Predictive Models in Business Analytics Project

December 2nd, 2022
Marco Bijvank
Group 01
OPMA 419

Daniel Jovanovic (ID: 30024206)
Rupin Sehgal (ID: 30088101)

Task 1

Data Mining Process:

The data mining process for task 1 of this project started with identifying which attributes in dataset were adequately correlated with our target variable "CARAVAN". This was done with the "Correlation Matrix", and only predictor attributes that had either a positive or negative correlation of at least 0.05 were chosen to be a further part of the model development process (**Appendix 1**). This was primarily done to ensure that our future classification models would only interact with the most useful attributes within the dataset in order to enhance our predictive capabilities while simultaneously reducing computational requirements. Furthermore, upon an initial exploration of the given dataset, we noticed that all of the attributes were registered as integers, however, many of them were in fact categorical. As a result, we utilized the "Numerical to Polynomial" operator to select the relevant predictor attributes and convert them to the proper data type. This was a necessary component of the data mining process because if we were to treat the categorical attributes as numerical variables, this would assume an order with regard to the records within each attribute, and would ultimately yield inaccurate results. Additionally, virtually every classification model that we built incorporated a threshold that was optimized using the relevant operator and then later applied to our validation set, which allowed us to avoid simply doing a majority vote for the prediction task. Due to the relatively large asymmetry with regard to the number of policy buyers and non-policy buyers in the dataset, we felt that it was necessary to incorporate stratified rather than shuffled sampling, and after some experimentation, we observed better results with the stratified sampling. Furthermore, observing this asymmetry allowed us to realize that accuracy and sensitivity would not be appropriate performance measures to use for this task. Sensitivity as a performance metric in this context is especially problematic as it would likely classify most records as being potential buyers even when they are not. Therefore, we chose AUC as our main criterion for performance. With the exception of the logistic regression model, we also utilized the "Cross Validation" operator in order to ensure a more extensive evaluation of each model that was created. Finally, normalization was also applied to a select number of our models, where applicable.

Decision Tree:

Best Parameter Values	
splitting criterion	Gain Ratio
maximal depth	10
minimal leaf size	1
minimal size for split	3
number of prepruning alternatives	0
cut-off threshold value	0.900

	Training Performance		Validation Performance		
confusion matrix		false	true		false	true
	Pred. false	3261	204	Pred. false	1350	33
	Pred. true	22	6	Pred. true	839	107
accuracy	93.53% (+/- 0.41%)			62.56%		
sensitivity	2.84% (+/- 3.25%)			76.43%		
specificity	99.33% (+/- 0.37%)			61.67%		
lift	335.01% (+/- 398.66%)			188.16%		
AUC	0.679 (+/- 0.057)			0.692		

Approach used to find the best decision tree model:

In order to find the best decision tree model, we utilized the "Optimize" operator to explore different parameters within the decision tree for the purpose of maximizing the AUC metric. Additionally, we also optimized for the threshold value to achieve the same goal, which required us to run the process again so that the new threshold could be applied to the validation data set once an optimal value for this metric was found. Our general approach to optimizing our decision tree was to run the model with a standard set of values for each parameter that was being optimized, review our results, and then make successive tweaks to the range of values being optimized until we were able to maximize the AUC metric. As was previously mentioned, we also experimented with stratified and shuffled sampling, cross validation and different splits of the data that were used for the training and validation sets. Ultimately, a gain ratio splitting criterion, maximal depth of 10, minimal leaf size of 1, minimal size for split of 3, number of prepruning alternative of 0, and a cut-off threshold value of 0.900 generated the most optimal value of AUC, which was 0.692 for the validation set.

