Assignment 3 - Ensemble II Models using SAS Enterprise Miner on the Universal Bank Dataset

Tanushree Kumar | tanushree.kumar@outlook.com

DATA 640 - Fall 2024

Professor Steve Knode

University of Maryland Global Campus

Due: September 24, 2024

Introduction and Data Set Description

The objective of this analysis is to develop and evaluate different models using the Ensemble Node SAS Enterprise Miner to predict whether a customer will accept a personal loan based on the customer's demographic and financial data. Alongside these models, four simple models will also be developed to compare with the Ensemble Node models. The goal is to determine which model provides the best performance in classifying whether a customer is likely to take the loan, addressing imbalances in the dataset while exploring model variations.

The problem domain revolves around personal loan marketing in the banking sector. Banks like Universal Bank provide various financial services and often promote personal loans to existing customers. However, not all customers may be inclined to accept such offers. By analyzing customer demographics, financial information, and historical data, predictive models can be created to identify potential loan acceptors. Such models are critical for banks to target marketing efforts efficiently, increase loan acceptance rates, and reduce costs associated with customer acquisition.

The four simple models will consist of a decision tree, logistic regression, neural network, and naïve Bayes. Each model's parameters, like tree depth or learning rate for neural networks, can be carefully tuned through trial and error to enhance their predictive accuracy. The intuition behind ensemble models is to improve predictive performance by combining the strengths of multiple models. Instead of relying on a single model, ensemble methods use different algorithms to create multiple classifiers and then aggregate their predictions. In SAS Enterprise Miner, three ensemble node combination approaches are utilized: average, maximum, and voting. The average method works by combining the average of the posterior probabilities if the target is a class variable, and by averaging predicted values if the target is an interval scale

variable. The maximum method works by combining the maximum of the posterior probabilities if the target is a class variable, and by taking the maximum predicted value if the target is an interval-scaled variable. The voting approach can be split into voting average and voting proportion methods. The voting average method works by combining the average of the posterior probabilities from the models that are in the majority group. The voting proportion method, on the other hand, works by calculating the posterior probabilities as a ratio of the number of models in the majority group and the total number of models included (Sarma, 2013; Knode, 2016c). Key input parameters for these ensemble algorithms include the type of base models, e.g., decision tree, logistic regression, neural network, and the combination rule of either average, maximum, or voting. Proper parameter tuning, such as adjusting the decision threshold or boosting parameters, can significantly impact model performance.

The ensemble approach is relevant to this problem due to the heavily imbalanced nature of the target variable and the complexity of relationships between demographic and financial variables in predicting loan acceptance. Single models may struggle with such an imbalanced dataset, often biasing predictions toward the majority class. Ensemble methods, such as those using average or voting, can counter this by improving generalization and reducing model variance. By combining diverse models—such as the decision tree's ability to handle non-linear relationships, logistic regression's interpretability, and the neural network's ability to model complex patterns—ensemble methods offer robust and accurate predictions. These models capture the subtle interactions in customer behavior that might otherwise go unnoticed in a single-model approach, making ensemble techniques particularly suited for financial decision-making tasks like this.

The dataset is sourced from the class portal and consists of 5000 rows and 14 columns. Each row represents a customer, and each column represents a demographic or financial attribute, except for the target variable. The target variable is Personal Loan, which is binary with 1 = customer accepted the loan, 0 = customer did not accept the loan. This variable is highly imbalanced, with a large proportion of customers declining the offer, approximately 10% of customers accepted the loan, while 90% did not. The key features and their relevant details can be seen in the table below showing no missing values. The dataset also has no significant outliers which also results in minimal skew that is within an acceptable range.

Feature Name	**Feature Description**	**Missing Cases**	**Skewness**
ID	Customer ID	0	0
Age	Customer's age in completed years	0	-0.02934
Experience	Experience: Number of years of professional experience	0	-0.02632
Income	Annual income of the customer ($000)	0	0.841339
ZIPCode	Home address ZIP code	0	-
Family	Family size of the customer	0	-
CCAvg	Average spending on credit cards per month ($000)	0	1.598443
Education	Education level; 1 = Undergrad, 2 = Graduate, 3 = Advanced/Professional	0	-
Mortgage	Value of mortgage if applicable ($000)	0	2.104002
Personal Loan	Target variable; Did this customer accept the personal loan offered in the last campaign?	0	-
Securities Account	Does the customer have a securities account with the bank? (binary)	0	-
CD Account	Does the customer have a Certificate of Deposit (CD) account with the bank? (binary)	0	-
Online	Does the customer use internet banking facilities? (binary)	0	-

Credit Card	Does the customer have a credit card issued by UniversalBank? (binary)	0	-

Table 1. Table of key features and statistical values.

(Knode, 2024a)

Data Cleansing and Preparation

Based on the dataset, approximately 10% of customers accepted the loan, while 90% did not. This imbalance in the target variable will be addressed in the modeling process using techniques such as resampling, cost-sensitive learning, or cutoff adjustment. The resampling method uses oversampling or undersampling methods to balance the dataset. Cost-sensitive learning works by adjusting the cost function to penalize misclassification of the minority class. The cutoff adjustment works by altering the classification cutoff threshold from the default 0.5 value to better classify the minority class.

There were no missing values present in the dataset which meant no imputations were performed. However, if there were any missing values, the mean or the mode of the specific column would have been imputed to handle the missing data. There were also no significant outliers present, which were gathered from observing the minimum, maximum, and mean values. None of the variables were extremely skewed as they had a skewness value between the [-2,2] range, which is deemed acceptable. Thus, no transformations were required either. However, if some variables were skewed, the recommended approach would be to develop models without transforming any variables and then redevelop those models after transforming the highly skewed variables. The ‘ZIPCode’ column was removed as it wasn't deemed necessary to the models. Overall, minimal feature engineering was required since the dataset was in good condition with no errors.

A correlation matrix was developed and examined to see if there were any highly correlated features. The matrix can be seen in Figure 7 in Appendix A. It can be observed from the matrix that the 'Age' and 'Experience' variables are highly correlated with a correlation value of 0.994215. Besides that strong correlation, there is another correlation between 'Income' and 'CCAvg' with a value of 0.645984. A temporary sample model was used to assess if dropping any of the variables would give drastic results, however, no significant result changes were noticed so the variables remained in the dataset.

Development of Predictive Models

The data was partitioned into a training and validation split of 70% and 30% respectively. Starting with most of the data in the training set allows for better and more accurate models, but also increases the likelihood of overfitting. To check if any of the models are overfitting, the accuracy for the training and validation set will be observed since an overfitting model performs well on the training set but poorly on the validation set.

The first set of models consisted of just the four simple models developed as a baseline to compare with the ensemble models. These models had no adjustment or imbalance treatment which further helps to see if and how these ensemble models deal with an imbalanced target. This was also done to see if the adjustment makes a difference in the first place. These models were built with their default parameters in place with no external changes.

The second set of four models consisted of the 4 ensemble models of average, maximum, voting average, and voting proportion. These models were built by combining the four simple models. This second set also had no adjustment or imbalance treatment so it can serve as a baseline for comparison.

The third set of four models is similar to the second set with the exception of a cost function being applied as the imbalance treatment. The cost function was applied using the Decision Table and the Decision Matrix. The cost function essentially forces the model to find and correctly classify the rare cases (Knode, 2016b). However, a caveat of this technique is that it may result in more false positives but will also identify more true positives. Ideally, it would be good practice to experiment with different cost values and evaluate how they affect the model's performance. The goal is to find a balance that aligns with the business goals, maximizing the detection of the minority class without overly sacrificing accuracy in the majority class. From the previous analysis of Ensemble I models, it was found that a cost value of 2 proved to be the most effective value, hence all the models used a cost value of 2.

