in enhancing prediction accuracy.

Results and Sensitivity Analysis

Upon running the Gradient Boosting Machine (GBM), we obtained a feature importance matrix. The 'Gain' column measures a feature's impact on model improvement before a node split. 'Cover' and 'Frequency' highlight the number of observations affected by each feature. The 'Importance' column consolidates these values, offering a comprehensive score for each feature's relevance.



Interpretation of GBM Feature Importance

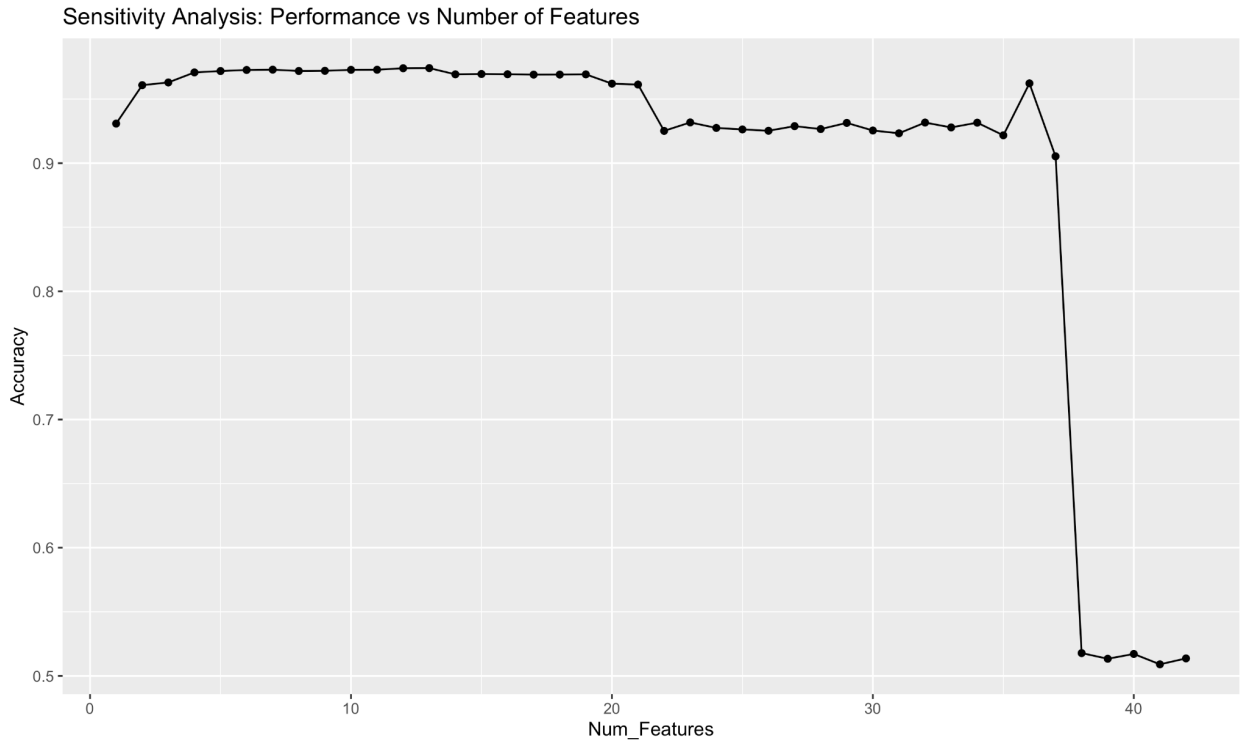
From the Gradient Boosting Machine (GBM) algorithm, we acquired an importance matrix that systematically ranked each variable based on its contribution to predictive accuracy. Intriguingly, the variable heopa_status scored an importance value exceeding 0.7, substantially outweighing the impact of any other variables in the model. This elevated importance of heopa_status aligns well with our prior exploratory data analysis (EDA). We observed that when heopa_status is set to 2, the associated action_taken is always true. This could indicate that heopa_status serves as a strong predictor for the outcome variable, potentially capturing some underlying policy-driven factors that directly influence the loan application's outcome. However, it is crucial to exercise caution while interpreting this result. The observed pattern may be specific to our training dataset and may not hold in a generalized context. Over-reliance on a single feature, especially one so influential, can introduce model bias and impact its generalizability.