Naïve Bayes:

Best Parameter Values	
cut-off threshold value	0.900

	Training Performance		Validation Performance		
confusion matrix		false	true		false	true
	Pred. false	3283	210	Pred. false	1719	64
	Pred. true	0	0	Pred. true	470	76
accuracy	93.99% (+/- 0.19%)			77.07%		
sensitivity	0.00% (+/- 0.00%)			54.29%		
specificity	100.00% (+/- 0.00%)			78.53%		
lift	Unknown			231.56%		
AUC	0.719 (+/- 0.065)			0.763		

Approach used to find the best Naïve Bayes model:

To find the best Naïve Bayes model, we only optimized for the most optimal threshold value since this particular model did not have many parameters that we could explore. However, we still experimented with different sampling and evaluation methodologies to arrive at the highest possible AUC. Ultimately, an AUC value of 0.763 was achieved for the validation set, which is much higher compared to the value generated by our decision tree model.

SVM:

Best Parameter Values	
Kernel type	Anova
Kernel degree	2
Kernel gamma	2
C	0
cut-off threshold value	0.900

	Training Performance	Validation Performance
confusion matrix		

	false	true
Pred. false	3283	210
Pred. true	0	0

	false	true
Pred. false	0	0
Pred. true	2189	140

accuracy	93.99% (+/- 0.19%)	6.01%
sensitivity	0.00% (+/- 0.00%)	100.00%
specificity	100.00% (+/- 0.00%)	0.00%
lift	Unknown	100.00%
AUC	0.649 (+/- 0.107)	0.648

Approach used to find the best SVM model:

After optimizing for the kernel type, C value and threshold, we were arrived at a SVM model which produced a validation set AUC value of 0.648. This suggests that our SVM model underperformed by a relatively large margin compared to both the decision tree and Naïve Bayes models. For the kernel type parameter, we included dot, radial, polynomial and anova in our optimize process since they appeared to be the most relevant to our underlying dataset. Furthermore, we started by optimizing several values for the C parameter, and then adjusted the min, max and steps options within the optimize parameter to enhance the efficiency of our exploratory analysis and lessen computational requirements.

k-NN:

Best Parameter Values	
numerical measure	Camberra Distance
k	70
cut-off threshold value	0.900

	Training Performance		Validation Performance		
confusion matrix					
	false	true		false	true
Pred. false	3283	210	Pred. false	1865	72
Pred. true	0	0	Pred. true	324	68
accuracy	93.99% (+/- 0.19%)		83.00%		
sensitivity	0.00% (+/- 0.00%)		48.57%		
specificity	100.00% (+/- 0.00%)		85.20%		
lift	Unknown		288.58%		
AUC	0.733 (+/- 0.063)		0.733		

Approach used to find the best k-NN model:

The best k-NN model that we were able to create resulted in an AUC of 0.733, which outperformed the prior decision tree and SVM models, however, the Naïve Bayes remained the best model up to this point. For the k-NN model, the Euclidean, Camberra and Manhattan distances were explored in our optimization process, along with correlation and cosine similarities. Near the end of our exploration, k-values from one to one hundred were optimized as well in steps of ten, however, the steps value was sent much higher initially. The cut-off threshold was also optimized to arrive at the best model, as before.

Neural Network:

Best Parameter Values	
training cycles	50
learning rate	0.067
momentum	0.800
cut-off threshold value	0.900

	Training Performance	Validation Performance
confusion matrix		

	false	true
Pred. false	3282	210
Pred. true	1	0

	false	true
Pred. false	1840	65
Pred. true	349	75

accuracy	93.96% (+/- 0.24%)	82.22%
sensitivity	0.00% (+/- 0.00%)	53.57%
specificity	99.97% (+/- 0.10%)	84.06%
lift	0.00%	294.26%
AUC	0.753 (+/- 0.056)	0.762

Approach used to find the best neural network model:
The optimal neural network model generated an AUC value of 0.762 for the validation set, which out performed all of the previously mentioned models except for Naïve Bayes. For this model, we optimized the training cycles, learning rate and momentum parameters. Once again, the min, max and steps were gradually adjusted either upward or downward until we failed to notice any significant changes with regard to the AUC metric in the training set.