The final set of four models is similar to the second set with the exception of a cutoff threshold being applied as the imbalance treatment. The cutoff threshold was applied using the Cutoff Node in the Assess tab of SAS Eminer. This method works by changing the cutoff criterion from the default value of 0.5 to a specified user input value based on the cutoff graph. A cutoff node makes it easier for rare cases to be classified. Compared with the default 0.5 cutoff criterion, the specified input should be able to identify more True Positives, and as a result, more False Positives which is a drawback.

Results

The results table, Table 3, in Appendix A shows the summary and relevant statistics for all the models. As mentioned earlier in the data cleansing and preparation section, no imputing or transformations were performed. The table includes the model number; model name; training/validation set; imbalanced target adjustment; accuracy; misclassification rate; raw numbers for TP, FP, TN, and FN; sensitivity; precision; F1 score; and ROC index. The table also

includes a summary of the relevant statistics from the previously constructed SVM and Ensemble I models for an accurate comparison.

The sensitivity measurement measures how well the rare cases are being identified, in other words, how sensitive the model is to the minority case. The precision measurement helps ensure that the sensitivity is not achieved just by declaring cases to be positive. The ROC index measurement tells if the model is good enough to be used. This can be assessed by seeing the value: a value of .9 or better is an excellent model; a value of 0.8 is a pretty good model; and a value of 0.7 is subpar. The F1 value is a harmonic mean of sensitivity and precision values and can be used as a tiebreaker when models are close to other factors. Each of these measures helps to decide which, if any, of the models is best suited to be the champion model. The sensitivity, precision, and F1 values were all calculated with their respective formulas using the TP, FP, TN, and FN values.

As can be observed in the table, all the models had a ROC value of ranging from 0.96 to 1 with the majority of the models having a value of 0.99 or 1. Figures 8, 9, 10, and 11 in Appendix A show the ROC curves for all 16 models in order. Just by looking at the curves themselves, it can be said that all of the models performed well. By looking at the accuracy, misclassification rates, and sensitivity values, it can be observed that overfitting did not occur for any of the 16 models since the values for all three criteria did not have drastic differences between the training and validation sets.

Based on the metrics provided for all 16 models, the champion model is determined by evaluating performance on accuracy, precision, recall (sensitivity), and the ROC Index. These metrics help balance correct predictions (true positives and true negatives) with minimizing errors (false positives and false negatives).

Model 9, an ensemble model using the average method with a cost parameter of 2, was selected as the champion model because it provides the most balanced performance across all key evaluation metrics: accuracy, sensitivity (recall), precision, and ROC Index. With a validation accuracy of 98.47%, it ranks among the highest-performing models, ensuring that the model correctly predicts loan acceptance or rejection in the vast majority of cases. More importantly, it demonstrates a balanced sensitivity and precision—both close to 0.90—which means that the model is adept at not only identifying loan acceptances (true positives) but also minimizing false positives. This balance is crucial for a banking scenario where misclassifying a customer as likely to accept a loan could lead to unnecessary marketing or outreach costs. The ROC Index of 0.99 further confirms the model's excellent ability to distinguish between accepted and rejected loans.

While other models such as Model 6 (EN - Maximum) had higher sensitivity, they suffered from very low precision, leading to a significant number of false positives. Meanwhile, models like Model 7 (EN - Voting Avg.) had lower precision, making them less reliable for high-stakes financial decisions. Model 9 strikes the optimal balance between identifying true positives and keeping false positives low, making it the most suitable choice as the champion model.

The results of Model 9 show that it effectively predicts whether a customer will accept a loan based on demographic and financial data. With a validation sensitivity of 0.89, it identifies 89% of actual loan acceptances, which is important for targeting potential customers. Its precision of 0.95 in validation means that 95% of the customers it predicted to accept a loan were actual acceptances, ensuring minimal wastage of resources on customers unlikely to take up the

offer. This makes the model particularly useful in a marketing or customer acquisition strategy where the bank needs to allocate resources efficiently.

From the perspective of misclassification, Model 9 has a very low misclassification rate of 0.01531 on the validation set, further confirming that it performs well in separating the two classes. The ROC Index of 0.99 reinforces the model's high discriminatory power between customers who will accept a loan and those who will not. These results align well with the bank's objective of identifying potential loan customers while minimizing outreach to those who are not interested.

In this analysis, the cost parameter adjustment was the optimal imbalance correction method for building the models. While cutoff-based models (such as Model 13) performed well, they often led to either a lower precision or a less balanced sensitivity and precision. For instance, Model 13, which used a cutoff of 0.265, had a lower validation accuracy (94.54%) and precision (0.65) compared to Model 9, indicating that adjusting the cutoff alone was not as effective in balancing the model's performance.

Cost-sensitive learning, as applied in Model 9, effectively accounted for the imbalanced nature of the target variable by increasing the penalty for misclassifying positive examples (loan acceptances). This approach worked better because it allowed the model to focus more on minimizing false negatives (loan acceptances incorrectly classified as rejections), which was crucial for the bank's objective. Therefore, using the cost parameter (C = 2) to handle the imbalance proved to be the more effective method in this analysis.

The Ensemble II champion model (Model 9, EN - Average, C = 2) offers a strong balance between accuracy, sensitivity, and precision, with a validation accuracy of 98.47%, sensitivity of 0.89, and precision of 0.95. This balance makes it highly effective for identifying loan

acceptances while keeping false positives low. When compared to the Ensemble I champion model (Model 4, Random Forest with C = 2), the performance is quite close, but Random Forest edges out slightly in accuracy (98.47% vs. 98.20%) and precision, making it more reliable for minimizing false positives. However, Random Forest has a lower sensitivity (0.85) compared to Ensemble II, indicating that it might miss more actual loan acceptances. This makes Ensemble II a better choice for applications where correctly identifying true positives is critical, while Random Forest might be preferred when precision (fewer false positives) is prioritized.

On the other hand, SVM 7 (Polynomial 5 kernel) outperforms both ensemble models in training with a perfect accuracy of 100%, but this does not generalize as well to the validation set, where it achieves 97.00% accuracy. Its validation sensitivity of 0.88 and precision of 0.88 suggest that it is slightly less balanced compared to Model 9 in handling both true positives and false positives. While SVM 7 performs admirably and may work better in very specific high-dimensional scenarios, its tendency to overfit during training and lower precision in validation makes Model 9 the more reliable model overall for this problem. Ensemble II's model is better suited for generalization across the dataset, offering a more practical balance of key metrics compared to both the Random Forest and SVM models.

The handling of the imbalanced target variable played a crucial role in differentiating the performance of the various ensemble and SVM models. In the context of predicting loan acceptance, where there is a disproportionate distribution of positive and negative outcomes, balancing accuracy with the ability to detect true positives (loan acceptances) was critical.

For the Ensemble II models, the application of cost-sensitive learning (through adjusting the cost parameter) and the use of different cutoff thresholds had a noticeable impact. Model 9 (EN - Average, C = 2), which applied a cost of 2, emerged as the best performer. The cost

function effectively balanced the trade-off between false positives and false negatives, as seen in its high precision of 0.95 and sensitivity of 0.89. This indicated that the model performed well in identifying loan acceptances without overestimating false positives. On the other hand, Model 13 (EN - Average, Cutoff = 0.265), which used a cutoff adjustment, showed a significant drop in both accuracy and precision compared to Model 9. While adjusting the cutoff improved sensitivity (0.94), it reduced precision (0.65), highlighting the fact that cutoff adjustments alone may not be as effective as cost adjustments when managing imbalanced datasets.