	Feature	Gain	Cover	Frequency	Importance
1	hoepa_status	7.651920e-01	1.663164e-01	0.0127737226	7.651920e-01
2	automated_underwriting_system_1	7.895963e-02	8.052368e-02	0.0237226277	7.895963e-02
3	occupancy_type	7.696295e-02	8.348956e-02	0.0358880779	7.696295e-02
4	property_value	2.156455e-02	1.250350e-01	0.1140510949	2.156455e-02
5	loan_amount	1.645399e-02	5.571759e-02	0.0924574209	1.645399e-02
6	income	1.099235e-02	1.117483e-01	0.1654501217	1.099235e-02
7	combined_loan_to_value_ratio	7.306893e-03	9.106317e-02	0.1408150852	7.306893e-03
8	loan_purpose	5.217762e-03	2.147235e-02	0.0395377129	5.217762e-03
9	loan_term	3.789985e-03	4.826187e-02	0.0553527981	3.789985e-03
10	submission_of_application	3.581410e-03	5.987577e-03	0.0133819951	3.581410e-03
11	co_applicant_or_co_borrower_name_and_version_of_c...	1.142719e-03	4.807915e-03	0.0142944039	1.142719e-03
12	open_end_line_of_credit	1.028488e-03	3.661721e-03	0.0066909976	1.028488e-03
13	state	9.133576e-04	3.451099e-02	0.0608272506	9.133576e-04
14	automated_underwriting_system_2	9.050131e-04	1.166631e-02	0.0115571776	9.050131e-04
15	race_applicant	6.755612e-04	9.331072e-03	0.0209854015	6.755612e-04
16	age_of_applicant_or_borrower	6.088938e-04	1.205463e-02	0.0295012165	6.088938e-04
17	applicant_or_borrower_name_and_version_of_credit_s...	5.238020e-04	6.674265e-03	0.0176399027	5.238020e-04
18	age_of_co_applicant_or_co_borrower	3.915116e-04	1.598903e-03	0.0167274939	3.915116e-04

Additional highly-ranked variables in terms of importance include automated_underwriting_system_1, occupancy_type, property_value, loan_amount, and income. This ranking aligns well with an in-depth understanding of loan application processes. For example, automated_underwriting_system_1 employs an automated system to provide a fairly objective estimate of the loan's approval likelihood. Occupancy_type, whether a principal residence or investment property, carries varying risk levels that influence loan approval rates. Similarly, property_value and loan_amount are financial metrics that directly affect the associated loan risk. Higher values in either category may signal increased stakes, thus influencing decision-making. Lastly, an applicant's income serves as a key indicator of repayment capability, which naturally impacts loan application outcomes. The key variables identified align with our discoveries in the exploratory analysis section as well.

Despite the interpretation of key variables, determining the optimal number of features to include posed a challenge. To tackle this issue, we conducted a sensitivity analysis that iteratively runs a prototype generalized linear model (GLM) on varying subsets of the most

important features, which enabled us to monitor shifts in model accuracy as different subsets of features were selected.



As illustrated in the accompanying graph, the model's binary classification accuracy plateaus at a transitional point with around 21 predictors. Accordingly, our refined model—referred to as 'recipe_21'—includes only these top 21 features for optimal performance.

Imputation of Missing Values

Our dataset contained missing values in both numerical and categorical variables. To handle this, we used median imputation for numerical variables and mode imputation for categorical variables.

- Numerical Variables: Median imputation was used because it is less sensitive to outliers.
- Categorical Variables: Mode imputation was chosen to maintain the existing category distribution.

These methods were selected for their robustness and simplicity, effectively preparing the dataset for subsequent analysis.