Logistic Regression:

Best Parameter Values	
solver	L_BFGS
cut-off threshold value	0.900

	Training Performance	Validation Performance
confusion matrix		

	false	true
Pred. false	1095	70
Pred. true	0	0

	false	true
Pred. false	905	31
Pred. true	189	39

accuracy	93.99%	81.10%
sensitivity	0.00%	55.71%
specificity	100.00%	82.72%
lift	Unknown	284.44%
AUC	0.752	0.802

Approach used to find the best logistic regression model:

The logistic regression model generated the highest AUC metric for the validation set compared to all of the other models we built for task 1 of this project, and therefore logistic regression represents our final submission in regard to the competition aspect of this project since it was able to correctly identify the highest number of customers who will purchase the caravan policy. Although we were only able to optimize for the solver parameter, we decided to experiment with forward selection along with backward elimination in order to arrive at the most optimal logistic regression model. Subsequently, including the forward selection operator resulted in the highest AUC value of 0.802 for the validation set.

Ensemble:

	Validation Performance		
confusion matrix		false	true
	Pred. false	934	38
	Pred. true	161	32
accuracy	82.92%		
sensitivity	45.71%		
specificity	85.30%		
lift	275.94%		
AUC	0.750		

Approach used to find the best ensemble model:

Our best ensemble model consisted of logistic regression with bagging. For all applicable models we used cross validation to more robustly validate our data and we used AUC to compare the performance of the various models as mentioned above. Another model we explored was with the vote operator. Within vote, we used logistic, SVM, KNN, and neural net. Unfortunately, the model only generated an AUC value of 0.506 and was quickly eliminated from our selection of the best model. Additionally, we also tested bagging with neural net which resulted in the second highest AUC of 0.737. We also tried KNN with bagging which resulted in the third highest AUC of 0.721. Alongside bagging, we re-ran all the aforementioned models using adaboost to compare results. Our logistic regression model resulted in an AUC of 0.724, which is significantly lower than the bagging logistic regression model. Furthermore, the neural net adaboost model performed significantly worse with adaboost, only acquiring an AUC of 0.500. Lastly, we did adaboost with k-NN and the model resulted in an AUC of 0.500. Other than bagging and boosting, we also experimented with implementing a random forest operator. Using the optimize parameters operator we were able to optimize the parameters for the random forest. Resulting in an optimized AUC of 0.748; this could be a valid alternative to the logistic regression model due to its similar AUC performance. In the end after testing over eight ensemble models, we choose the bagging logistic regression model to be our final ensemble model as it had the highest AUC of 0.750.

Task 2

Data Mining Process:

Similar to the first task, the data mining process for task 2 also began by carefully selecting predictor attributes that displayed a correlation of at least +/- 0.05 with the target variable, for the same aforementioned reasons. Likewise, categorical attributes were once again converted to the correct datatype using the appropriate operator. Furthermore, the models that were built for the classification problem in this task incorporated thresholds, stratified sampling, along with cross validation and normalization where appropriate. However, the most notable change in our approach to task 2 was the inclusion of a cost matrix. Since there is a cost of $1 associated with selecting any record, we determined that the net profit gained by selecting a policy buyer is $14 ($15 – $1), which was entered as a negative in the cost matrix since this is positive cash in-flow. Subsequently, if a non-policy holder is selected then there is only a cost associated with this action, and therefore this was entered as a positive in the cost matrix (**Appendix 2**). Furthermore, it was stated that for this specific task we were allowed to determine the number of selected records to be submitted. Therefore, after we sorted the confidence/propensity scores from largest to smallest within the excel file, we selected all of the records that had a propensity value of at least 0.100, because we determined that anything lower then this would be almost entire inaccurate.

Decision Tree:

Best Parameter Values	
splitting criterion	Gain Ratio
maximal depth	10
minimal leaf size	1
minimal size for split	3
number of prepruning alternatives	10
cut-off threshold value	0.900

	Training Performance	Validation Performance
confusion matrix	<table><tr><td></td><td>false</td><td>true</td></tr><tr><td>Pred. false</td><td>2329</td><td>86</td></tr><tr><td>Pred. true</td><td>954</td><td>124</td></tr></table>	<table><tr><td></td><td>false</td><td>true</td></tr><tr><td>Pred. false</td><td>1350</td><td>33</td></tr><tr><td>Pred. true</td><td>839</td><td>107</td></tr></table>
accuracy	70.22% (+/- 11.04%)	62.56%
sensitivity	59.20% (+/- 26.29%)	76.43%
specificity	70.94% (+/- 13.29%)	61.67%
lift	195.12% (+/- 34.90%)	188.16%
AUC	0.679 (+/- 0.057%)	0.692
misclassification cost	-0.224 (+/- 0.116)	-0.283

