Machine-Learning Algorithms for Predicting Automobile Repurchase

Benedict Brunker

28/04/2024

36106 - Machine Learning Algorithms and Applications
Master of Data Science and Innovation
University of Technology of Sydney

Table of Contents

1. Business Understanding

a. Business Use Cases

Sales of used cars represent more than 65% of all cars purchased in Australia, totaling 206,417 sales in 2023, out of 303,732 used vehicles listed for sale (AADA & AutoGrab, 2023). The figures on the average sale price of these vehicles are less solid, but we could estimate a rough average price at around $35,000 (Ballard, 2023). The market is booming, and the potential revenue is huge, but purchasing a vehicle is a major financial commitment for the average consumer, with the average days to complete a sale hovering around 50 (AADA & AutoGrab, 2023, p.3).

Our company has a large existing customer base, on which we have amassed a large database. Using this data, we can build Machine-Learning Algorithms (MLAs) which powerfully predict which of our customers are most likely to repurchase an additional vehicle. Armed with such a model, we can more efficiently target our advertising and promotions. If such a model succeeds, and is intelligently deployed, we can capture a greater share of the used-vehicle market.

b. Key Objectives

Our key objective is to maximize the efficiency of our advertising efforts by building and deploying a classification model which can powerfully predict the likelihood that a given customer will purchase an additional vehicle from us, given some set of characteristics about that customer.

The key stakeholders in this project are:

1. **Shareholders,** who expect to see the maximum return on their investments. We believe this project, if successful, could expect to generate revenue in the tens of millions, stipulating an average sale-price of $35,000 and a potential sales-pool of 2000 customers.
2. The **executive team,** who have taken on the fiduciary responsibility to protect the best interests of our shareholders. If the project succeeds, this is all but assured.
3. Staff, and in particular, our **public relations** department**,** who can maximize the efficient allocation of their budget by using a machine-learning model to target their efforts.
4. **Customers**, including future purchasers who will benefit from our products, outreach and promotions; and non-purchasers who will avoid being bothered by unwanted advertising.

2. Data Understanding

Our dataset contains a range of information about customer profile and behaviour, with a focus on the customer’s experience with their existing vehicle, answering questions such as:

- What model of vehicle does the customer own? (car_model)
- How much has the customer paid in scheduled services? (sched_serv_paid)
- How many different dealers had the customer visited to service their vehicle? (mth_since_last_service)

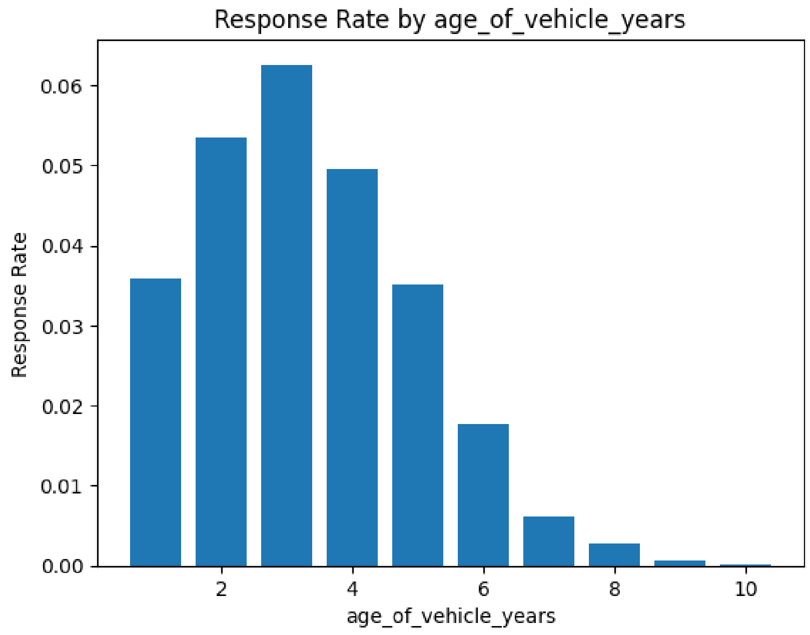
The dataset is already very clean, and contains 131,337 observations, which is sufficient for training, validating and testing our models. Most of these variables are denominated in deciles, with 1 representing the bottom 10% of the distribution for a given variable, and 10 representing the top 10%. For example, a 1 in the ‘sched_serv_paid’ column represents that this customer is in the bottom 10% of expenses paid for scheduled services, while a 10 represents that the customer is in the top 10%.

Four variables contain text values representing some category, these are: ‘age_band’, ‘gender’, ‘car_model’ and ‘car_segment’. Fortunately, these variables only admit of a manageably small number of unique values.

The main data quality issue is the large number of missing values in two columns: ‘age_band’ contains 112,375 missing values and ‘gender’ contains 69,308.

Our target variable is labelled ‘Target’ and is represented in binary: 0 represents “this customer purchased only one vehicle”, while 1 represents “this customer purchased an additional vehicle”. There is a heavy class imbalance in the dataset, with 127,816 0s and 3521 1s (approximately 2.68% of the total). This is unsurprising – customers who purchase a second vehicle are rare – and gives us no reason to doubt the reliability of the data.

Figure 1



We can observe some interesting trends in the data by calculating a *response rate* for each unique value in a column, and observing the results. By *response rate*, we mean simply the proportion of 0s to 1s for each value. For example, *Figure 1* (left) displays a barchart showing the response rate for ‘age_of_vehicle_years’.

We can see that the response rate rises before peaking at the third decile for vehicle age, then steadily declines. It is interesting that charting the response rate like this reveals a normal probability distribution. We might intuitively expect that the older the vehicle, the more likely a customer will be to purchase a new one, but this is not the case. The distribution makes sense when we consider that the customers with the oldest vehicles are precisely those who have demonstrated their resistance to purchasing a new one. This is an important insight in that it cuts against the “common-sense” instinct that the customers with the oldest vehicles would be most receptive to buying new ones.