Final Recipes

- baseline_recipe: all transformed features without selection
- recipe_21: selected top 21 features
- scaled_recipe: baseline_recipe with feature imputation

- scaled_recipe_21: recipe_21 with feature imputation
- Alternative recipe: baseline_recipe with partial dummy encoding

Models

Candidate Models

Our team has conducted tests on a variety of potential models, each with a unique mix of model architectures, algorithms, and tuning parameters. Our attention has been especially directed towards tree-based algorithms and logistic regression techniques. We also experimented with Support Vector Machines (SVM) and K-Nearest Neighbors (KNN) algorithms to assess their suitability for our analysis. However, we found that these models were not well-suited for handling our large dataset, which comprises over 100,000 instances. Specifically, SVMs tend to have a high computational cost and can become inefficient as the dataset size increases, making it less feasible for real-time or quick iterative evaluations. Similarly, KNN is computationally expensive and time-consuming when dealing with large datasets, as it needs to compute the distance to every point in the dataset for each prediction. Therefore, these models were not ideal choices for our specific use case, given the dataset's size and complexity.

Tree based models

We explored tree-based algorithms because of their capacity to represent intricate decision-making boundaries. These algorithms generate predictions by navigating through a set of decision criteria arranged in a tree-like format.

Random Forest

The ensemble approach enhances the capabilities of a solitary decision tree by resampling the data and constructing multiple trees, each trained on a random subset of features. We have two variants: one called 'rf', which employs a scaled-down basic algorithm, and another named 'rf_21', which is configured to use 21 handpicked features. Critical tuning parameters in these models are 'mtry', 'trees', and 'min_n'.

Extreme Gradient Boosting (XGBoost)

The sophisticated gradient boosting technique employs a chain of decision trees, each designed to amend the mistakes made by the preceding ones. Similar to the Random Forest approach, we've developed two configurations: 'xgb' uses the original, unscaled set of features, while 'xgb_21' employs a subset of 21 chosen features. Key tuning parameters such as 'trees', 'min_n', 'tree_depth', 'learn_rate', and 'loss_reduction' are fine-tuned for optimal performance.

LightGBM

We've also experimented with another gradient-boosting method, originated by Microsoft, which prioritizes both scalability and computational efficiency. We have two variations: 'lightgbm', which relies on the complete scaled baseline set of features, and 'lightgbm_21', which utilizes a selection of 21 scaled features. Important tuning parameters such as 'trees', 'min_n', 'tree_depth', 'learn_rate', and 'loss_reduction' are adjusted for peak performance.

Logistic Regression - Elastic Net

In addition to conventional logistic regression, we've incorporated Elastic Net, a method that blends L1 and L2 regularization techniques. We have three implementations: 'en' utilizes the entire scaled baseline feature set, 'en_21' employs a subset of 21 handpicked features, and 'en_a' combines the scaled baseline feature set with a 'step_dummy' preprocessing step to evaluate the baseline's efficacy. Key hyperparameters like 'penalty' and 'mixture' are fine-tuned to achieve the best performance.

Summary Table of Candidate Models

Model Identifier	Model Type	Engine	Recipe	Hyperparameters
xgb	XGBoost	"xgboost", objective = "binary:logistic"	baseline_recipe	trees = 1760, min_n = 8, tree_depth = 14, learn_rate = 0.000379, loss_reduction = 0.00204
xgb_21	XGBoost	"xgboost", objective = "binary:logistic"	recipe_21	trees = 1847, min_n = 11, tree_depth = 12, learn_rate = 1.63e-2, loss_reduction = 6.98e-4
rf	Random Forests	"ranger", importance = "impurity"	scaled_recipe	mtry = 24, trees = 786, min_n = 14
rf_21	Random Forests	"ranger", importance = "impurity"	scaled_recipe_21	mtry = 7, trees = 533, min_n = 22
lightgbm	LightGBM	lightgbm	scaled_recipe	trees = 1847, min_n = 11, tree_depth = 12,

				learn_rate = 0.01627485, loss_reduction = 0.0006982271
lightgbm_21	LightGBM	lightgbm	scaled_recipe_21	trees = 1847, min_n = 11, tree_depth = 12, learn_rate = 0.01627485, loss_reduction = 0.0006982271
enet	Logistic Regression	glmnet	scaled_recipe	penalty = 4.950587e-08, mixture = 0.9987151
enet_21	Logistic Regression	glmnet	scaled_recipe_21	penalty = 1.313373e-10, mixture = 0.9762183
enet_a	Logistic Regression	glmnet	Alternative recipe (baseline recipe with partial dummy encoding)	penalty = 3.392973e-07, mixture = 0.9998133