Approach used to find the best decision tree model:

In order to arrive at the best decision tree model, we used a similar approach as in task 1, however we now optimized for the lowest possible misclassification cost, instead of attempting to maximize AUC. As a result, our best decision tree model for task 2 generated a misclassification cost of -0.283 for the validation set after the splitting criterion, maximal depth, minimal leaf size, minimal size for split, number of prepruning alternatives and cut-off threshold parameters were explored and optimized. Likewise, we also experimented with different sampling methods, cross validation, and different data splits, just as before.

Naïve Bayes:

Best Parameter Values	
cut-off threshold value	0.900

	Training Performance		Validation Performance		
confusion matrix		false	true		false	true
	Pred. false	2785	113	Pred. false	1719	64
	Pred. true	498	97	Pred. true	470	76
accuracy	82.51% (+/- 1.84%)			77.07%		
sensitivity	46.17% (+/- 13.77%)			54.29%		
specificity	84.83% (+/- 1.84%)			78.53%		
lift	270.08% (+/- 66.46%)			231.56%		
AUC	0.719 (+/- 0.065%)			0.763		
misclassification cost	-0.246 (+/- 0.119)			-0.255		

Approach used to find the best Naïve Bayes model:

Our best model for Naïve Bayes produced a misclassification cost of -0.255 for the validation performance, which was lower than the misclassification cost generated by the decision tree model. Only the cut-off threshold was able to be optimized in this case, because the Naïve Bayes model does not have many adjustable parameters.

SVM:

Best Parameter Values	
Kernel type	Anova
Kernel degree	2
Kernel gamma	2
C	0.150
cut-off threshold value	0.800

	Training Performance	Validation Performance
confusion matrix	false / true Pred. false: 2750 / 137 Pred. true: 533 / 73	false / true Pred. false: 1797 / 93 Pred. true: 392 / 47
accuracy	80.81% (+/- 10.86%)	79.18%
sensitivity	34.70% (+/- 28.83%)	33.57%
specificity	83.76 (+/- 13.20%)	82.09%
lift	200.37	178.10%
AUC	0.636 (+/- 0.096%)	0.627
misclassification cost	-0.140 (+/- 0.150%)	-0.114

Approach used to find the best SVM model:

The best SVM model that was built resulted in an AUC of -0.114 for the validation set, which implies that SVM is one of the worst performing models for this task, particularly in comparison to the decision tree and Naïve Bayes model. The kernel type, C-value and cut-off threshold were once again explored in this model, however, the performance remained low regardless of any adjustments to values, parameters or operators that were made.

k-NN:

Best Parameter Values	
numerical measure	Cosine Similarity
k	70
cut-off threshold value	0.900

	Training Performance			Validation Performance		
confusion matrix		false	true		false	true
	Pred. false	2737	103	Pred. false	1810	64
	Pred. true	546	107	Pred. true	379	76
accuracy	81.42% (+/- 1.73%)			80.98%		
sensitivity	50.89% (+/- 10.26%)			54.29%		
specificity	83.37% (+/- 1.64%)			82.69%		
lift	272.89% (+/- 52.08%)			277.87%		
AUC	0.731 (+/- 0.051)			0.740		
misclassification cost	-0.273 (+/- 0.096)			-0.294		

Approach used to find the best k-NN model:
After optimizing the numerical measure, k-value, and cut-off threshold parameters, our most optimal k-NN model resulted in a misclassification cost of -0.294, which represents the best performance compared to the previous models that were attempt up to this point. Much like before, the numerical measures that were explored include Euclidean, Camberra and Manhattan distances, along with correlation and cosine similarities. Furthermore, k-values in the range of one to one hundred were explored as well, with a variety of steps.