When comparing the Ensemble II models with the previously created Ensemble I models, specifically the Random Forest (Model 4, C = 2), both models showed strong performance. However, Random Forest had slightly lower sensitivity (0.85) compared to Ensemble II Model 9 (0.89), which means that it missed more true positives (loan acceptances). This gap indicates that the cost adjustment in Ensemble II Model 9 helped it capture more true positives, a direct result of handling the imbalanced target better. The higher sensitivity suggests that cost-sensitive learning in the Ensemble II models allowed them to deal with imbalanced data more effectively.

In contrast, the SVM models, particularly SVM 7 (Polynomial 5), showed a different outcome. Despite achieving 100% accuracy in training, its validation performance (97.00% accuracy, 0.88 sensitivity, 0.88 precision) was lower than the Ensemble models. This suggests that SVM 7 might have overfitted the training data, and it struggled to maintain the same level of precision and sensitivity on the imbalanced validation set. SVM's performance, especially in cases with imbalanced targets, can be highly sensitive to the kernel type and its ability to generalize from high-dimensional data. In this case, polynomial kernels performed well but were less stable than ensemble methods in handling the imbalance.

Adjusting for the imbalanced target variable made a significant difference in the results, particularly for the ensemble models. The application of cost-sensitive learning in the Ensemble II models allowed them to outperform their counterparts by finding a better balance between sensitivity and precision. Cost adjustment was shown to be more effective than cutoff adjustments, as the former led to better overall model performance without sacrificing precision. In comparison, the Ensemble I models also handled the imbalance well but showed a slight trade-off in sensitivity for precision. The SVM models, while strong in some cases, were more prone to overfitting and did not generalize as well to the imbalanced data. Overall, cost-sensitive learning emerged as the optimal approach for handling imbalanced targets in this context.

Conclusions

The objective of this analysis was to predict whether a customer would accept a personal loan based on demographic and financial data using ensemble models. The key finding from this study is that ensemble methods, particularly the average ensemble with a cost function adjustment (Model 9), provided the best overall performance across various metrics, including accuracy, sensitivity, and precision. This model successfully balanced identifying true positives (customers likely to accept loans) while minimizing false positives. Additionally, cost-sensitive learning emerged as the most effective approach to handling the imbalanced nature of the dataset, outperforming cutoff adjustments and yielding better generalization for real-world applications.

One limitation of this analysis is the imbalanced nature of the dataset, where a large majority of customers did not accept the loan, which posed a challenge for model training. While cost-sensitive learning mitigated this issue, the models may still struggle in cases with extremely rare events or in datasets with more variability in customer behavior. Another limitation is the

reliance on the default parameter settings for certain models, such as decision trees and neural networks, which might have benefited from more extensive hyperparameter tuning.

Future improvements could focus on more advanced methods for handling data imbalance or experimenting with other cost-sensitive algorithms to further enhance model performance. Additionally, incorporating more feature engineering, such as using derived variables or interaction terms, could provide better insights into customer behavior. Moreover, hyperparameter tuning for more complex models like neural networks and SVMs could result in more refined and robust predictions.

In terms of model comparison, the ensemble models, particularly the average method with cost adjustment, outperformed simple models in terms of accuracy. While SVM models showed strong training performance, they tended to overfit, making them less reliable than ensemble methods for validation sets. Models with cost adjustment, especially Model 9, achieved a better balance in sensitivity and precision, making them adept at identifying true positives. In contrast, models like SVM performed well but were more prone to overfitting and lower sensitivity in validation. Ensemble methods are inherently more complex than simple models like logistic regression or decision trees. However, they provide the benefit of combining the strengths of multiple models, improving generalization. Simple models like decision trees and logistic regression are easier to implement and interpret but tend to be less powerful in handling complex, imbalanced datasets compared to ensemble methods. Ensemble methods require more computational resources and careful parameter tuning but deliver superior performance in complex cases. Ensemble methods, especially with cost adjustment, offer the best balance of performance, while future work could focus on addressing data imbalance more robustly and further optimizing model parameters.

References

Knode, S. (2016a). Adjusting for skewed target population [Vimeo]. In *Vimeo*. https://vimeo.com/186471846

Knode, S. (2016b). Cost function to adjust for skewed target population [Vimeo]. In *Vimeo*. https://vimeo.com/189827241

Knode, S. (2016c). ensemble models - ensemble node [Vimeo]. In *Vimeo*. Vimeo. https://vimeo.com/190769939

Knode, S. (2016d). ensemble models walkthrough - heterogeneous models [Vimeo]. In *Vimeo*. https://vimeo.com/191211534

Knode, S. (2024a). *universal bank description*. University of Maryland Global Campus. https://learn.umgc.edu/d2l/le/content/1226105/viewContent/33213198/View. Dataset description.

Knode, S. (2024b). *UniversalBank data*. University of Maryland Global Campus. https://learn.umgc.edu/d2l/le/content/1226105/viewContent/33213199/View. CSV file for the dataset.

Sarma, K. (2013). Chapter 7: Comparison and Combination of Different Models. In Predictive Modeling with SAS Enterprise Miner. SAS.

SAS® Enterprise Miner™ 15.1: Reference Help. (2020). In *documentation.sas.com*. SAS Help Center. https://documentation.sas.com/api/docsets/emref/15.1/content/emref.pdf?locale=en#nameddest=emrefwhatsnew143. Contains information and references to the following: Control Point Node; Decision Tree Node; Ensemble Node; HP Bayesian Network Classifier Node; HP Neural Node; and HP Regression Node.

Appendix A

All figures and visualizations mentioned in the report can be seen below.

Figure 1

Figure of the dataset.

Sample Table

ID	Age	Experience	Income	Family	CCAvg	Education	Mortgage	Online	CreditCard	Personal Loan	Securities Account	CD Account
1	25	1	49	4	1.6	1	0	0	0	0	1	0
2	45	19	34	3	1.5	1	0	0	0	0	1	0
3	39	15	11	1	1	1	0	0	0	0	0	0
4	35	9	100	1	2.7	2	0	0	0	0	0	0
5	35	8	45	4	1	2	0	0	1	0	0	0
6	37	13	29	4	0.4	2	155	1	0	0	0	0
7	53	27	72	2	1.5	2	0	1	0	0	0	0
8	50	24	22	1	0.3	3	0	0	1	0	0	0
9	35	10	81	3	0.6	2	104	1	0	0	0	0
10	34	9	180	1	8.9	3	0	0	0	1	0	0

Figure 2

Figure of the dataset showing no missing values and summary statistics with no missing and high skew values.

Name	ype	Number of Levels	Percent Missing	Minimum	Maximum	Mean	Standard Deviation	Skewness	Kurtosis
Age		.	0	23	67	45.3384	11.46317	-0.02934	-1.15307
CCAvg		.	0	0	10	1.937938	1.747659	1.598443	2.646706
CD_Account		2	0	.	.	.	.	.	.
CreditCard		2	0	.	.	.	.	.	.
Education		3	0	.	.	.	.	.	.
Experience		.	0	-3	43	20.1046	11.46795	-0.02632	-1.12152
Family		4	0	.	.	.	.	.	.
ID		.	0	1	5000	.	.	.	.
Income		.	0	8	224	73.7742	46.03373	0.841339	-0.04424
Mortgage		.	0	0	635	56.4988	101.7138	2.104002	4.756797
Online		2	0	.	.	.	.	.	.
Personal_Loan		2	0	.	.	.	.	.	.
Securities_Accou		2	0	.	.	.	.	.	.