[Table: Details of Individual Models]

Model Tuning

We employed 'tune_bayes' for hyperparameter optimization, which utilizes Bayesian optimization techniques. Unlike random or exhaustive search methods, this algorithm constructs a probabilistic model based on past evaluations to forecast which hyperparameters are most likely to deliver optimal results. This model is updated iteratively, enabling the algorithm to focus on the most promising areas within the hyperparameter landscape, thereby potentially speeding up the search and improving accuracy.

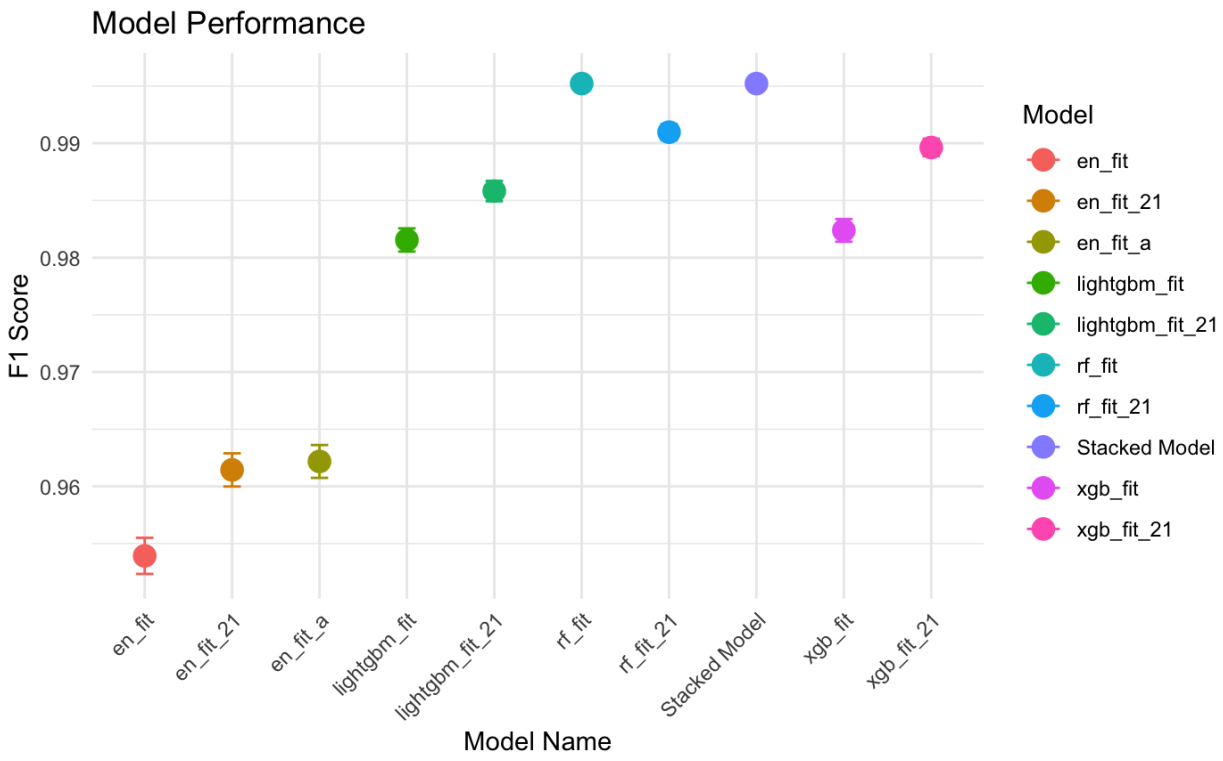
For our final candidate models, we chose Bayesian optimization with an initial set of 10 random parameter configurations, followed by 80 total iterations, including those initial 10. To assess the performance of various models and their hyperparameters, we used 10-fold cross-validation. We opted for the F1 score as our evaluation metric to balance both precision and recall effectively. If the performance of the model does not improve in 10 iterations, the tuning process will terminate and retain the best performing hyperparameters. This approach minimizes the risk of selecting hyperparameters that overfit to a specific data subset and offers a more reliable performance estimate for each model and its parameters.

Model Evaluation

As previously stated, we employed 10-fold cross-validation for assessing our prospective models. Once each model was fine-tuned, we included them in a workflow set for easier comparison. Ultimately, the random forest and stacked models emerged as the top performers, with their names highlighted in bold. The stacked model was assembled using the predictions from the individual models, and it was integrated via an untuned random forest model. The concluding cross-validation outcomes for each candidate model are compiled in the subsequent table and graphical representation:

Model Identifier	**F1**	**SE of F1**
xgb	0.9823840	0.0009905057
xgb_21	0.9896285	0.0007628147
rf	**0.9952209**	**0.0005192709**
rf_21	0.9909712	0.0007122127
lightgbm	0.9815434	0.0010134302
lightgbm_21	0.9858182	0.0008902808
enet	0.9539233	0.0015785572
enet_21	0.9614385	0.0014497759
enet_a	0.9621779	0.0014363615
Stacked model	**0.9952209**	**0.0005192710**

[Table: Evaluations of Individual Models]



[Figure: Plots of Evaluations of Individual Models]

Discussion of Final Models

We selected the stacked model as our ultimate choice based on its superior F1 score and minimal standard error. Although the random forest model demonstrated equivalent performance in these metrics, we opted for the stacked model. Our rationale is that the stacked model, by leveraging diverse model architectures, is likely better equipped to identify intricate patterns when making predictions on test data.

Stacking Mechanism

In our stacked ensemble, we employ Random Forests as the meta-model to consolidate predictions from an array of base models, namely XGBoost, LightGBM, Logistic Regression, and Random Forests. Due to their non-parametric nature, Random Forests can capture intricate relationships within the data and are less prone to overfitting, making them a suitable choice for a meta-model designed to amalgamate predictions from diverse base models.

Initially, these base models are trained on the original training dataset, and their subsequent predictions act as the "features" for the Random Forest meta-model. To generate the final predictions, new data points are fed through the base models, and their resultant predictions serve as inputs to the Random Forest meta-model. By integrating these base models in such a

way, we construct a high-performance, unified final model proficient at tackling the complex aspects of the regression problem we're addressing.

Strengths

The stacking ensemble boosts predictive power by integrating the advantages of its individual base models. This not only taps into the unique strengths of each underlying model but also diminishes the potential for overfitting, thus enhancing the model's applicability to new data. One notable merit is the overall resilience of the assembled model. It gains this by counterbalancing the limitations and inaccuracies of its component models, resulting in a more dependable and stable final product. This resilience is amplified by the varied approaches to feature analysis among the base models. This allows the ensemble to capture a diverse range of data relationships, including those that may not be linear and could be overlooked by a singular model.

As for efficiency and scalability, the final model benefits from the high-speed algorithms of XGBoost and LightGBM. These are built for rapid data processing and can efficiently manage large datasets. They also support parallel computation, which expedites the learning process. The ensemble's flexibility is another plus, capable of being tailored to different predictive scenarios and allowing for the individual calibration of its component and overarching models. This opens up several avenues for performance enhancement. Although ensembles generally trade off some level of transparency, the inclusion of base models that can generate confidence levels adds a degree of interpretability and insight into the prediction outcomes.

When compared to a scenario where a lasso regression model might act as the aggregating meta-model, the use of a random forest could introduce a richer array of predictive elements, thereby potentially amplifying the model's prediction accuracy.

Weaknesses

While the stacking ensemble model brings several advantages to the table, it's not without its drawbacks. One of the main issues is the computational resources it demands. The requirement to train multiple constituent models, in addition to a meta-model, makes the approach both time-intensive and computationally taxing. This could be a hurdle in situations that call for real-time predictions or quick model rollouts. The tuning process is also complex, adding another layer of difficulty to optimization efforts.

Another downside is the model's limited interpretability. The combined complexity of the base models within the ensemble can make it challenging to understand how decisions are made, which might be a problem in contexts that require transparent decision-making or feature significance analysis. The model's intricate architecture also complicates deployment and ongoing management, requiring specialized expertise and infrastructure for optimal functioning.

Data availability is another issue. The model's need to train multiple base models generally necessitates a larger dataset for effective performance. A smaller dataset could lead to

overfitting, even given the model's built-in safeguards against it. Additionally, there's the risk of redundancy if the base models generate similar outputs, which could diminish the ensemble's overall effectiveness. Lastly, the selection and tuning of the meta-model are crucial steps; a poorly chosen or improperly optimized meta-model can offset the benefits of the ensemble approach, potentially resulting in less-than-ideal performance.

Possible Improvements

Meta-Model

An interesting opportunity for enhancing the model centers on the choice of the meta-model. So far, our team has mainly employed conventional methods to aggregate predictions from base models. Investigating non-traditional meta-models, such as specialized decision trees or alternative ensemble techniques, could potentially lead to marked gains in the model's predictive accuracy.

Additional Data Sources

For boosting the accuracy of our predictions, incorporating a broader array of data variables could be highly beneficial. Adding the types of data listed below could enhance the effectiveness of our model:

- Credit Score: A critical metric that lenders use to evaluate the creditworthiness of an applicant.
- Employment History: Duration of employment could be a predictor for stability.
- Education Level: Higher education levels may correlate with higher income, affecting the ability to repay.

Integrating these extra layers of data into our models could enable us to more effectively grasp the intricate aspects of the phenomena we aim to forecast. This would likely enhance the resilience and precision of our models.