This normal distribution in response rate is fairly common in the dataset, though in most cases, like ‘sched_serv_warr’, the normality of the distribution is more approximate (cf. *Figure 2*).

Other variables, like ‘sched_serv_paid’, which groups customers into deciles based on how much they paid for scheduled services, exhibit a right-skewed distribution, as shown in *Figure 3* overleaf. Again, the distribution in response rate surprises the natural expectation that those who pay more in services would be

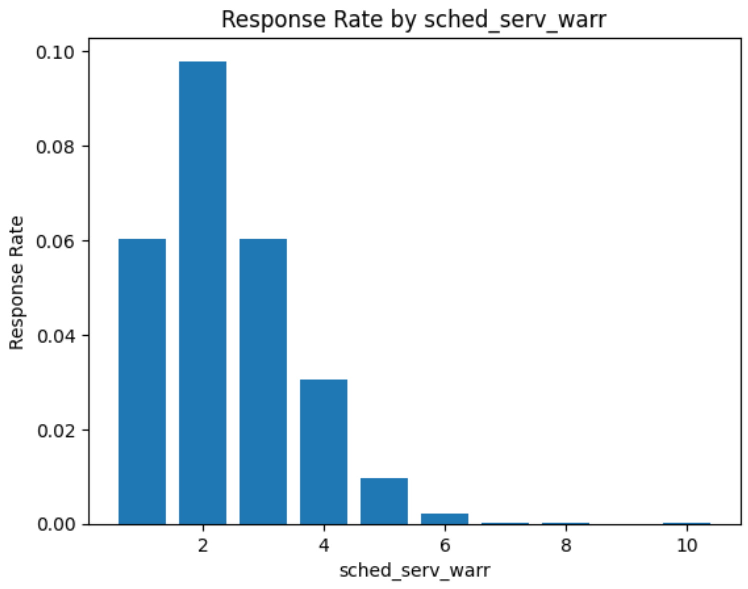
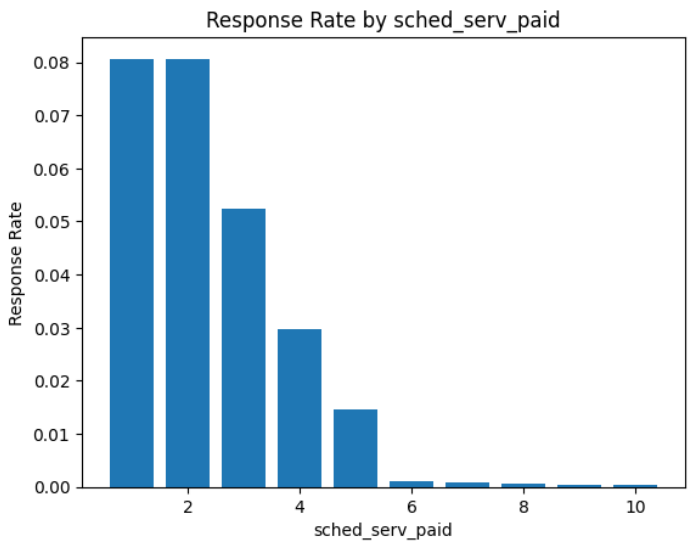


Figure 2

more likely to cut their losses and purchase a new vehicle. The distribution we see may be explicable by the phenomenon economists call *the sunk cost fallacy*, whereby the more an individual invests into something, the more likely they become to invest more regardless of a rational assessment of the rewards (more colloquially known as “throwing good money after bad”).

Figure 3



This analysis of response rate also helped us determine whether categorical variables like ‘car_model’ and ‘car_segment’ could be assigned ranks, or whether they were merely nominal (that is, whether their values were arbitrary with respect to order). In both cases the analysis suggested that these variables could not be ranked, as shown in *Figure 4* and *Figure 5* below.

Figure 4

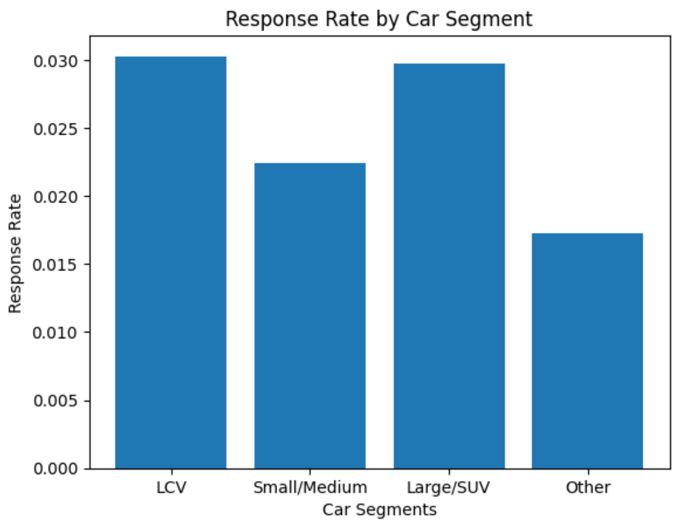
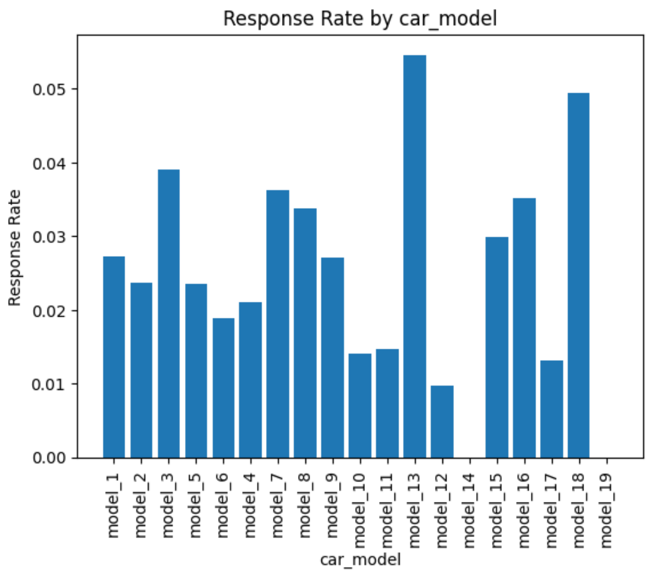


Figure 5



3. Data Preparation

We dropped two columns, 'age_band' and 'gender', due to the prevalence of missing values in these columns. We also dropped 'ID', since it essentially functions as an index of the customer, and is not useful information for any MLA. However, if we want to use the model to target customers in this database, ID would of course have to be restored so as to match the customer profiles with the actual customers.

We used one-hot encoding to create dummy variables for the two remaining categorical variables: 'car_segment' and 'car_model'.

In our first experiment we built Random Forest models (cf. §4 below) which used all the remaining variables as features (*X*). Based on a Feature Importance analysis conducted on our best Random Forest models (cf. *Figure 6* below), we curated a narrower selection of features (*X2*) which we hoped might be better suited to the simpler models we built, like Decision Trees and Logistic Regressors. X2 takes the top 11 most important features displayed in the chart below. A steep cliff can be observed separating 'non_sched_serv_paid' at 1.34% importance, and 'car_model_10' at 0.19% importance. Setting the cut-off point here gave us a manageable number of features for simpler models, while still capturing over 95% of feature importance. It also meant we neatly excluded all dummy variables, leaving only the decimals.

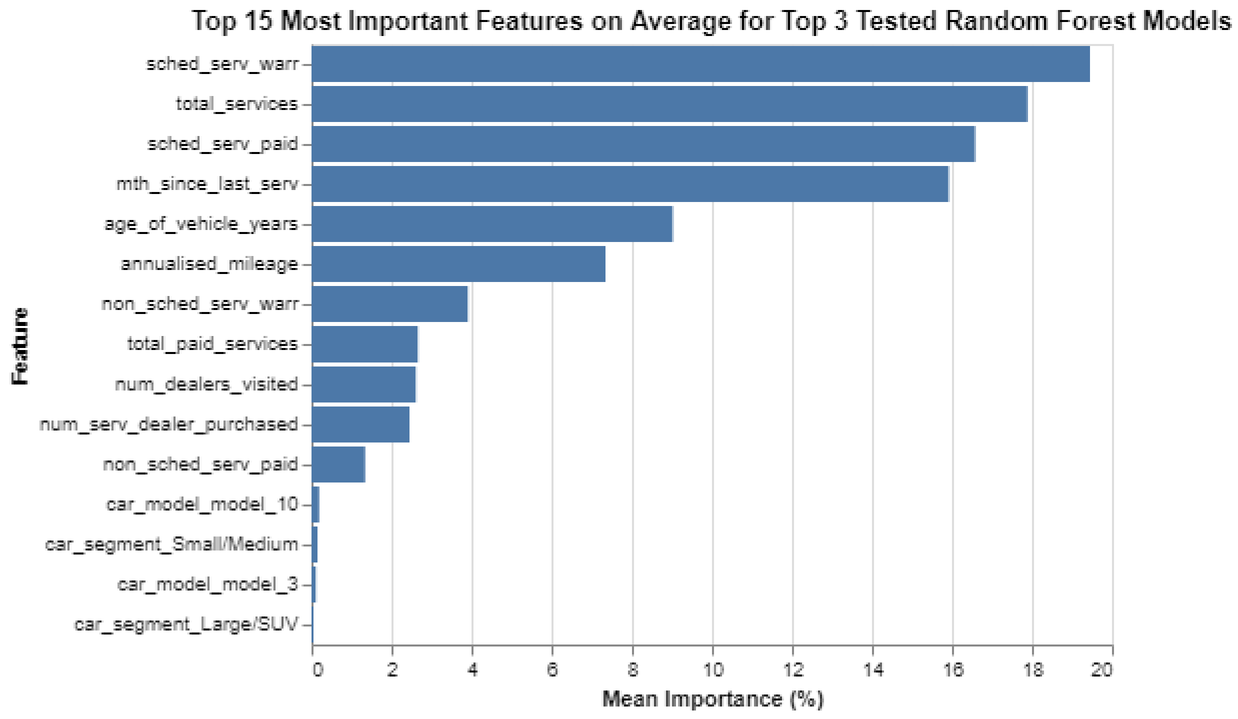


Figure 6

For Experiments 4 and 5, we scaled X using the Min-Max scaling method, which scales all features as values between 0 and 1. The decimalized variables are already scaled from 1-10, so this process was to ensure parity between decimals and dummy variables. X2 contained only these decimals, and so didn't require scaling. The Min-Max method seemed logical since we had no outlier values, and already had 0 and 1 as the outer ranges for the binary variables. Scaling was not necessary for Experiments 1-3, in which we built Decision Tree-based models, for which the scale of features are irrelevant, since only one feature is considered at each node of the tree (more below).

Prior to any modelling, our data is split into three sets:

1. 20% is reserved for truly unseen **testing** data, which is unused until the end of each experiment, and is only used to test models which are serious candidates for deployment. No further adjustment of the models can take place once the test data has been used, since the purpose of test data is to simulate real-world deployment, in which the model cannot adjust in real-time based on its results.
2. Of the remaining 80%, another 20% is reserved for **validation.** This data is used to assess the difference in a model's performance on unseen data relative to its performance on training data, and then adjust parameters to improve performance.
3. The remaining data is used to **train** the models.

We use the same data-split for all our modelling experiments to ensure a controlled comparison of the models' performance.

As we've mentioned, Negatives (0) outweigh Positives (1) by approximately 36:1. We used the following strategies to deal with the imbalance:

1. When splitting the data, we **ensure the splits are stratified** by the distribution of the target class, so that the training, validation and testing sets all contain roughly the same proportion of Negatives to Positives. This is a necessary precaution because splitting the data purely at random might result in quite divergent distributions of the target class, and so make it difficult to assess the models' performance on training against unseen data.
2. **Tuning the models' hyperparameters** to bias the models slightly towards predicting Positives over Negatives. With a class imbalance this strong, classification algorithms will always find it overly easy to detect patterns relating features to the strongly predominant Negative class. Therefore, the models need to be told to assign much greater significance to the Positive class. We do this by tuning the *class_weight* hyperparameter to 'balanced', which assigns weights to each class in inverse proportion to their distribution in the training data.
3. **Assigning special importance to Recall Score** in assessing model performance. We will discuss the reasons why in §4 and §5 below.

4. Modeling

We built five different kinds of models over the course of five experiments:

a. **Decision Trees**
b. **Random Forest Classifiers**
c. **Extra Trees Classifiers**
d. **Logistic Regressors**
e. **Support Vector Classifiers**

a. Decision Trees

These models classify data by asking yes/no questions of the observations regarding their features at the tree's *nodes*, and recursively splitting the data based on the answers until the tree reaches its *leaf node*, at which no further splitting takes place. The tree then classifies all the samples in that node based on which class forms a majority. We built Decision Trees in our third experiment but mention them here first because in the first two experiments we built models which *ensemble* a number of these trees together.

Decision Trees are accepted in the industry as a powerful classification model. They also have the advantage of ease of interpretability: the tree can be easily visualized and understood. By analyzing well-performing Decision Trees, we can see which of the features appear to be most important for the model in classification.

Two parameters were most important in improving model performance, especially when measured by Recall Score. The first is *max_depth,* which establishes a limit as to how deep the tree can grow, that is, how many questions it is allowed to ask of the observations. Limiting *max_depth* was important because of the class imbalance in the data. Because there are so many negative cases, a Decision Tree will tend to classify samples as negative if allowed to ask enough questions. Limiting *max_depth* cuts short the tree's questioning, forcing it to classify samples earlier than it would otherwise, which can bias it somewhat to making Positive classifications so long as we also tune *class_weight* to 'balanced'. *Min_samples_leaf* has a similar effect in preventing aggressive over-splitting leading to Negative classification. This parameter forces the tree to stop splitting and form leaf (end) nodes once the samples in a node reach some minimum number.

Our best performing Decision Tree was named **tree2a,** and took *max_depth=10, min_samples_leaf=20* and *class_weight='balanced',* while its other parameters were the defaults for SciKitLearn's DecisionTreeRegressor module.

b. Random Forest

These models use an ensemble of Decision Trees, and classify each observation based on taking a majority "vote" from all the trees. These were the first models we built, in Experiment 1. We started out with Random Forests for three main reasons:

1. They are widely accepted in the industry as powerful classification algorithms.
2. Relative to algorithms like Support Vector Machines, they are relatively easy to explain and interpret.
3. They return an attribute, *feature_importances_,* which shows us how important the features fitted to the model were in classifying the data. We hoped this attribute would help us identify the most predictive features in our dataset, giving us general insights into what features make a customer most likely to purchase an additional vehicle, which is generally useful information for our business. We also hoped this attribute would help us narrow down feature selection for our other models, creating a leaner set of features which would reduce noise.

Along with the two important parameters mentioned in §a above, we will mention two other parameters that were important in improving model performance in our experiment with Random Forests. The *max_features* parameter puts a limit on the number of features each tree can use to make predictions. We noticed that increasing this parameter tended to make models more precise at a small cost to sensitivity. However, we ultimately determined the default value of 'sqrt' (the square root of the number of features, ≈6) to strike the optimal balance between Precision and Recall.

The *n_estimators* parameter determines the number of trees in the forest. Increasing this parameter can help reduce variance (the drop in performance when the model encounters new data) when paired with other parameters like *max_features, max_depth* and *min_samples_leaf*, which limit each individual tree's fitting to the training data. If each tree in the forest is prevented from overfitting, but observations are classified by ensembling a large number of predictions together, Random Forests can strike a healthy balance between reducing bias (initial training error) and variance (the increase in error on unseen data).

Our best performing Random Forest model was named **rf_gs.** 'gs' stands for GridSearch, meaning the model's ultimate parameters were determined using a Machine Learning algorithm which optimizes the parameters within a specified range. The model's parameters were:

```
n_estimators=300, min_samples_leaf=5, criterion='entropy', max_depth=5,
class_weight='balanced'
```

We instructed GridSearch to return parameters which maximized the model's Recall Score, because our focus in the first experiment was to see how sensitive we could make the model, and how that would affect Precision Score.

a. ExtraTrees

ExtraTrees are a variant of RandomForest which introduce "extra" randomness into how the model makes classifications, and so can further reduce variance over RandomForest. ExtraTrees can perform better than RandomForest for certain classification problems, so we wanted to see if these models could improve further over the RandomForest models. ExtraTrees also have the advantage of cheaper computational expense.

ExtraTrees' parameters are nearly identical to RandomForest's, and the same logic as applied to the Decision Tree and RandomForest models seemed to hold for ExtraTrees too. Our best performing ExtraTrees model was called **ET_recall_opt_2.** Its parameters were determined by GridSearch, optimizing for Recall Score:

```
n_estimators=250, criterion='entropy', min_samples_leaf=9, max_depth=9,
class_weight='balanced'
```

Its other parameters were the defaults for SciKitLearn's ExtraTreesRegressor module.

b. Logistic Regressors

Logistic Regressors are a basic type of classification algorithm which classify data by drawing a line that best separates the two classes when the samples are plotted together by their features. They take a fundamentally different approach to the three tree-based models above, so we thought it worthwhile experimenting with these models in case the classes in our data were more susceptible to this kind of regression solution. Logistic Regression also has the advantage of returning a Probability Function which is relatively easy to interpret. This means that they could make not just a binary prediction – the customer buys an additional vehicle or does not -- but also help us understand the probability that the customer purchases an additional vehicle. This could potentially offer an additional refinement to our allocation of advertising resources, whereby we could target customers according to probability of repurchase and not just a binary Yes/No prediction.

c. Support Vector Classifiers

Support Vector Classifiers (SVCs) are more similar to Logistic Regressors than the tree-based models, in that they also draw a classification boundary, a line separating the data-points into different classes, by calculating the "support vectors" between data-points, meaning their distance from the boundary. However, SVCs are often much more powerfully predictive models than Logistic Regressors for certain kinds of problems. Given the failure of our experiment with the latter, we wanted to see if these more sophisticated algorithms would perform better than the tree-based models.

We found that SVCs might become competitive with our best ensemble models (RandomForest and ExtraTrees) by appropriately tuning some key parameters. The best performing models used the polynomial kernel trick (kernel=’poly’), meaning that they used a polynomial function to draw a classification line separating customers into repurchasers and non-repurchasers. The best polynomial SVCs tuned the degree parameter to 5, meaning that the polynomial function was raised to the fifth power. We tuned the C parameter, which controls Regularization strength: lower C values will apply greater penalties to some of the features in order to simplify the models and reduce variance, while higher C values will do the opposite. The best performing models tended to take C between 0.05 and 0.1. We also tuned the gamma parameter, which controls how many data-points the SVC will take into consideration when drawing the classification boundary. The models with the best F10 Score (balancing between Precision and Recall) took the default ‘scale’ value for gamma, which automatically scales gamma according to the distribution of features, while the most sensitive models took a value between 0.001 and 0.005. As an example, here are the parameters for SVC6c, a model which performed well on F10 Score:

```
C=0.1, kernel='poly', degree=5, class_weight='balanced'
```

Unfortunately, we could not feasibly use automated hyperparameter tuning programs like GridSearch for these models as we could with the other models, because of the prohibitive computational expense of fitting these kinds of models. Even our manual tuning had to be restrained by these long fit times: some parameter values would simply not allow the models to fit in a reasonable amount of time. Therefore, even our best performing models almost certainly had sub-optimal parameters, and may have overfitted to the validation set, since we were manually tuning parameters to improve performance on that set.

5. Evaluation

a. Results and Analysis

i. Classification Metrics

The most important metrics for evaluating how our models meet our business objective are Recall Score and Precision Score.

Recall Score, also known as *sensitivity*, measures the ratio of True Positives (TP) (the model correctly predicts that the customer will purchase an additional vehicle) to TP + False Negatives (FN) (the model incorrectly predicts that the customer will not purchase an additional vehicle). In essence, Recall Score measures how many potential sales the model would miss. This measure is especially important given the enormous inequality between the opportunity cost incurred by a missed sale, and the relatively miniscule cost of targeting that customer. If we stipulate an average sale price of around $35,000 per vehicle, every False Negative could represent an opportunity cost of around $35,000.

Precision Score measures the ratio of True Positives to the sum of True Positives plus False Positives (FP) (the model incorrectly predicts that the customer would purchase a vehicle). For this project, precision translates to the efficiency of a targeted marketing campaign, or the “hit-rate” of that campaign. For example, a Precision Score of 50% would mean that half the targeted advertising efforts were successful.

If we only cared about Recall Score, we could easily maximize it by building a “hyper-optimistic” model that always predicts the positive class. Such a model would of course be useless, because it would “target” the entire market, not making any discrimination between customers. Precision Score therefore represented an important constraint on maximizing Recall Score.

Because there is necessarily a trade-off between Precision and Recall, data scientists use F-beta Scores to calculate a balance between the two. The most common measure is F1 Score, which finds the harmonic mean of the two, treating them as equally important. For our purposes, however, Recall is much more important than Precision, because the potential revenue of a sale vastly outweighs the expense of targeted advertising. We therefore experimented with higher values for beta in our F-beta Scores, eventually settling on F4 and F10 Scores, which weight Recall 4 and 10 times more highly than Precision respectively.

For a model to be seriously worthy of consideration for deployment, **we set a floor of 98% Recall Score and 10% Precision Score.** This would translate to F10 Scores above 90% and F4 Scores above 75%. Our best models could achieve over 98% Recall on unseen data while remaining above 10% Precision. We set these floors after assessing the performance of our best performing models in our first experiment with RandomForest. The exact numbers are somewhat arbitrary, but are based on our sense of how well the models could reasonably perform in both directions. We didn't want to sacrifice any more than 2% Recall due to the potentially large opportunity cost in missed sales, but we also wanted to retain a 10% hit-rate as the minimum standard for targeted advertising. F10 Score is often used to judge the trade-off between Precision and Recall for the models that exceed both these benchmarks.

ii. Tree-based Models

By the end of Experiment 1, there were three models which we considered serious candidates for deployment: RF30, RF30A and RF_GS. RF30 and RF30A had marginally higher Recall Scores than RF_GS, but at a significant cost to their Precision. We ultimately decided that **RF_GS** was the best model for our business objective in balancing Precision against Recall, as it beat the other two models on F10 Score, meaning its relative gain in Precision was more than 10 times greater than its loss in Recall.

Table 1

Top performing Random Forest models on testing, ranked by F10 Score

Model	**Recall**	**Precision**	**F1**	**F4**	**F10**	**Accuracy**	**FNs**	**FPs**
RF_GS	98.01	13.83	24.24	72.17	92.44	83.58	14	4299
RF30	98.15	12.88	22.78	70.65	92.12	82.16	13	4672
RF30A	98.86	10.31	18.68	65.69	91.12	76.93	8	6052

Figure 7 and *Figure 8* below show us the model's performance on the test data. The model would classify 98% of potential repurchasers correctly and miss 2%. It would falsely label 17% of single purchasers as repurchasers, and correctly label the remaining 83%.