Figure 3

Figure of the graph of the Chi-square values.

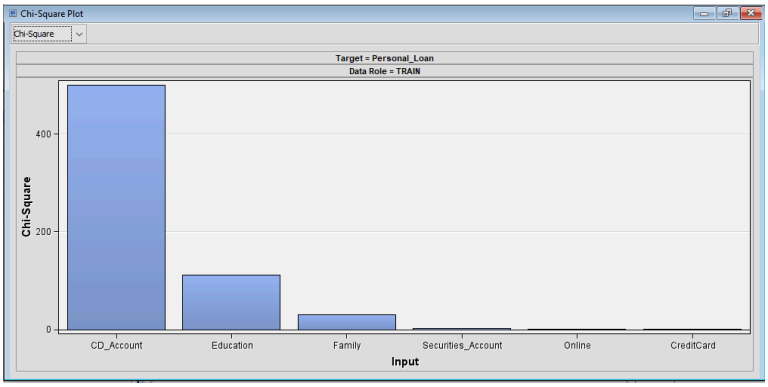


Figure 4

Figure of the graph of the variable worth.

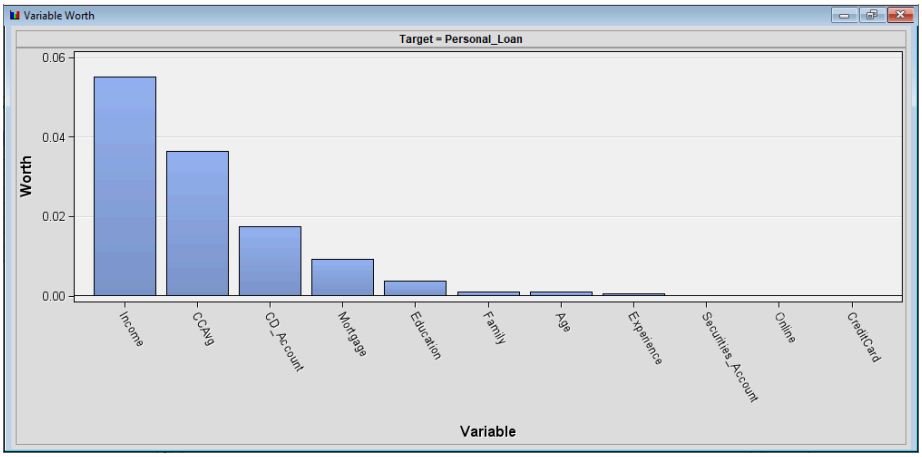


Figure 5

Figure of the cost matrix used for the cost function.

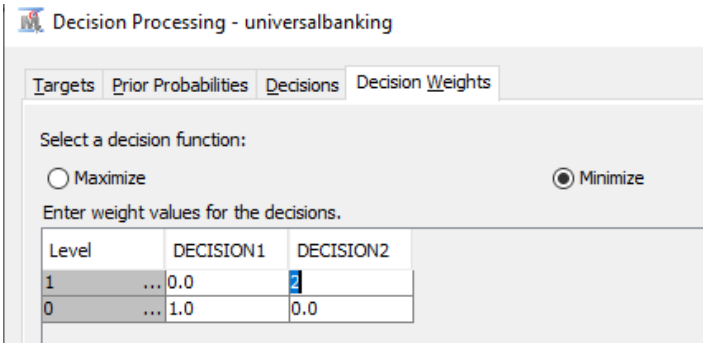
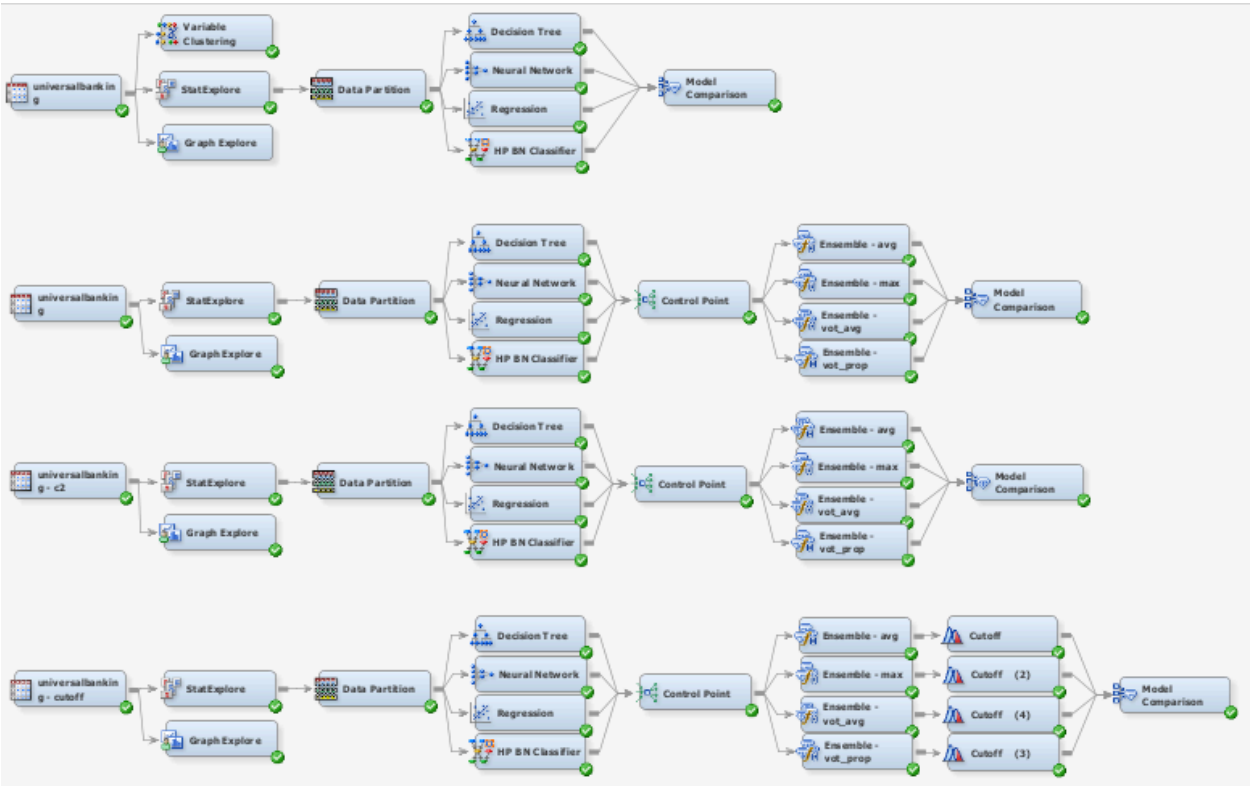
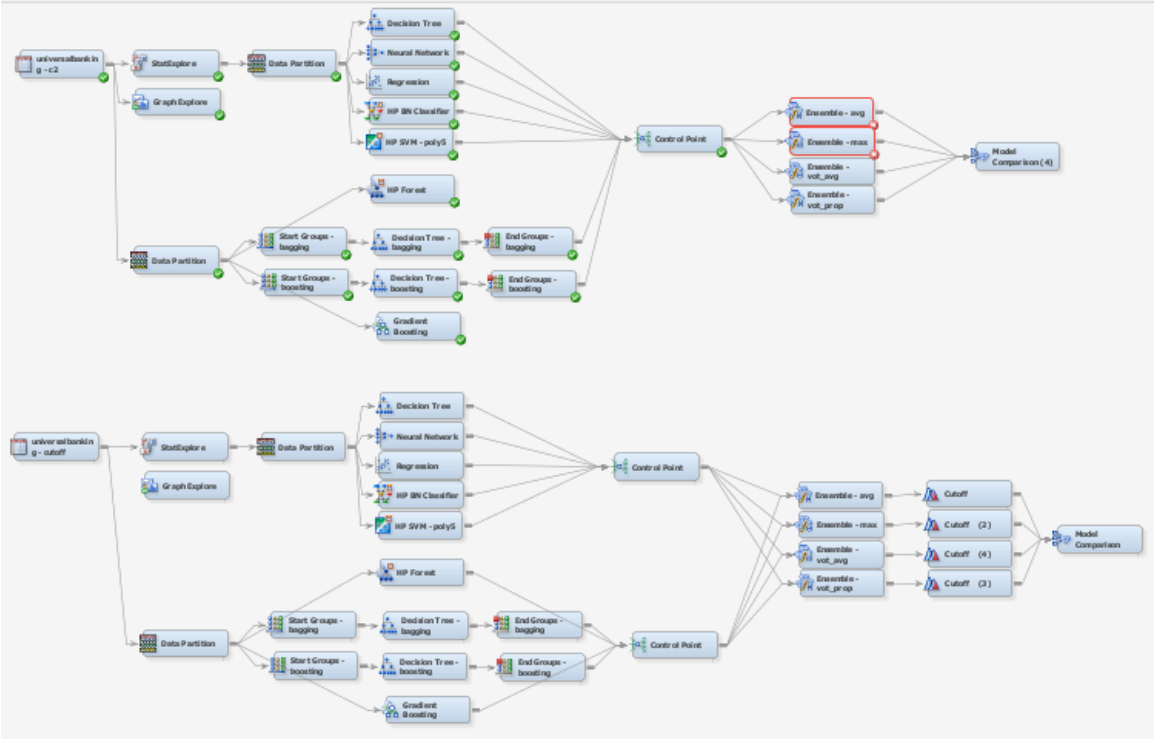