Appendix I: Final Annotated Script

[Note: The scripts for hyperparameter tuning, fitting models and the models required for the script below can be accessed here.]

```
set.seed(42)

library(tidymodels)
library(xgboost)
library(yardstick)
library(dials)
library(recipes)
library(stacks)
```

```
library(readr)
library(bonsai)
library(lightgbm)
library(kknn)
library(randomForest)
library(kernlab)

# load train set
train_data <- read.csv('train.csv')
train_data = train_data %>% mutate(action_taken =
as.factor(action_taken))
test_data <- read.csv('test.csv')
xgb_fit <- readRDS("xgb_fit.rds")
xgb_fit_21 <- readRDS("xgb_fit_21.rds")
rf_fit <- readRDS("rf_fit.rds")
rf_fit_21 <- readRDS("rf_fit_21.rds")
lightgbm_fit <- readRDS("lightgbm_fit.rds")
lightgbm_fit$fit$fit$fit <- readRDS.lgb.Booster("lgbm_booster.rds")
lightgbm_fit_21 <- readRDS("lightgbm_fit_21.rds")
lightgbm_fit_21$fit$fit$fit <-
readRDS.lgb.Booster("lgbm_booster_21.rds")
en_fit <- readRDS("en_fit.rds")
en_fit_21 <- readRDS("en_fit_21.rds")
en_fit_a <- readRDS("en_fit_a.rds")

train_data$pred_xgb <- predict(xgb_fit, new_data=train_data)
train_data$pred_xgb_21 <- predict(xgb_fit_21, new_data=train_data)
train_data$pred_rf <- predict(rf_fit, new_data=train_data)
train_data$pred_rf_21 <- predict(rf_fit_21, new_data=train_data)
train_data$pred_lightgbm <- predict(lightgbm_fit, new_data=train_data)
train_data$pred_lightgbm_21 <- predict(lightgbm_fit_21,
new_data=train_data)
train_data$pred_en <- predict(en_fit, new_data=train_data)
train_data$pred_en_21 <- predict(en_fit_21, new_data=train_data)
train_data$pred_en_a <- predict(en_fit_a, new_data=train_data)

# Create a new data frame to hold the predictions and the actual
target
train_meta_data <- data.frame(
  pred_xgb = train_data$pred_xgb,
  pred_xgb_21 = train_data$pred_xgb_21,
  pred_rf = train_data$pred_rf,
```

```
  pred_rf_21 = train_data$pred_rf_21,
  #pred_knn = train_data$pred_knn,
  #pred_knn_21 = train_data$pred_knn_21,
  pred_lightgbm = train_data$pred_lightgbm,
  pred_lightgbm_21 = train_data$pred_lightgbm_21,
  pred_en = train_data$pred_en,
  pred_en_21 = train_data$pred_en_21,
  pred_en_a = train_data$pred_en_a,
  target = train_data$action_taken
)

train_meta_data = train_meta_data%>%
  group_by(target) %>%
  sample_frac(0.4) # memory limit of stacked model

# Train a meta-model
meta_model <- randomForest(target ~ ., data=train_meta_data)

test_data$pred_xgb <- predict(xgb_fit, new_data=test_data)
test_data$pred_xgb_21 <- predict(xgb_fit_21, new_data=test_data)
test_data$pred_rf <- predict(rf_fit, new_data=test_data)
test_data$pred_rf_21 <- predict(rf_fit_21, new_data=test_data)
test_data$pred_lightgbm <- predict(lightgbm_fit, new_data=test_data)
test_data$pred_lightgbm_21 <- predict(lightgbm_fit_21,
new_data=test_data)
test_data$pred_en <- predict(en_fit, new_data=test_data)
test_data$pred_en_21 <- predict(en_fit_21, new_data=test_data)
test_data$pred_en_a <- predict(en_fit_a, new_data=test_data)

# Create a new data frame to hold the test predictions
test_meta_data <- data.frame(
  pred_xgb = test_data$pred_xgb,
  pred_xgb_21 = test_data$pred_xgb_21,
  pred_rf = test_data$pred_rf,
  pred_rf_21 = test_data$pred_rf_21,
  pred_lightgbm = test_data$pred_lightgbm,
  pred_lightgbm_21 = test_data$pred_lightgbm_21,
  pred_en = test_data$pred_en,
  pred_en_21 = test_data$pred_en_21,
  pred_en_a = test_data$pred_en_a
)
```

```
# Generate final predictions using the meta-model
final_predictions <- predict(meta_model, newdata=test_meta_data)
result_df <- bind_cols(test_data %>% select(id), .pred_class =
final_predictions)
result_df <- result_df %>% rename(action_taken = .pred_class)
write.csv(result_df, "team_2_stack_3.csv", row.names = FALSE)
```

Appendix II: Team Member Contributions

Jun Yu Chen:

- Utilizes both Akaike Information Criterion (AIC) and Gradient Boosting Machine algorithms for variable selection.
- Executes sensitivity analysis to determine the optimal number of features, interpretation on top ranked features, and specifically focusing on the top 21 features.
- Wrote the Variable selection, baseline recipe and recipe 21 section

Anna Deng:

- Explores potential relationship between predictor variables and assists in transformation of variables
- Assists in conducting initial analysis and cleaning of data and writes the introduction and EDA section of the final report
- Creates data visualization to visually represent variable transformations and data analysis

Angelina Sun:

- Tuned the base models and the stacked models
- Project Management
- Wrote up description of models and the final model selection

Eric Xia:

- Assisted in transformation of variables, specifically ethnicity and race
- Explored various encoding methods
- Helped in training and tuning XGBoost, and logistic regression models

Nicole Xu:

- Conducted a comprehensive analysis of all categorical variables and created visualizations of the relationships between predictor variables and the response variable.
- Assisted in data cleaning and variables transformation.
- Wrote the EDA section.