Neural Network:

Best Parameter Values	
training cycles	150
learning rate	0.100
momentum	0.600
cut-off threshold value	0.900

	Training Performance	Validation Performance
confusion matrix		

	false	true
Pred. false	2727	109
Pred. true	556	101

	false	true
Pred. false	1842	65
Pred. true	347	75

accuracy	80.96% (+/- 1.30%)	82.31%
sensitivity	48.12% (+/- 10.11%)	53.57%
specificity	83.07% (+/- 1.63%)	84.15%
lift	254.54% (+/- 39.34%)	295.66%
AUC	0.752 (+/- 0.056)	0.762
misclassification cost	-0.246 (+/- 0.079)	-0.302

Approach used to find the best neural network model:

After exploring different values for the training cycles, learning rate, momentum and cut-off threshold value, the best neural network model generated a misclassification cost of -0.302 for the validation set, which outperformed all of the previously mentioned models. In other words, this was our best performing model out of all the ones that were tried up to this point.

Logistic Regression:

Best Parameter Values	
solver	L_BFGS
cut-off threshold value	0.900

	Training Performance	Validation Performance
confusion matrix		

	false	true
Pred. false	875	32
Pred. true	220	38

	false	true
Pred. false	905	31
Pred. true	189	39

accuracy	78.37%	81.10%
sensitivity	54.29%	55.71%
specificity	79.91%	82.72%
lift	245.13%	284.44%
AUC	0.752	0.802
misclassification cost	-0.268	-0.307

Approach used to find the best logistic regression model:

Representing our best overall model for task 2, the logistic regression model generated a misclassification cost of -0.307 for the validation set. As a result, it is unsurprising that this model also produced the highest profit compared to our other two submissions for the second task. Other than the typical exploration of the solver parameter and various cut-off threshold values, we once again decided to experiment with forward selection and backward elimination. Forward selection ended up generating a more negative misclassification cost, and therefore that was chosen as our final model for logistic regression.

Ensemble:

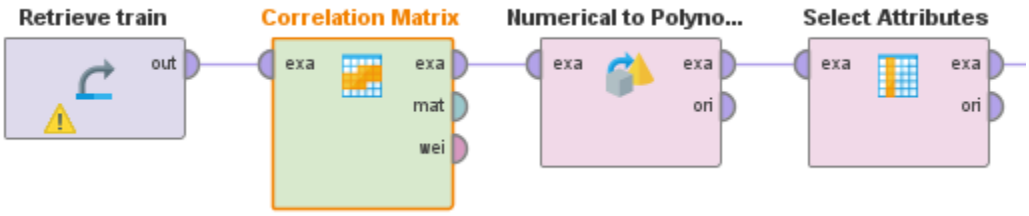
	Validation Performance		
confusion matrix		false	true
	Pred. false	1508	45
	Pred. true	681	95
accuracy	68.83%		
sensitivity	67.86%		
specificity	68.89%		
lift	203.66%		
AUC	0.708		
misclassification cost	-0.279		

Approach used to find the best ensemble model:

Our best ensemble model for task 2 ended up being the neural net operator with adaboost; it produced a misclassification cost of -0.279. We used misclassification costs to compare our various models. Alongside neural network, we also implemented K-NN with adaboost, but it only resulted in a misclassification of 0.00. Additionally, after testing KNN with bagging it resulted in a better misclassification of -0.252. We also implemented logistic regression with bagging to compare performance; this model had a misclassification cost of -0.252. Not only that, but we also implemented neural net with bagging, but it resulted in a worse misclassification cost of -0.246. Similar to our approach to task 1, we included a vote operator with neural net, k-NN, logistic regression, and SVM. This model only resulted in a misclassification cost of -0.010. Lastly, we attempted to implement the random forest operator within an optimize parameter operator. With the optimized random forest we were able to achieve a misclassification cost of -0.238. After implementing and comparing our various ensemble models, we came to the conclusion the neural net operator with adaboost provided the best performance by delivering the lowest misclassification cost of -0.279.

Appendix

Appendix 1:



Appendix 2: