40.016 The Analytics Edge

Discrete Choice Modelling

Transportation Analytics

Team 26

August 9, 2024

Prepared by:
1006617 Michael Hoon Yong Hau
1006651 Wong Qi Yuan Kenneth
1006616 Evan Ang Jun Ting
1007492 Nathan Ansel

Course Instructors:
Karthik Natarajan
Bikramjit Das

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

Contents

1 Introduction

Discrete choice models have been used extensively in many areas in economics, marketing, and transportation research. In this competition, we aim to predict the choice among bundles of safety features in cars [7]. The choice-based conjoint data was collected by General Motors to understand consumer's trade-offs for various vehicle features [4].

2 Methodology

As we are working with a discrete choice dataset for transportation analytics, we separated our efforts into two categories: classical models grounded on Discrete Choice Theory, and modern Machine Learning (ML) methods.

2.1 Discrete Choice Theory

The main reference literature used for interpreting the data and investigating discrete choice based models was the paper "On Theoretical and Empirical Aspects of Marginal Distribution Choice Models" [4]. In addition to novel semiparametric discrete choice models proposed by the authors, they also highlight the importance of feature engineering techniques via one hot encoding, and feature selection via parameter significance. A significant portion of our time was dedicated to understanding the author's methods, approaches to applying choice models, and their key findings.

We also note the importance of modelling consumer partworth heterogeneity, and explored different methods such as the Nested Logit model which relaxes the Independence of Irrelevant Alternatives (IIA), and the Mixed Logit model which models the random taste variation present in different consumers. Different parameter distributions were tested based on repeated iterations, however we found that **assuming consumer homogeneity** with the Multinomial Logit (MNL) model yielded the lowest log-loss scores. Similar to the authors, we also found that there was little variation in results when applying either of the 3 models.

The final discrete choice model we selected was thus the MNL model, due to its relative simplicity and performance. We employed one hot encoding as we recognise that the safety features and their levels may not necessarily be ordered in terms of customers' desires. As a cautionary note, we prevented the dummy variable trap by removing one variable for each categorical variable that we encoded. To illustrate the advantage of one hot encoding, after building the model on the training data that was split based on tasks, we found that there were several variables such as `AF3` and `BU6`, that were seen to not be statistically significant, where the other levels of `AF` and `BU` were. Significant encoded variables were interpreted to be important in affecting the customer's decision on choosing the package, and vice versa. As such, the final MNL model, where all variables were statistically significant, had the following variables removed: `AF3, BU6, FA2, FC2, FP4, GN1, GN2, HU2, LB1, LB2, LB4, LD3, MA2, MA4, NS2, NS3, NS5, NV3, PP3, RP2, SC4 and TS3`.

2.2 Boosting

Moving on to modern machine learning (ML) methods, based on past experience of our team members, we have seen that gradient boosting methods generally perform exceedingly well on Kaggle competitions [1]. Boosting methods in ML primarily seeks to reduces bias and variance, utilising decision trees as base learners with regularisation techniques to enhance model generalisation. Our group specifically chose the eXtreme Gradient Boosting (XGBoost) variant, which is designed for computational efficiency and resistance to over-fitting. We were aware of several other variants of boosting algorithms such as Adaboost, Catboost, and LightGBM, however we found that they were

difficult to optimise and properly fine tune. Our group has seen that Kaggle winners generally leverage the fast computation times of XGBoost to experiment with feature engineering techniques, as this was what gave them a significant edge over other participants. Here, we decided to remove the following features before training our XGBoost model:

1. `Task, Case and Number`
   We intuitively felt that these features would not contribute to the building of the model, as they were just indicators of the number of questions answered.

2. `educ, gender, region, segment, ppark, night, miles, urb, income, age`
   We removed the above listed categorical features, as there already were similar features which converted the features into the form of indices.

3. `CC4,GN4,NS4,BU4,FA4,LD4,BZ4,FC4,FP4,RP4,PP4,KA4,SC4,TS4,NV4,MA4,LB4,AF4,HU4, Price4`
   We removed all features associated with choice 4, as it means that the customers did not end up selecting any of the 3 previous packages, and thus all the features just have a value of 0.

As with all ML algorithms, performance of the model is also highly dependent on careful hyperparameter tuning and feature selection. Here, our initial idea was to utilise existing hyperparameter tuning techniques such as built-in grid search methods to iterate through all specified ranges. However, as we wanted to evaluate the performance of our model via our own computation of log-loss, accuracy, and overfitting metrics, we had to improvise and create a custom implementation of this. With this in mind, we had to select the most important parameters that we intend to fine-tune to optimise the resultant log-loss. Most of our understanding of the parameters were obtained from insightful blogs (e.g. `https://gabrieltseng.github.io/posts/2018-02-25-XGB/`) as well as the official R documentation [2]. From here, we found that the most significant parameters to fine tune are:

1. Number of Iterations/Trees, `nrounds`
   As number of trees increases, the training and validation log-loss decreases. However, using too many results in high variance and overfitting when applied to the test set.

2. Learning Rate, `eta`
   Step size shrinkage used per update, in order to prevent overfitting. As the value of `eta` increases, the model becomes more conservative, as the feature weights are shrunken to a greater degree.

3. Maximum Tree Depth, `max_depth`
   The maximum allowed depth of a tree. We found that increasing this value resulted in models which had a better validation log-loss. However, by increasing this value too much, it resulted in high levels of overfitting.

4. Minimum Loss Reduction for Branching, `gamma`
   The minimum loss reduction required to further branch on a leaf node of the tree. We found that increasing this value prevented overfitting, as it was harder for branches to be made. However, by increasing this value too much, it resulted in high levels of bias, and thus ineffective models. This could be attributed to the final tree being too shallow, and thus underfitted.

5. L2 Regularisation Term on Weights, `lambda`
   We found that increasing the value of lambda prevented overfitting, by forcing weights to be small, but not 0. However, similarly to gamma, should the value be too high, it would result in the model being underfitted.

k-fold cross validation was also used to assess the performance of our models. We used a value of k = 5, By dividing the data into 5 equally sized folds and iteratively training and testing the model 5 times, each with different folds, it ensured that the performance of the model was consistent when trained and tested on different folds, thus preventing overfitting when later tested on the test set. It also helped us in the process of hyperparameter tuning, as it allowed us to more consistently see the changes in performance in the model when certain parameters were changed. We subsequently also applied the same method of cross validation when building the random forest model below.

2.3 Bagging

Bootstrap Aggregating (Bagging) is another technique that we attempted, which combines predictions of multiple models to produce a final prediction that is more accurate and stable than individual models. Our group decided to select **Random Forest** (RF) model, which utilises bagging with decision trees as the base models. Similar to the MNL model, we used the encoded safety feature variables as predictors. In addition, we also added the alternative specific effects such as `segmentind, yearind, milesind, nightind, pparkind, genderind, ageind, educind, regionind, Urbind` and `incomeind`. In total, we had 71 predictor variables for this model.

Each decision tree in our random forest model is a classification tree, with only a subset of predictor variables (mtry) being considered for each split. After tuning the parameter "mtry" using the `tuneRF()` function, we found that the ideal "mtry" for us was 16, which is different from the default value of 8, the rounded value of the square root of the number of predictor variables, 71. Another important parameter to consider was the number of trees (ntree) that make up our random forest. Due to the many number of rows used as training data and the large pool of predictor variables, too few trees could end up missing the effect of some features. However, too many trees would lead to the overfitting of the training data. Thus, we iterated through a list of potential ntree values and found the ideal range of ntree values to be from 500 to 700, as seen in Figure 3. We did not consider ntree = 50 as we thought it would be too few trees for such a large dataset. We chose ntree = 500 in the end, as the performance between these models were very similar, and we wanted to minimise the chance of overfitting the model.

2.4 Stacking

Lastly, stacking is a more recent technique in ML that feeds the predictions of numerous base models into a higher-level model known as a *meta-learner* model, which then combines them to get a final prediction. Here, meta-learning combines the strengths of various base models and improves generalisability of the overall model as we fine-tune the weights used to evaluate each of the ensembled models. This can frequently result in higher performance than utilising a single model alone. Similar to the XGBoost model, our group has noticed that stacking methods generally performed well on Kaggle competitions and are quite popular in the community, providing submissions with slight marginal gains in performance which is key to winning competitions [13] [9]. Many hours of reading *Stackexchange* discussions also spurred us to experiment with stacking methods.

There are various stacking methods in ML. Generally, multiple models are initially trained on the training data (base models), and they can be of different architectures. The predictions of these base models on both the training and test data are then used to train the final meta-learner model, and the predictions on the test data is used to predict the final outcome of by meta-learner model. The final meta-learning model could also be any model such as another XGBoost implementation. However, our group also realised that as the model complexity increases, interpretability decreases and there is no guarantee that the meta-learner will generalise well on unseen samples. Thus, the meta-learner model has to undergo another process of parameter fine-tuning, and our group decided to approach this via a more direct method instead.

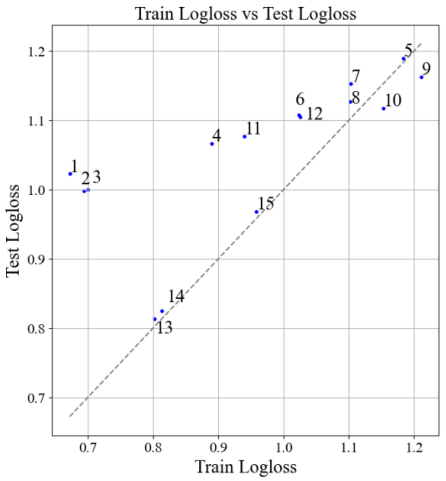
2.4.1 Soft Voting Classifier

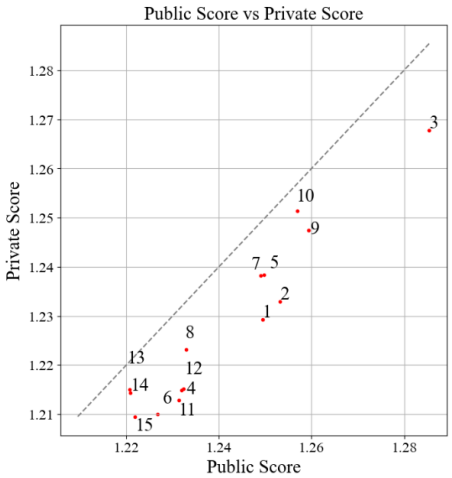
Instead of layering our stacking approach with another XGBoost model, we decided to take the **weighted geometric mean** of the predictions of three base models, namely the **XGBoost**, **MNL** and **RF** models. We chose the geometric mean in favour of the arithmetic mean because the geometric mean will be less sensitive to extreme values, although we found that the stacking model based on weighted geometric mean differed very little from the stacking model based on the weighted arithmetic mean. In ML methods, this is known as a soft voting classifier. In soft voting, the output class is the prediction based on the average of probability given to that class. Soft voting entails combining the probabilities of each prediction in each model and picking the prediction with the highest total probability [1]. Here, to best improve the final prediction performance, the models which we included had to be **varying** in **architecture and performance**, in hopes that they all capture the **different underlying features** of our dataset. Similarly, we had to fine tune the weights used for the final stacking approach, which was iteratively computed over an exhaustive grid search of possible values.

As can be seen in Figure 6, increasing the weights of the XGBoost and Random Forest models decreases the log-loss of the predictions. Meanwhile, increasing the weight of the Multinomial logit model increases the log-loss of the predictions. As a result, we assigned the XGBoost model with the highest weight (as it is the most reliable model), followed by the random forest and the multinomial logit models.

3 Results Analysis

Throughout the competition, we made a total of 15 submission attempts. During the model development phase, we split the `train2024.csv` data to train and test sets and computed the log-loss based on these sub-datasets. We also tracked the public score and, once the competition ended, the private score. The results are displayed in the scatterplots in Figure 1





Submission	Model
1	XGB Mark 1
2	XGB Mark 5
3	XGB Mark 7
4	XGB Mark 1.1
5	Multinomial Logit Mark 3
6	XGB Mark 1.1.2
7	Multinomial Logit (ASC) Mark 1
8	XGB Mark 1.1.4
9	Mixed Logit Mark 4.5.4.3
10	Multinomial One-Hot Logit
11	XGB Mark 1.1.5
12	XGB Mark 1.1.10
13	Ensemble Mark 1
14	Ensemble Mark 2
15	Ensemble Mark 3

Figure 1: Scatterplot of Model Performance

When evaluating our models, we compared the test and train log-loss values and aimed to minimize the gap between them. We initially hypothesised that a smaller gap would indicate better performance on unseen data. Additionally, we observed that the train and test log-loss values directly impact the public and private scores. Extremely low log-loss values might indicate overfitting, as seen with model 3, which scored poorly on both the public and private leaderboards. Conversely, model 5, despite having a small gap between train and test log-loss, did not perform well because its

[1] Obtained from: `https://www.kaggle.com/code/marcinrutecki/voting-classifier-for-better-results#9.-Precision-Recall-Curve-(PR-curve)-for-Voting-Classifier-with-soft-voting`

log-loss values were high from the start. Therefore, we need to find a balance between minimising the log-loss values and reducing the gap between train and test log-loss values. Models 13 to 15, which were developed using the Soft Voting Classifier, exemplify this balance.

4 Interpretability

In addition to using models for prediction, the ability to interpret what a model has learned is receiving an increasing amount of attention [6]. Interpretability in ML and statistical models in general refer to the "extraction of relevant knowledge from a machine-learning model concerning relationships either contained in data or learned by the model" [6].

4.1 Discrete Choice Interpretabiity

Interpretability in discrete choice models involves understanding how different attributes of alternatives and individual characteristics influence the choice probabilities. Choice models are especially interpretable as they are grounded in discrete choice theory. The coefficients in an MNL model represent the impact of predictors on the utility of each alternative. This is especially important in managerial and business insights analysis as companies are able to leverage feature importance to make important decisions, in this case transportation analytics. This is also corroborated by V.K. Mishraa and Natarajan et. al. [4], where they found that technically advanced feature is not always preferred by the consumer over the less advanced one. We interpret the variables with positive coefficients in our MNL model as desirable to customers, while negative coefficients mean that the presence of that variable in the package makes it less desirable. It is thus natural that most safety features have a positive coefficient while price has a negative coefficient.

4.2 Bagging Interpretability

While a single classification tree can be easily intepretable, where the variables used in the earlier splits could be said to be more important, a bagging model such as random forest, that aggregates the predictions of hundreds of trees is not easy to interpret. However, we can utilise the variable importance plot using the `varImp()` function to see which variables contribute most to the model's accuracy and the homogeneity of the nodes in the trees [3]. As seen in Figure 4 in the Appendix, it is clear that the `Price` variables are the most important, followed by the variables that describe the alternative specific effects, such as information on the person making the choice and the car in consideration. This is true for both the accuracy of the model and the homogeneity of the leaves in the model.

4.3 eXplainableAI (XAI)

4.3.1 SHapley Additive exPlanations (SHAP)

SHAP is a popular method to interpret the output of machine learning models such as Extreme Gradient Boosting (XGBoost). It is based on cooperative game theory and assigns an importance value to each feature for a particular prediction. In order to help interpret the relative importance of features in a customers' decision to choose a certain option, we plotted a SHAP graph for each option, as seen in Figure 8 in the Appendix. It is thus evident that regardless of choice, the Price feature of every option is highly important in a user's choice, followed by the features related to the respective choice chosen by the user. In the case of those who selected none of the packages, all features apart from their background and the prices of the other packages were not as important in their decision.

5 Limitations

One important limitation of our methodology was the choice of hyperparameter tuning method, which as mentioned previously, was mainly based on a localised and self-implemented grid search technique. Tuning with this approach was significantly time-consuming, as we had a large parameter space to conduct the search over. Furthermore, Grid Search techniques are also prone to overfitting to the training set even with cross-validation techniques [5], which might cause the model to be less generalised and perform slightly worse on unseen data. An improve that could have been made here, given more time for experimentation, was the use of advanced techniques such as Bayesian optimisation, which uses a surrogate model to approximate the true objective function [8] [11]. The Bayesian optimisation technique for hyperparameter tuning is more efficient compared to a Grid or Random search (especially in higher dimensional spaces), as it uses information from previous evaluations to inform the search for better hyperparameters, and might reduce the risk of overfitting to particular hyperparameter settings. In `R`, there are several packages that could be used here, including `ParBayesianOptimization` and `MlBayesOpt`.