Figure 6

Figures of the Ensemble Models II diagram.





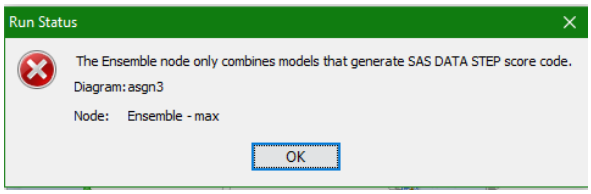


Figure 7

Figure of the correlation matrix.

Figure 8

Figure of the ROC curve for models 1-4.

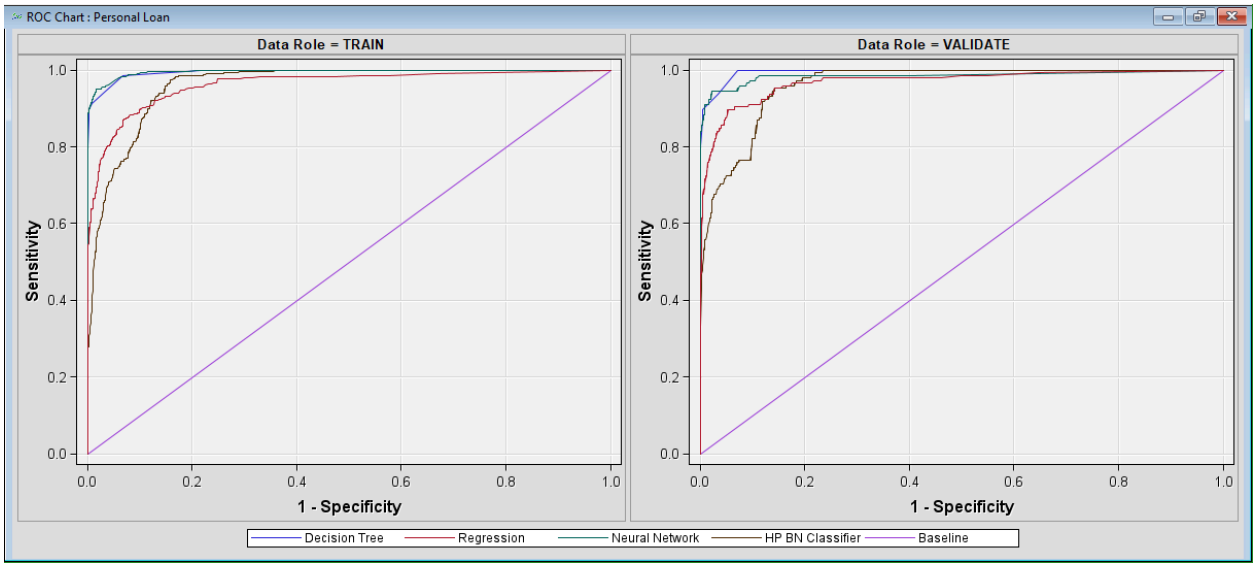


Figure 9

Figure of the ROC curve for models 5-8.

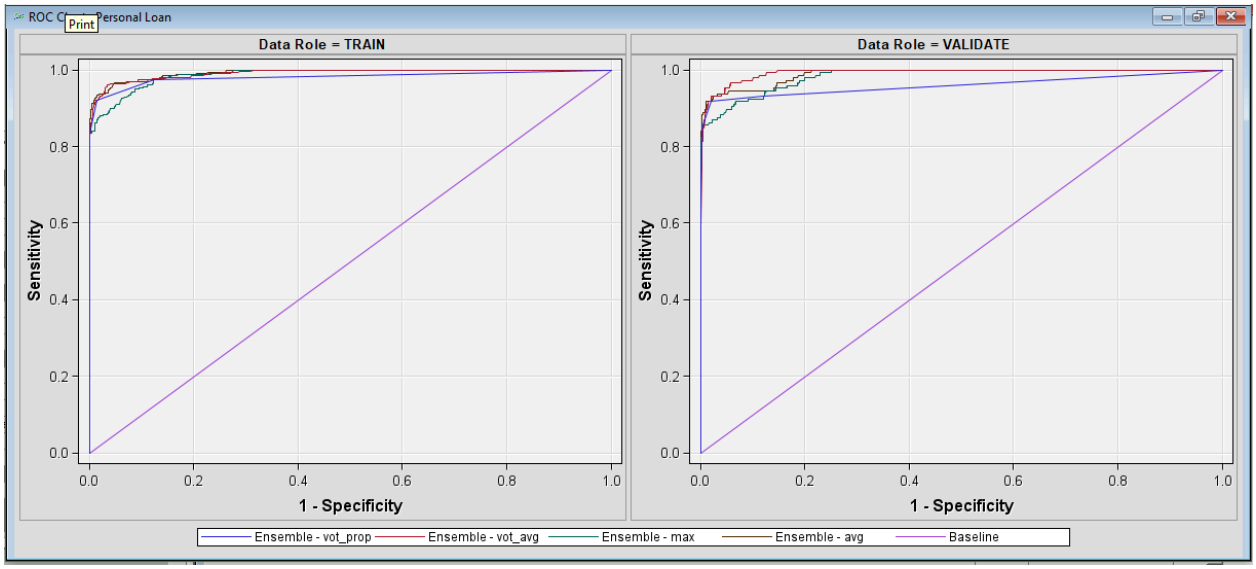


Figure 10

Figure of the ROC curve for models 9-12.