Figure 7

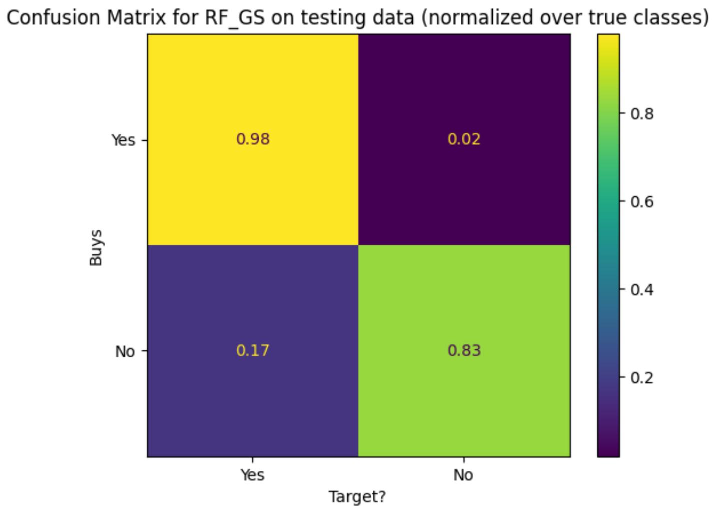


Figure 8

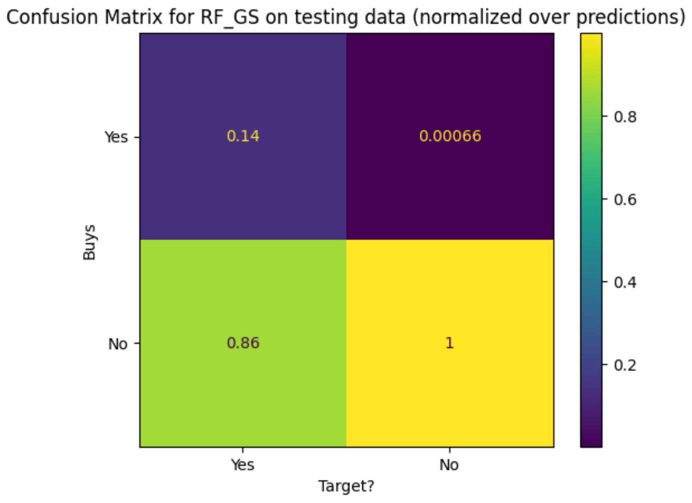
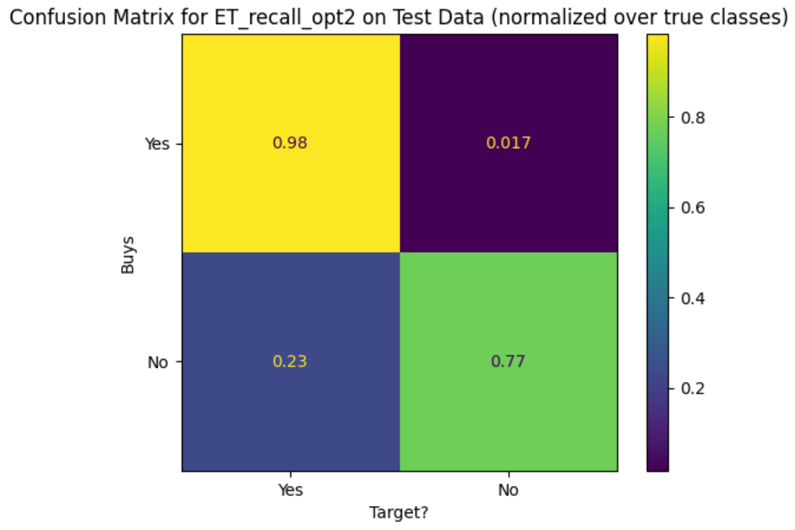


Figure 9



Our best performing ExtraTrees model was named **ET_recall_opt2.** This model was more sensitive than RF_GS, with a Recall Score ~0.29% higher, but it was ~3.4% less precise. RF_GS' F10 Score is ~ 1.69% higher than ET_recall_opt2's, indicating that it better balances Precision against Recall for our purposes.

The stand-alone Decision Tree models failed to be competitive with the ensemble models. Our best performing Decision Tree when measured by F10 Score was **tree2a,** but this model fell well short of our 98% Recall benchmark. A nearly 1-in-3 hit-rate would be welcome, but we would incur too high an opportunity cost by missing many more potential sales.

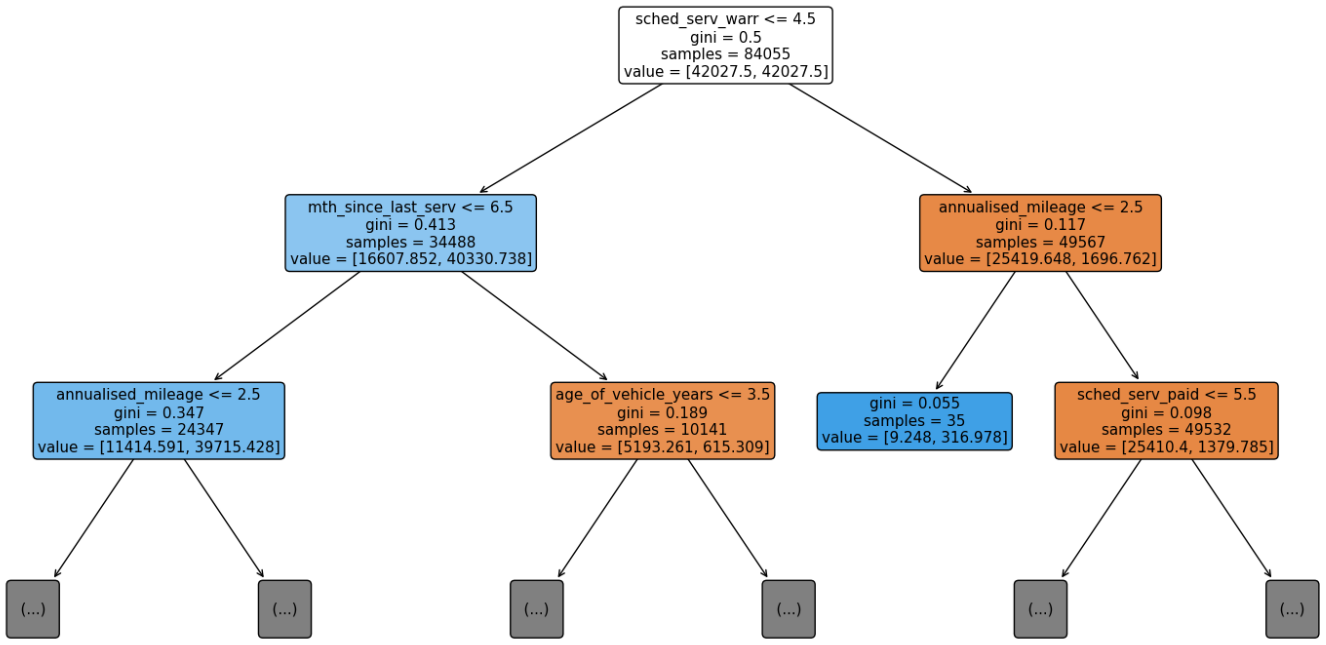
Table 2

Test Scores for Tree-Based Models (ordered by Recall Score)

Model	**Recall (%)**	**Precision (%)**	**Accuracy (%)**	**F4 (%)**	**F10 (%)**
ET_recall_opt2	98.30	10.46	77.41	65.80	90.75
RF_GS	98.01	13.83	83.58	72.17	92.44
tree2a	95.17	32.23	94.51	85.36	93.37

However, analyzing tree2a's decision structure can still offer valuable insights as to how this kind of Machine Learning Algorithm would classify our customer base. The whole tree is too large to effectively visualize here, but *Figure 10* (overleaf) shows its upper branches. We can see for instance that the first split the tree makes is that it creates a node with all the samples in deciles 1-4 of 'sched-serv-warr' which has a majority of positive samples (colour coded as blue), and a node with deciles 6-10, which has a negative majority (colour coded as orange).

Figure 10: tree2a's upper decision nodes



iii. Logistic Regression and Support Vector Classifiers

Our experiment with Logistic Regressors was the only one that could be counted as an unqualified failure. Even our best performing models scored much lower than the tree-based models from earlier experiments. For example, the lowest Recall Score we could achieve was with a model named *LR_balanced_saga_1,* at 86.68%, and this model had a Precision Score of 9.25%. Furthermore, many of our models failed to converge, meaning they failed to find a solution to the Logistic Regression problem within the maximum iterations (*max_iter*) allowed them. Increasing *max_iter* incurred prohibitive computational costs, and became too time-consuming to be practical. We had to conclude that our data was simply not susceptible to a Logistic Regression solution, as these models seemed to lack the ability to deal with the imbalanced classes in our target variable.

The Support Vector Classifiers (SVCs) were much more effective, though were similarly computationally expensive. However, even these models failed to match the performance of our best RandomForest or ExtraTrees models (see *Table 3*), along with the additional drawbacks that they are less easy to interpret and more computationally expensive. Two models are worthy of note. The first is **SVC6c**, which had the highest F10 Score of all our SVCs, but still failed to reach our 98% Recall target. The second is **SVC9a**, which was our only model to clear 99% Recall Score, but had very low Precision Score at 6.46%, yielding an F10 Score of 86.92%.

Table 3

Test Scores for Tree-Based Models vs. Support Vector Classifiers (ordered by F10 Score)

Model	**Model Type**	**Recall (%)**	**Precision (%)**	**Accuracy (%)**	**F10 (%)**
RF_GS	RandomForest	98.01	13.83	83.58	92.44
ET_recall_opt2	ExtraTrees	98.30	10.46	77.41	90.75
SVC6c	Support Vector Classifier	97.16	11.72	80.30	90.62
SVC9a	Support Vector Classifier	99.29	6.46	61.43	86.92

We believe we did not fully capture the power of SVCs for this project, due to the limitations discussed in *§4.e* above. We think it plausible that we did not optimize the parameters for the SVCs, and it remains possible that a better optimization could reach as high as 99% Recall Score while preserving a 10% Precision Score, but this is only a conjecture.

iv. Summary and Recommendation

In summary, our preferred model for deployment is **RF_GS**, because:

1. It meets the 98% Recall target and comfortably clears the 10% Precision floor on unseen testing data.
2. It has the highest F10 Score of the models we most seriously considered, meaning it finds a good balance between Precision and Recall. It also has the highest Accuracy Score, meaning it has the most correct classifications overall relative to total samples.
3. It is computationally cheap and its mechanics can be understood without too much effort or technical background.

4. Business Impact and Benefits

In RF_GS, we have a model that will fulfill our main business objective: it will allow us to more efficiently target our advertising efforts to only those customers most likely to purchase an additional vehicle. We can estimate the model will miss only 2% of potential repurchasers (capturing 98%), while allowing us to exclude 83% of the customer base from targeted advertising. Another way to look at it is that the model would give us an estimated 14% hit-rate in a targeted advertising campaign (see *Figure 7* and *Figure 8*).

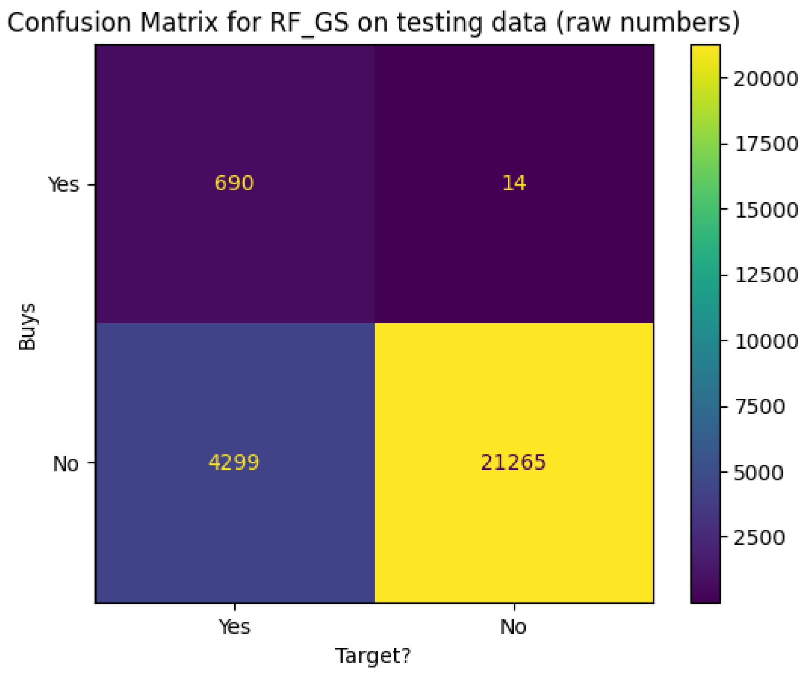


Figure 11

Figure 11 (left) shows the raw classification numbers from the test data. If we conceive this as a hypothetical scenario for a targetted advertising campaign, with 26268 data profiles corresponding to RF_GS' fitted features, we can roughly predict the model would result in 690 successful sales from a total of 4989 outreach efforts. 4299 of those efforts would fail to pay off. 14 potential sales would be missed, while 21,265 of the profiles would be excluded from the campaign.

If we further stipulate an average sale price of $35,000, we can estimate the gross revenue of an advertising campaign using this model at $24,150,000. We leave it to the Accounts Department to sanity-check this number, which should be seen as an upper estimate and only a very rough one at that. Accounts and Public Relations will also have to coordinate to estimate the expected costs of targetted advertising per-customer, and conduct a proper cost-benefit analysis. Our guess though is that profit from successful sales will still vastly outweigh the expense in targetted advertising per customer — the question is only by how much.

By the same premises, RF_GS' False Negative predictions would represent an estimated opportunity cost of ~ $490,000. Subtracting opportunity cost from estimated revenue would give the model an estimated value-add of ~ $23,660,000 (accounting only revenue, not expenses or assets).

5. Data Privacy and Ethical Concerns

The data privacy implications of this project are minimal, since we have next to no demographic data. Our only demographic variable in the dataset, ‘gender’, was dropped early on in the project. We have identification data, but these are anonymous and arbitrary numbers that were not used for modelling. The ID data would need to be re-merged with the observations if we wanted to try deploying the model for use with the customers represented in the dataset. Potential data concerns would therefore enter into the picture on model deployment, when the feature profiles will have to be matched with customer contact details and demographic information. However, these are areas of the data pipeline which did not figure into our project.

The primary ethical concern is whether the model is used to ‘prey’ on customers in difficult situations with their vehicles. Automobile faults can incur significant psychological stress, and we know from Feature Importance Analysis (see *Figure 6*) that customers who have had problems with their vehicle, for example those in the upper deciles for the number of services undertaken, were more likely to purchase an additional vehicle. There is a risk therefore that the project becomes exploitative towards those already in difficult circumstances. Furthermore, if the project is aimed at our own customer base, we would risk introducing perverse incentives which are not only unethical, but will harm our business in the long run. If modelling tells us that customers with a faulty vehicle will be more likely to buy a new one, will we be tempted to deliberately market poor-quality vehicles to our customers? Such an approach would risk destroying our reputation in the market, and is frankly immoral. We need to remain clear that good quality products remain at the core of our business, and Machine Learning Algorithms are only an aid to that, not a substitute.

We also need to attend to the potential risks and impacts for Indigenous Australians. According to Rosier & McDonald (2011):

> Cars in remote Indigenous communities are heavily used and, due to the fact that they are often purchased second hand and used in rough terrain, maintenance is expensive and they often have a short lifespan. (p.1)

When marketing to Indigenous Australians therefore, we have a special responsibility to prevent the further entrenchment of Indigenous disadvantage. The solution is the same as with our broader ethical concern regarding potential exploitation: continue to make the quality of the product the beating heart of the business.

6. Conclusion

The key finding of this project is that we can effectively use Machine Learning Algorithms to classify our market into the customers most and least likely to purchase an additional vehicle. We can use these tools to rationalize how we advertise to customers. Instead of a “one-size-fits-all” approach to marketing, we can fit our advertising to our best leads, using models which can tell us in seconds where we can find our next sale.

We hope our shareholders will benefit from increased revenue translating to profit, and more efficient allocation of their capital. Our executives, department heads and staff will be armed with better information as to how we can use the cutting-edge in Machine Learning to guide our marketing strategy. Finally, customers looking to purchase will be alerted to our superior offerings, while those who are satisfied with their current vehicle will be left well alone.

We recommend now moving to co-ordinate with the Public Relations Department to iron out the details of the model’s deployment, and teaching our sales representatives how to use it. We will move to building a more user-friendly interface for staff. We also would recommend bringing in the Accounts Department to conduct a proper cost-benefit analysis of the model’s deployment.

Should the executive team prefer a still superior model, we can move to drafting a second run of the project. If this option is preferred, we would recommend an upgrade to our current computational resources, and also bringing in Data Science experts, especially those trained in the use of Support Vector Machines, and still more sophisticated models which this project left unexplored. This will also require the continued collection and preprocessing of data to match the basic structure of the dataset used in this project.

We also need to carefully supervise the model’s performance on deployment, ensuring it maintains results, and assessing whether new adjustments to the model need to be made.

References

Australian Automotive Dealers Association & Autograb. (November 2023). *Automotive Insights Report (AIR).* https://www.aada.asn.au/wp-content/uploads/2023/12/AADA-AG-AIR-Nov-2023-Public.pdf.

Ballard, J. (2023, February 6). *OnlyCars.com.au.* 'Used Car Prices Australia 2023'. https://www.onlycars.com.au/news/used-car-prices.

Rosier, K. & McDonald, M. (2011, August). *The relationship between transport and disadvantage in Australia.* Australian Institute of Family Studies. https://aifs.gov.au/sites/default/files/publication-documents/rs4_2.pdf.