6 Further Research

In the scope of this competition, our team decided not to explore Deep Learning (DL) methods and instead focused on theory-grounded choice models and "simpler" ML algorithms with good Kaggle performance records. However, there have been multiple recent attempts at discrete choice modelling using novel ML and DL techniques which are highly interesting, as we are just entering the age of high-dimensional datasets [10]. Access to a larger dataset allows DL models to learn hidden underlying distributions which are unable to be captured by classical choice models [14] [12]. These methods show promising performance on higher dimensional data and are worth exploring for further research on discrete choice analysis.

References

[1] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16. ACM, August 2016.

[2] Tianqi Chen, Tong He, Michael Benesty, Vadim Khotilovich, Yuan Tang, Hyunsu Cho, Kailong Chen, Rory Mitchell, Ignacio Cano, Tianyi Zhou, Mu Li, Junyuan Xie, Min Lin, Yifeng Geng, Yutian Li, and Jiaming Yuan. xgboost: Extreme gradient boosting, September 2014.

[3] Fernando Martinez-Taboada and Jose Ignacio Redondo. Variable importance plot (mean decrease accuracy and mean decrease Gini). *Variable importance plot (mean decrease accuracy and mean decrease Gini)*, 4 2020.

[4] Vinit Kumar Mishra, Karthik Natarajan, Dhanesh Padmanabhan, Chung-Piaw Teo, and Xiaobo Li. On theoretical and empirical aspects of marginal distribution choice models. *Management Science*, 60(6):1511–1531, June 2014.

[5] Osval Antonio Montesinos López, Abelardo Montesinos López, and Jose Crossa. *Overfitting, Model Tuning, and Evaluation of Prediction Performance*, pages 109–139. Springer International Publishing, Cham, 2022.

[6] W. James Murdoch, Chandan Singh, Karl Kumbier, Reza Abbasi-Asl, and Bin Yu. Definitions, methods, and applications in interpretable machine learning. *Proceedings of the National Academy of Sciences*, 116(44):22071–22080, October 2019.

[7] Karthik Natarajan and Bikramjit Das. The Analytics Edge 2024, 2024. Obtained from `https://www.kaggle.com/competitions/the-analytics-edge-data-competition-2024`.

[8] Vu Nguyen. Bayesian optimization for accelerating hyper-parameter tuning. In *2019 IEEE Second International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*, pages 302–305, 2019.

[9] Bohdan Pavlyshenko. Using stacking approaches for machine learning models. In *2018 IEEE Second International Conference on Data Stream Mining Processing (DSMP)*, pages 255–258, 2018.

[10] Brian Sifringer, Virginie Lurkin, and Alexandre Alahi. Enhancing discrete choice models with representation learning. *Transportation Research Part B: Methodological*, 140:236–261, 2020.

[11] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[12] Sander van Cranenburgh, Shenhao Wang, Akshay Vij, Francisco Pereira, and Joan Walker. Choice modelling in the age of machine learning - discussion paper. *Journal of Choice Modelling*, 42:100340, March 2022.

[13] Anna Vettoruzzo, Mohamed-Rafik Bouguelia, Joaquin Vanschoren, Thorsteinn Rögnvaldsson, and KC Santosh. Advances and challenges in meta-learning: A technical review, 2023.

[14] Shenhao Wang, Baichuan Mo, and Jinhua Zhao. Deep neural networks for choice analysis: Architecture design with alternative-specific utility functions. *Transportation Research Part C: Emerging Technologies*, 112:234–251, 2020.

A Appendix - Code

The code used to run all experiments can be found in this Github Repository, which can be cloned here: `https://github.com/michael-hoon/tae-hackathon.git`.

B Appendix - Figures

Additionally, we found the relative importance of each of the features in the building of each model, and plotted the following:

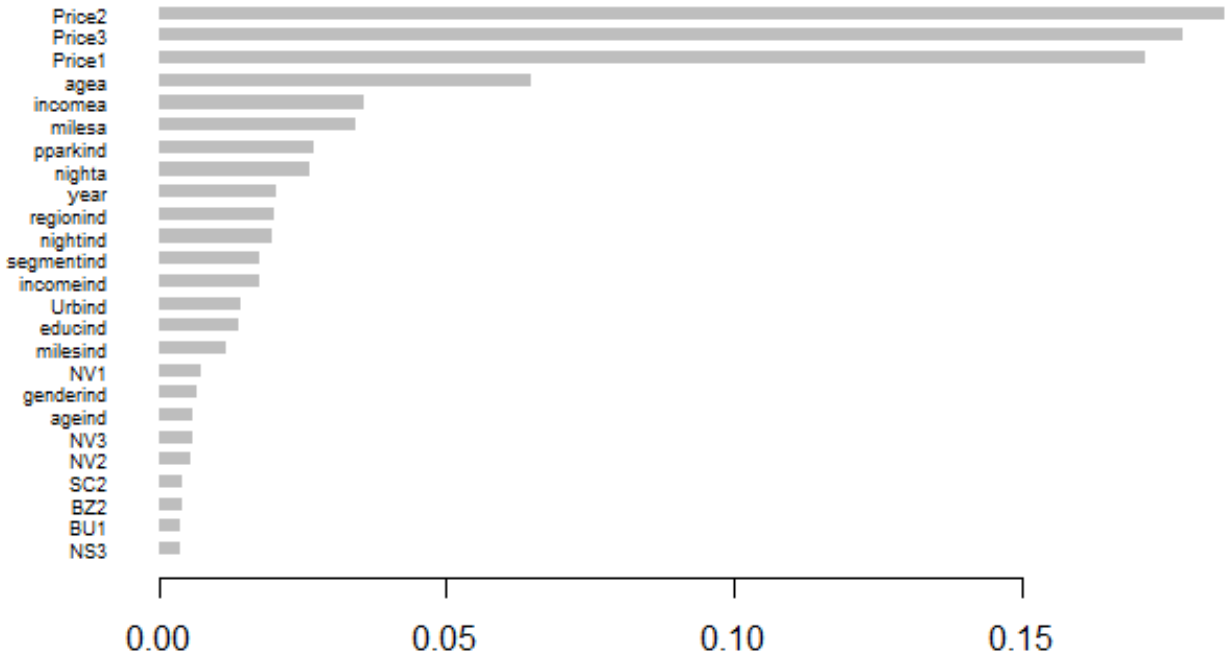


Figure 2: Importance Chart (XGBoost Model)

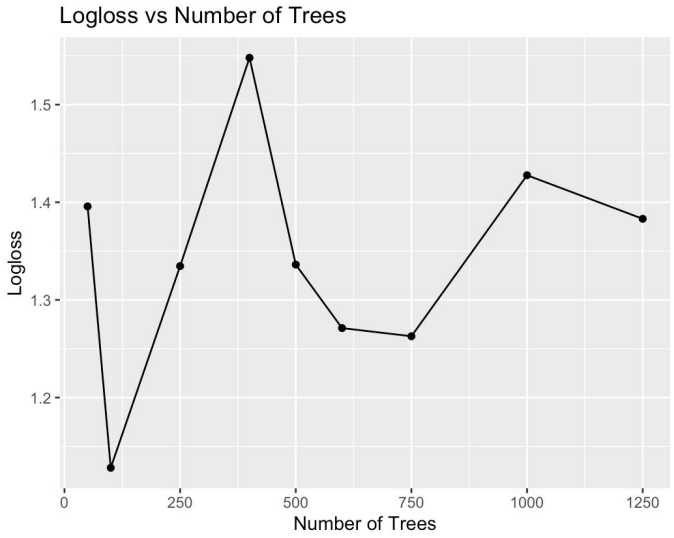


Figure 3: Cross-validated log-loss against ntree (Random Forest model)

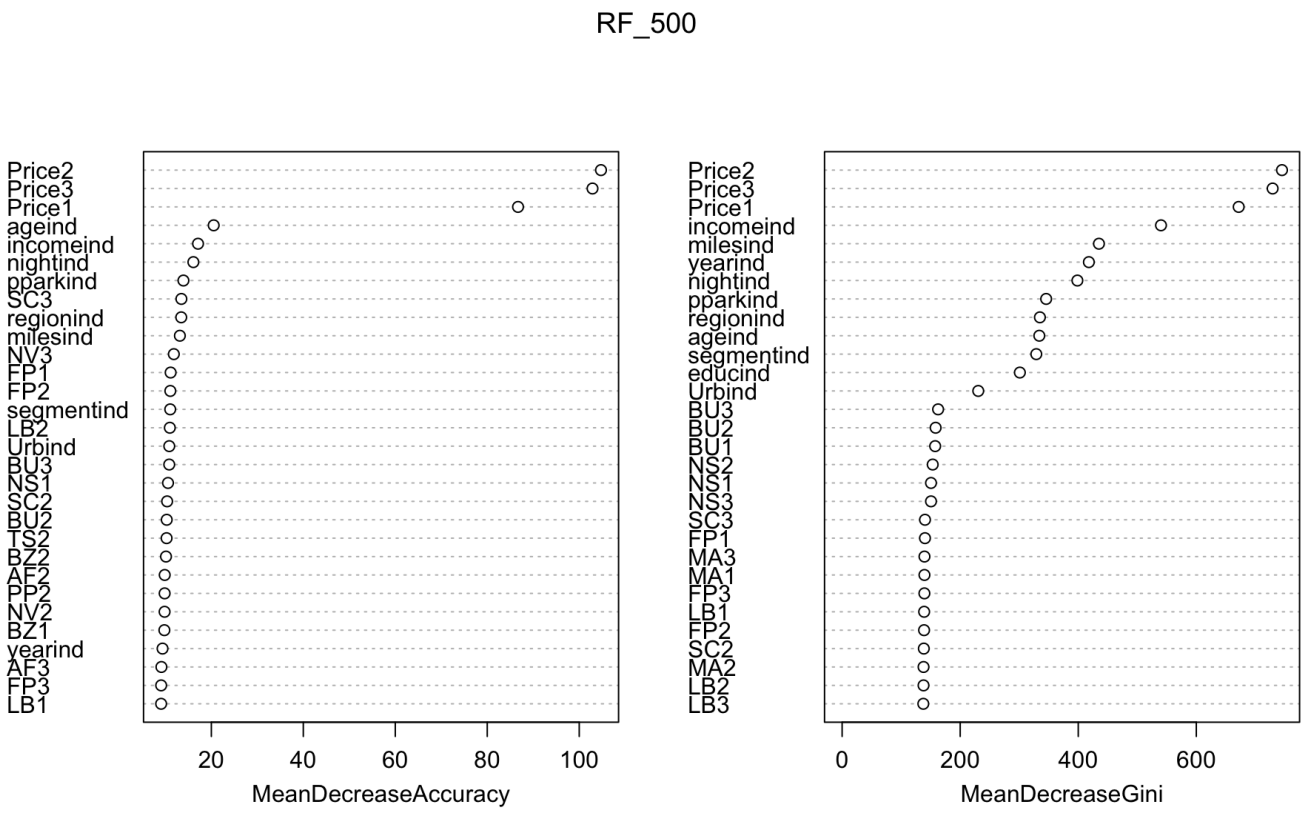
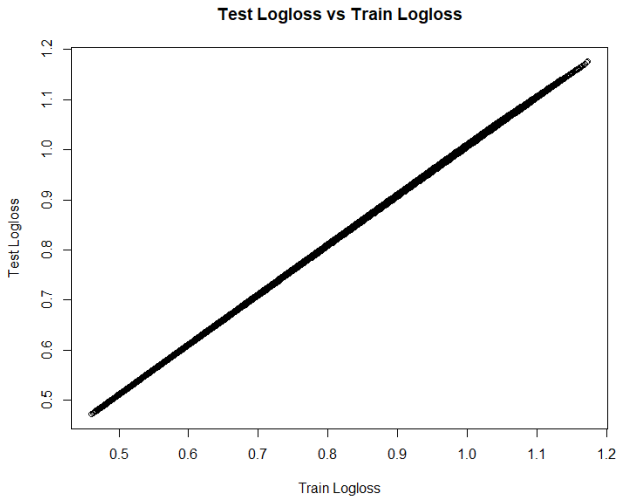
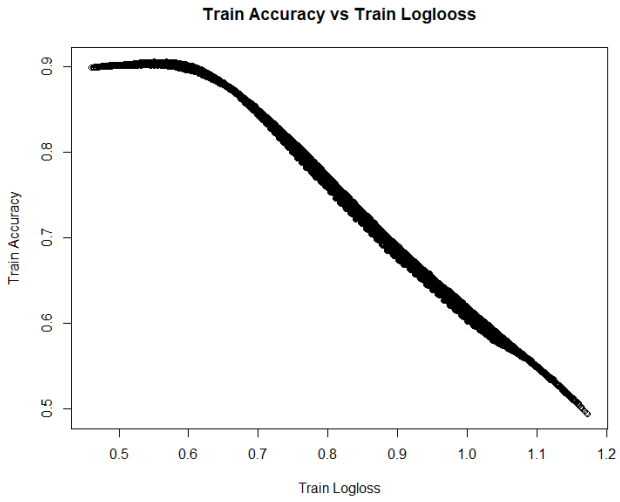


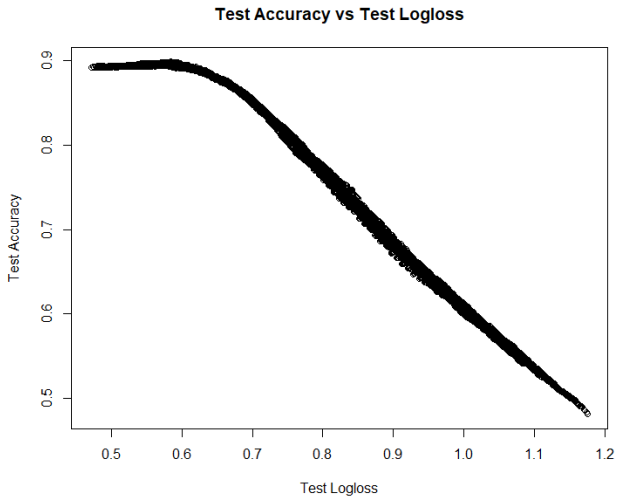
Figure 4: Importance Chart (RF Model)



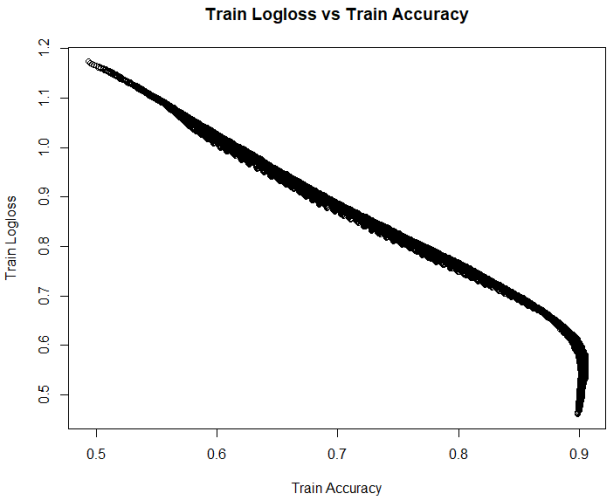
(a) Test Log-Loss vs. Train Log-Loss



(b) Train Accuracy vs. Train Log-Loss

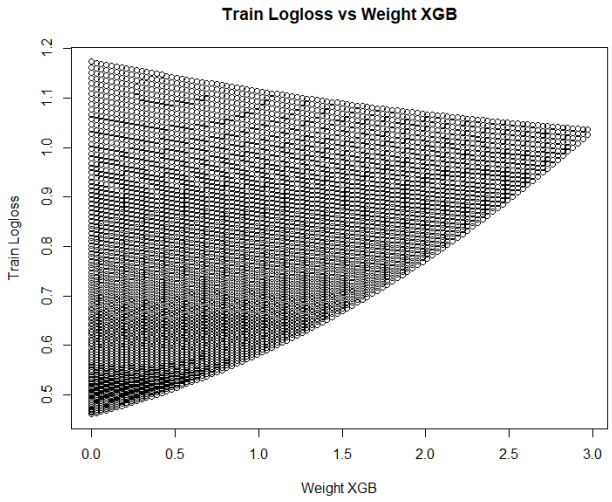


(c) Test Accuracy vs. Test Log-Loss

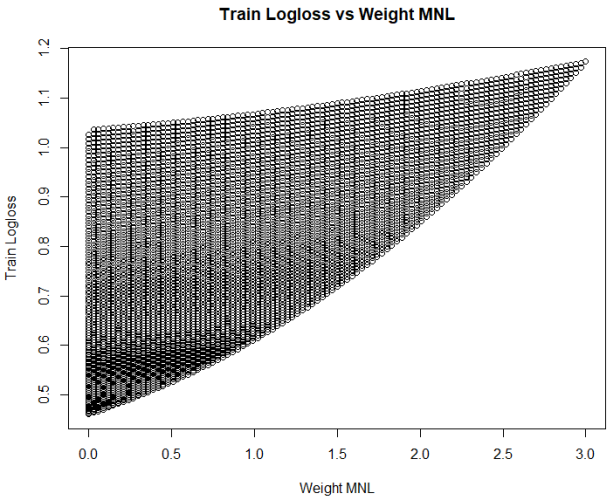


(d) Train Log-Loss vs. Train Accuracy

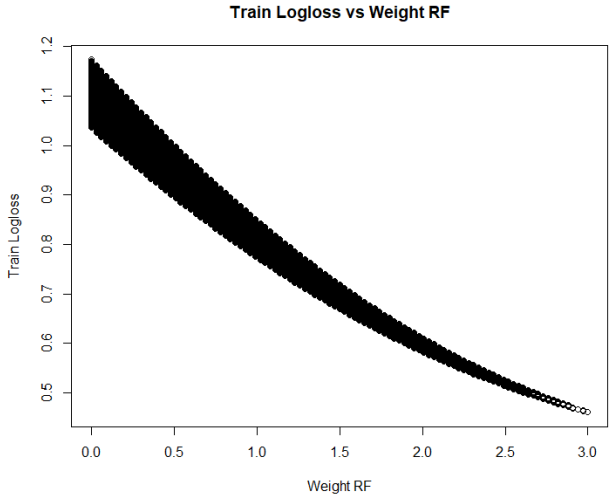
Figure 5: Train vs. Test Plots



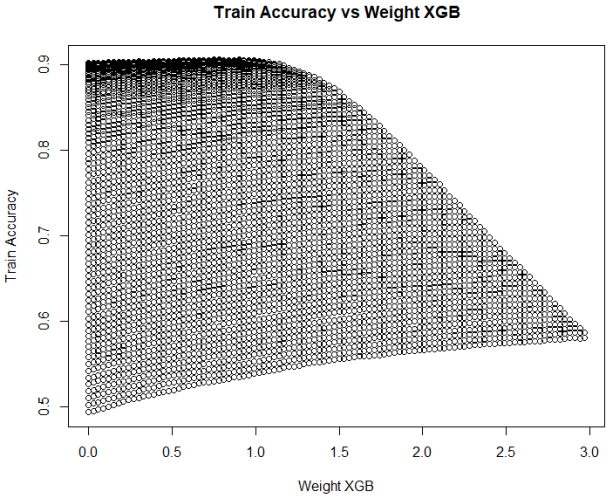
(a) Train Log-Loss vs. Weight XGBoost



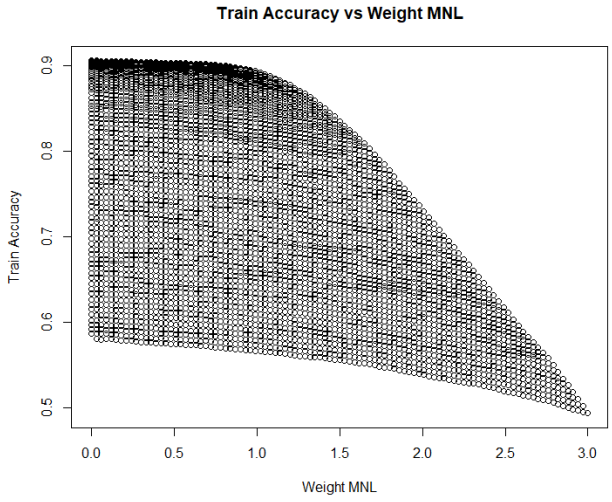
(b) Train Log-Loss vs. Weight MNL



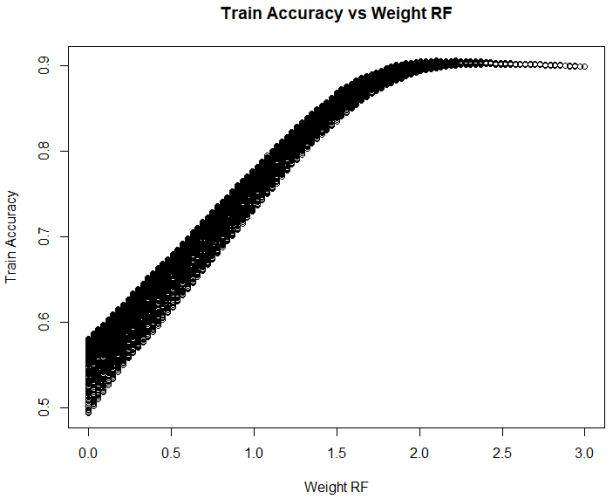
(c) Train Log-Loss vs. Weight RF



(d) Train Accuracy vs. Weight XGBoost



(e) Train Accuracy vs. Weight MNL



(f) Train Accuracy vs. Weight RF

Figure 6: Train vs. Weight All Models

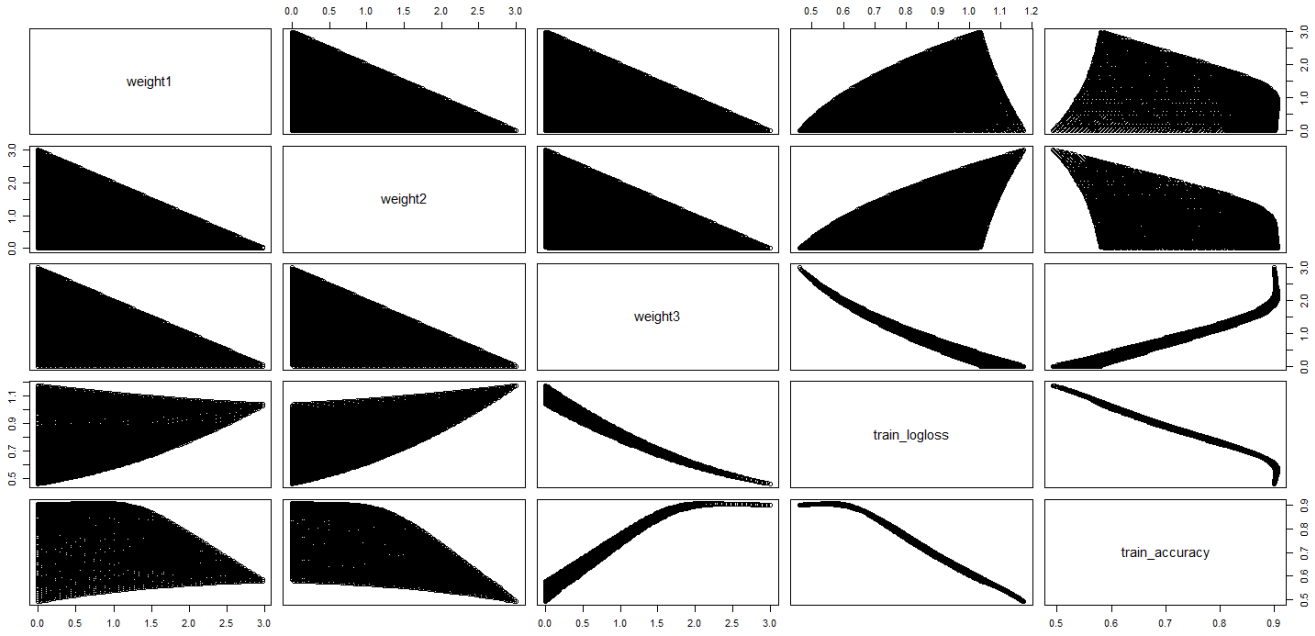
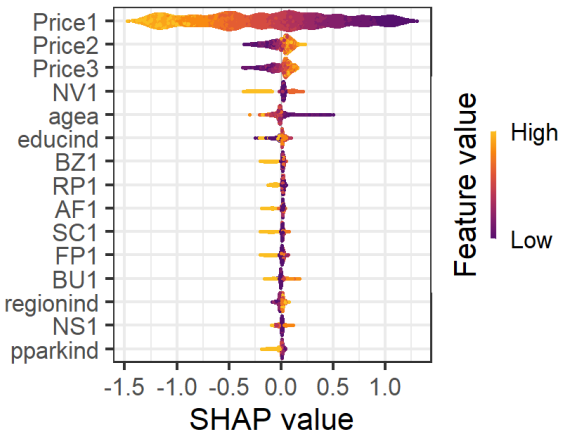
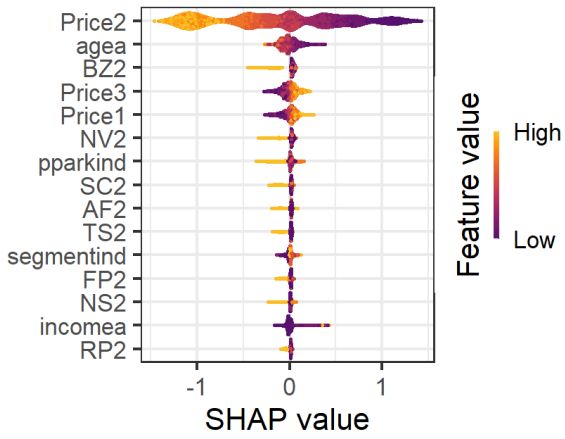


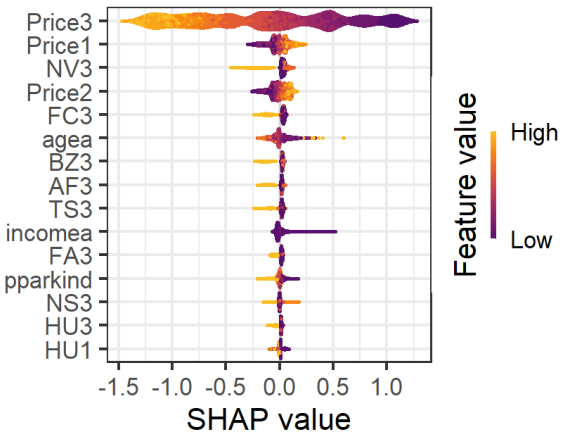
Figure 7: Pairplots



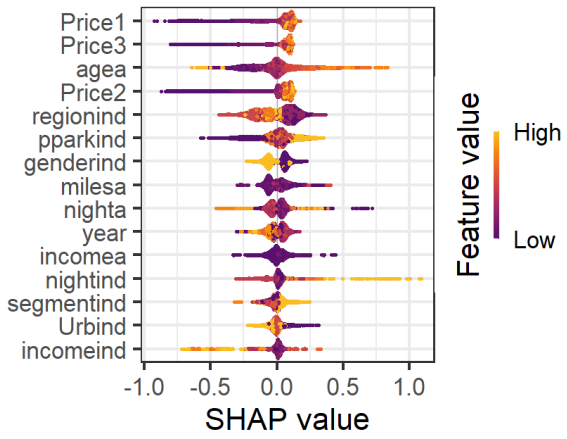
(a) SHAP Importance Plot - Ch1



(b) SHAP Importance Plot - Ch2



(c) SHAP Importance Plot - Ch3



(d) SHAP Importance Plot - Ch4

Figure 8: SHAP Importance Plots Based on Package Choice