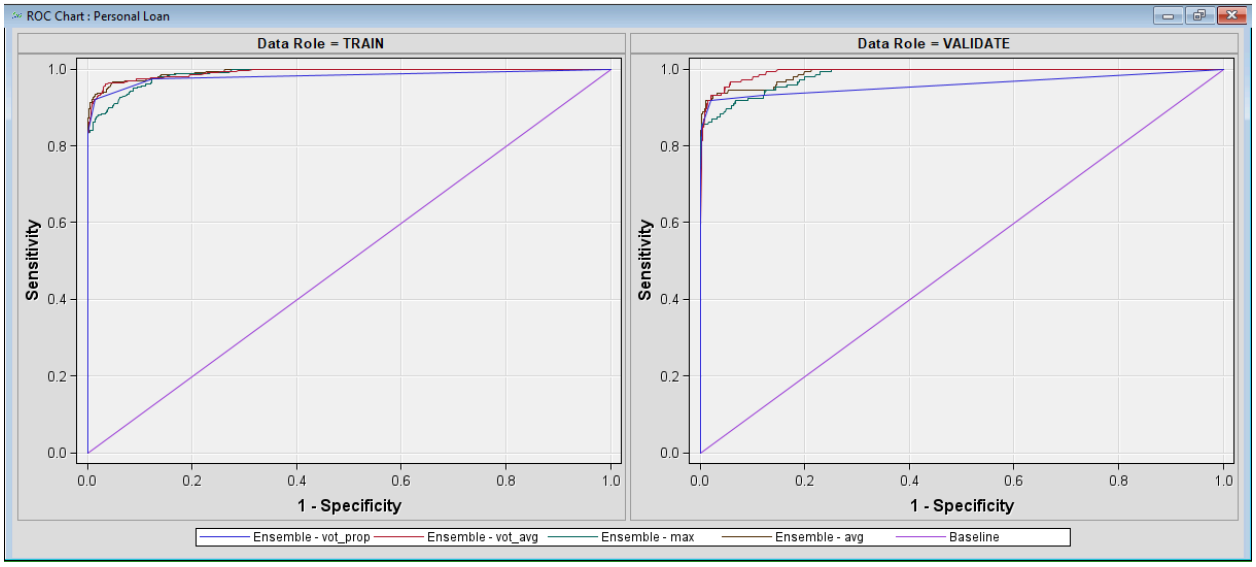


Figure 11

Figure of the ROC curve for models 13-16.

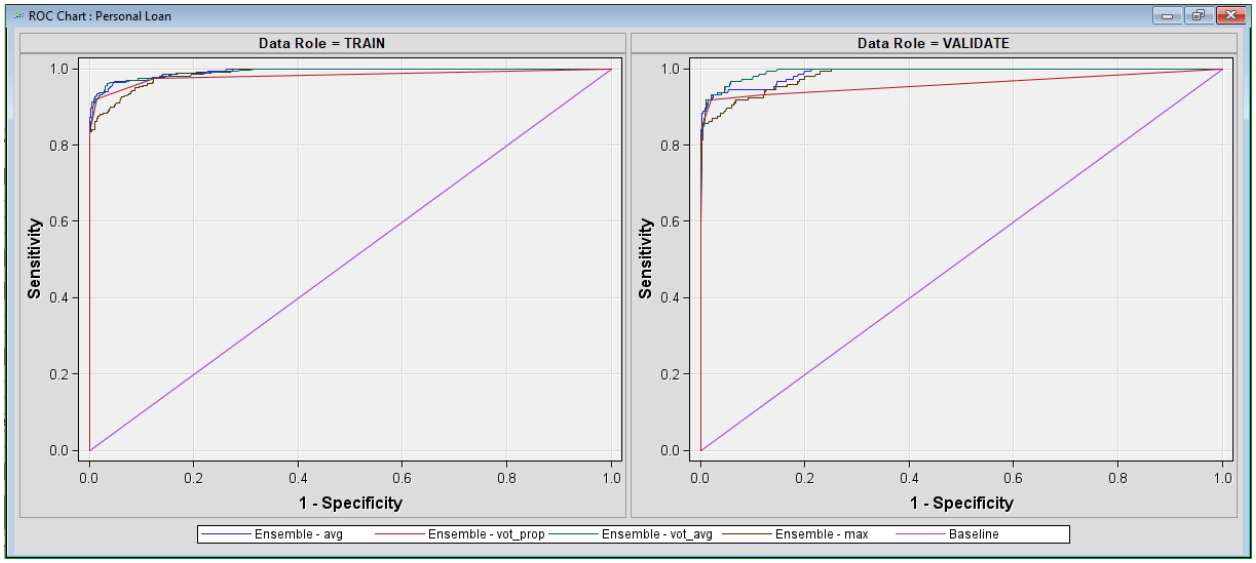


Figure 12

Figures of the Cutoff curves for models 13-16. Top left: Ensemble Average; top right: Ensemble Maximum; bottom left: Voting Average; and bottom right: Voting Proportion.

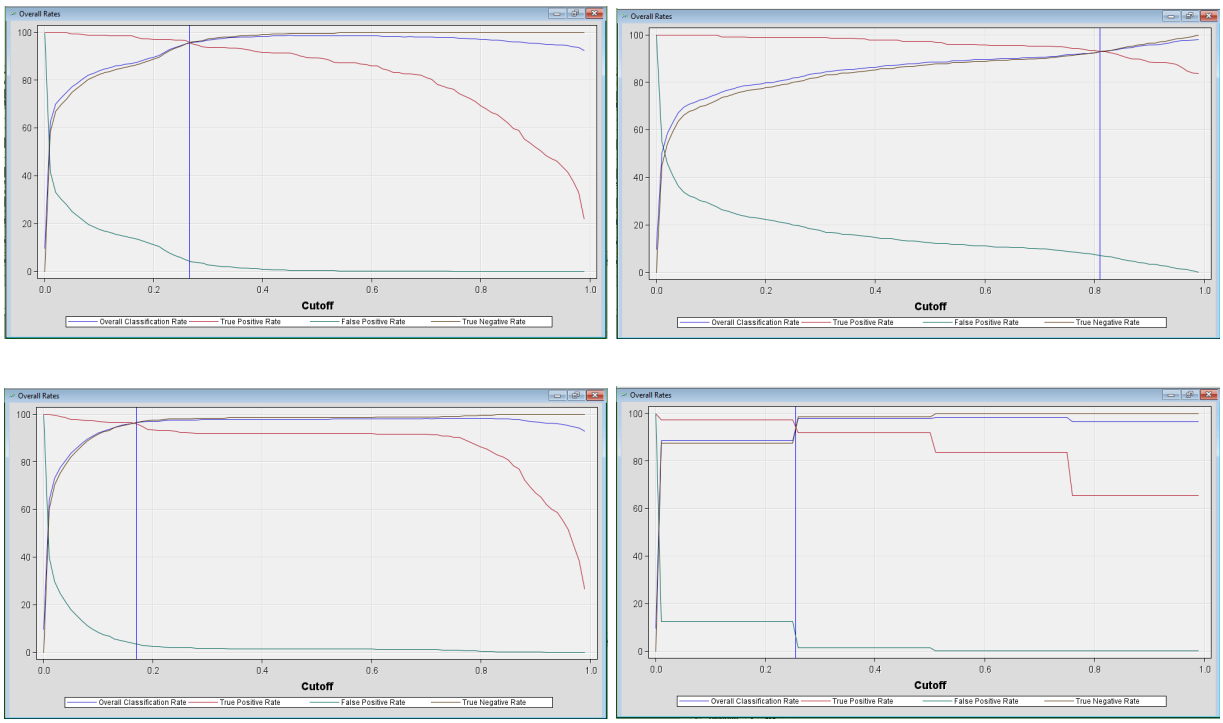


Table 2

Table of the model summary outlining key details and differences for Ensemble II, Ensemble I, and SVM models.

Model No.	Model Type	Adjustment	Imbalance Method	Parameters	Differences/Comments
1	SI - Decision Tree	No	N/A	No imbalance method. Max Depth: 6 Max Branch: 2; Min Category Size: 5 Leaf Size: 5; No. of rules: 5	4 different simple models for baseline comparison with no adjusments.
2	SI - Neural Network	No	N/A	No imbalance method. Hidden Units: No Residuals: Yes Standard.: No	
3	SI - Log. Reg.	No	N/A	No imbalance method. Regression Type: Logistic Selection Model: Stepwise	
4	SI - Naive Bayes	No	N/A	No imbalance method. Network Model: Naive Bayes Sig. Level: 0.2 Number of bins 10	
5	EN - Average	No	N/A	No imbalance method. Combines all 4 simple models above with the same parameters. Posterior Prob: Average	4 different simple models combined with each of the four ensemble node approaches with no adjustment for imbalance.
6	EN - Maximum	No	N/A	No imbalance method. Combines all 4 simple models above with the same parameters. Posterior Prob: Maximum	
7	EN - Voting Avg.	No	N/A	No imbalance method. Combines all 4 simple models above with the same parameters. Posterior Prob: Voting Average	
8	EN - Voting Prop.	No	N/A	No imbalance method. Combines all 4 simple models above with the same parameters. Posterior Prob: Voting Proportion	
9	EN - Average	Yes	Cost Function	Imbalance method: C = 2 Combines all 4 simple models above with the same parameters. Posterior Prob: Average	4 different simple models combined with each of the four ensemble node approaches with a cost function adjustment (c = 2) for imbalance.
10	EN - Maximum	Yes	Cost Function	Imbalance method: C = 2 Combines all 4 simple models above with the same parameters. Posterior Prob: Maximum	
11	EN - Voting Avg.	Yes	Cost Function	Imbalance method: C = 2 Combines all 4 simple models above with the same parameters. Posterior Prob: Voting Average	
12	EN - Voting Prop.	Yes	Cost Function	Imbalance method: C = 2 No imbalance method. Combines all 4 simple models above with the same parameters. Posterior Prob: Voting Proportion	
13	EN - Average	Yes	Cutoff	Imbalance method: Custom Cutoff Value Combines all 4 simple models above with the same parameters. Posterior Prob: Average	4 different simple models combined with each of the four ensemble node approaches with a cutoff approach adjustment for imbalance. Cutoff value adjusted accordingly for each ensemble model.
14	EN - Maximum	Yes	Cutoff	Imbalance method: Custom Cutoff Value Combines all 4 simple models above with the same parameters. Posterior Prob: Maximum	
15	EN - Voting Avg.	Yes	Cutoff	Imbalance method: Custom Cutoff Value Combines all 4 simple models above with the same parameters. Posterior Prob: Voting Average	
16	EN - Voting Prop.	Yes	Cutoff	Imbalance method: Custom Cutoff Value Combines all 4 simple models above with the same parameters. Posterior Prob: Voting Proportion	

Model No.	Model Type	Adjustment	Imbalance Method	Parameters	Differences/Comments
1	Bagging	Yes	Cost Function	Imbalance method: C = 2 Default parameters.	4 ensemble I models with cost function of c = 2 applied.
2	Boosting	Yes	Cost Function	Imbalance method: C = 2 Default parameters.	
3	Random Forest	Yes	Cost Function	Imbalance method: C = 2 Default parameters. Max number of trees: 100; Max depth: 50	
4	Gradient Boosting	Yes	Cost Function	Imbalance method: C = 2 Default parameters. Number of iterations: 50	
4	Bagging	Yes	Cost Function	Imbalance method: C = 4 Default parameters.	4 ensemble I models with cost function of c = 4 applied. Goal to increase accuracy and performace with higher c value
6	Boosting	Yes	Cost Function	Imbalance method: C = 4 Default parameters.	
6	Random Forest	Yes	Cost Function	Imbalance method: C = 4 Default parameters. Max number of trees: 100; Max depth: 50	
8	Gradient Boosting	Yes	Cost Function	Imbalance method: C = 4 Default parameters. Number of iterations: 50	
9	Bagging	Yes	Cost Function	Imbalance method: C = 6 Default parameters.	4 ensemble I models with cost function of c = 6 applied. Goal to increase accuracy and performace with higher c value
10	Boosting	Yes	Cost Function	Imbalance method: C = 4 Default parameters.	
11	Random Forest	Yes	Cost Function	Imbalance method: C = 4 Default parameters. Max number of trees: 100; Max depth: 50	
12	Gradient Boosting	Yes	Cost Function	Imbalance method: C = 4 Default parameters. Number of iterations: 50	

Model No.	Model Type	Adjustment	Imbalance Method	Parameters	Differences/Comments
SVM 1	HP SVM (lin)	Yes	Cost Function	Imbalance method: C = 2 Maximum Iteration: 25; Optimization Method: Interior Point; Kernel Type: Linear; Kernel Value: N/A	Standard linear model with default parameters. No kernel value. used interior point.
SVM 2	HP SVM (poly2)	Yes	Cost Function	Imbalance method: C = 2 Maximum Iteration: 25; Optimization Method: Interior Point; Kernel Type: Polynomial; Kernel Value: 2	Similar polynomial models with the exception of a different kernel value which can increase the accuracy of the model. Used interior point. Used default parameters.
SVM 3	HP SVM (poly3)	Yes	Cost Function	Imbalance method: C = 2 Maximum Iteration: 25; Optimization Method: Interior Point; Kernel Type: Polynomial; Kernel Value: 3	
SVM 4	HP SVM (RBF)	Yes	Cost Function	Imbalance method: C = 2 Maximum Iteration: 25; Optimization Method: Active Set; Kernel Type: RBF; Kernel Value: 1	Standard RBF model with default parameters with 1 kernel value. Used active set. Used default parameters
SVM 5	HP SVM (sig)	Yes	Cost Function	Imbalance method: C = 2 Maximum Iteration: 25; Optimization Method: Active Set; Kernel Type: Sigmoid; Kernel Value: [-1,1]	Standard sigmoid model with default parameters with [-1,1] kernel value. Used active set. Used default parameters
SVM 6	HP SVM (poly4)	Yes	Cost Function	Imbalance method: C = 2 Maximum Iteration: 25; Optimization Method: Active Set; Kernel Type: Polynomial; Kernel Value: 4	Similar polynomial models with the exception of a different kernel value which can increase the accuracy of the model. Used active set. Used default parameters.
SVM 7	HP SVM (poly5)	Yes	Cost Function	Imbalance method: C = 2 Maximum Iteration: 25; Optimization Method: Active Set; Kernel Type: Polynomial; Kernel Value: 4	
SVM 8	HP SVM (RBF2)	Yes	Cost Function	Imbalance method: C = 2 Maximum Iteration: 25; Optimization Method: Active Set; Kernel Type: RBF; Kernel Value: 2	Similar RBF models with the exception of a different kernel value which can increase the accuracy of the model. Used active set. Used default parameters.
SVM 9	HP SVM (RBF3)	Yes	Cost Function	Imbalance method: C = 2 Maximum Iteration: 25; Optimization Method: Active Set; Kernel Type: RBF; Kernel Value: 3	

Table 3

Table of the model outputs and relevant statistics including the model number; model name; training/validation set; imbalanced target adjustment which is the cost function; accuracy; misclassification rate; raw numbers for TP, FP, TN, and FN; sensitivity; precision; F1 score; and ROC index. This table contains results from previously constructed SVM models, Ensemble I models, and the current Ensemble II models.

Model No.	Model Type	Adjustment	Train/Validate	Accuracy	Misclassification Rate	TP	FP	TN	FN	Sensitivity (recall)	Precision	F1 Score	ROC Index
1	SI - Decision Tree	N/A	Train	98.69	0.01315	300	11	3152	35	0.90	0.96	0.93	1.00
	SI - Decision Tree	N/A	Validate	98.47	0.01531	130	8	1349	15	0.90	0.94	0.92	1.00
2	SI - Neural Network	N/A	Train	98.71	0.01286	303	13	3150	32	0.90	0.96	0.93	1.00
	SI - Neural Network	N/A	Validate	98.14	0.01864	129	12	1345	16	0.89	0.91	0.90	0.99
3	SI - Log. Reg.	N/A	Train	95.71	0.04288	223	38	3125	112	0.67	0.85	0.75	0.96
	SI - Log. Reg.	N/A	Validate	96.27	0.03728	107	18	1339	38	0.74	0.86	0.79	0.97
4	SI - Naive Bayes	N/A	Train	88.19	0.11807	307	385	2778	28	0.92	0.44	0.60	0.96
	SI - Naive Bayes	N/A	Validate	88.28	0.11718	129	160	1197	16	0.89	0.45	0.59	0.96
5	EN - Average	N/A	Train	98.69	0.01315	299	10	3153	36	0.89	0.97	0.93	0.99
	EN - Average	N/A	Validate	98.47	0.01531	129	7	1350	16	0.89	0.95	0.92	0.99
6	EN - Maximum	N/A	Train	88.48	0.11521	326	394	2769	9	0.97	0.45	0.62	0.99
	EN - Maximum	N/A	Validate	88.35	0.11651	135	165	1192	10	0.93	0.45	0.61	0.98
7	EN - Voting Avg.	N/A	Train	97.94	0.02058	308	45	3118	27	0.92	0.87	0.90	0.99
	EN - Voting Avg.	N/A	Validate	97.34	0.02663	133	28	1329	12	0.92	0.83	0.87	0.99
8	EN - Voting Prop.	N/A	Train	97.94	0.02058	308	45	3118	27	0.92	0.87	0.90	0.98
	EN - Voting Prop.	N/A	Validate	97.34	0.02663	133	28	1329	12	0.92	0.83	0.87	0.96
9	EN - Average	C = 2	Train	98.69	0.01315	299	10	3153	36	0.89	0.97	0.93	0.99
	EN - Average	C = 2	Validate	98.47	0.01531	129	7	1350	16	0.89	0.95	0.92	0.99
10	EN - Maximum	C = 2	Train	88.48	0.11521	326	394	2769	9	0.97	0.45	0.62	0.99
	EN - Maximum	C = 2	Validate	88.35	0.11651	135	165	1192	10	0.93	0.45	0.61	0.98
11	EN - Voting Avg	C = 2	Train	97.94	0.02058	308	45	3118	27	0.92	0.87	0.90	0.99
	EN - Voting Avg	C = 2	Validate	97.34	0.02663	133	28	1329	12	0.92	0.83	0.87	0.99
12	EN - Voting Prop	C = 2	Train	97.94	0.02058	308	45	3118	27	0.92	0.87	0.90	0.98
	EN - Voting Prop	C = 2	Validate	97.34	0.02663	133	28	1329	12	0.92	0.83	0.87	0.96
13	EN - Average	Cutoff = 0.265	Train	95.37	0.0463	323	150	3013	12	0.96	0.68	0.80	0.99
	EN - Average	Cutoff = 0.265	Validate	94.54	0.0546	137	74	1283	8	0.94	0.65	0.77	0.99
14	EN - Maximum	Cutoff = 0.81	Train	92.94	0.0706	311	223	2940	24	0.93	0.58	0.72	0.99
	EN - Maximum	Cutoff = 0.81	Validate	92.94	0.0706	133	94	1263	12	0.92	0.59	0.72	0.98
15	EN - Voting Avg	Cutoff = 0.17	Train	96.43	0.0357	322	112	3051	13	0.96	0.74	0.84	0.99
	EN - Voting Avg	Cutoff = 0.17	Validate	95.27	0.0473	136	62	1295	9	0.94	0.69	0.79	0.99
16	EN - Voting Prop	Cutoff = 0.255	Train	88.48	0.1152	326	394	2769	9	0.97	0.45	0.62	0.98
	EN - Voting Prop	Cutoff = 0.255	Validate	88.35	0.1165	135	165	1192	10	0.93	0.45	0.61	0.96

Model No.	Model Type	Train/Validate	Cost (C)	Accuracy	Misclassification Rate	TP	FP	TN	FN	Sensitivity (recall)	Precision	F1 Score	ROC Index
1	Bagging	Train	2	97.46	0.025443	246	0	3163	89	0.73	1.00	0.85	0.99
		Validate	2	97.60	0.023968	109	0	1357	36	0.75	1.00	0.86	0.99
2	Boosting	Train	2	97.63	0.023728	252	0	3163	83	0.75	1.00	0.86	1.00
		Validate	2	97.67	0.023302	110	0	1357	35	0.76	1.00	0.86	1.00
3	Random Forest	Train	2	99.74	0.002573	326	0	3163	9	0.97	1.00	0.99	1.00
		Validate	2	98.47	0.015313	126	4	1353	19	0.87	0.97	0.92	1.00
4	Gradient Boosting	Train	2	98.23	0.017724	278	5	3158	57	0.83	0.98	0.90	0.99
		Validate	2	98.20	0.017976	123	5	1352	22	0.85	0.96	0.90	0.99

Model No.	Model Type	Kernel Type	Kernel Value	Train/Validate	Cost (C)	Accuracy	Misclassification Rate	Sensitivity	ROC Index
SVM 1	HP SVM (lin)	Linear	-	Train	2	95.68	0.0432	0.6090	0.96
				Validate		96.47	0.0353	0.6828	0.97
SVM 2	HP SVM (poly2)	Polynomial	2	Train	2	98.77	0.0123	0.8985	0.99
				Validate		98.07	0.0193	0.8690	0.99
SVM 3	HP SVM (poly3)	Polynomial	3	Train	2	99.51	0.0049	0.9582	1.00
				Validate		97.60	0.0240	0.8690	0.98
SVM 4	HP SVM (RBF)	RBF	1	Train	2	97.74	0.0226	0.7701	0.99
				Validate		97.27	0.0273	0.7310	0.98
SVM 5	HP SVM (sig)	Sigmoid	[-1, 1]	Train	2	84.88	0.1512	0.2000	0.59
				Validate		83.29	0.1671	0.1172	0.54
SVM 6	HP SVM (poly4)	Polynomial	4	Train	2	99.86	0.0014	0.9881	1.00
				Validate		97.87	0.0213	0.9034	0.98
SVM 7	HP SVM (poly5)	Polynomial	5	Train	2	100.00	0.0000	1.0000	1.00
				Validate		97.00	0.0300	0.8759	0.98
SVM 8	HP SVM (RBF2)	RBF	2	Train	2	97.23	0.0277	0.7164	0.98
				Validate		97.27	0.0273	0.7241	0.98
SVM 9	HP SVM (RBF3)	RBF	3	Train	2	95.88	0.0412	0.5701	0.98
				Validate		96.27	0.0373	0.6207	0.98