Predicting if a car purchased from an auction will be good buy or bad buy

[WEB APPLICATION]

Bhavish Kumar, Sai Praharsha Devalla, Tejas Patil
School of Information Studies, Syracuse University

Dr. Ying Lin
Syracuse University

Abstract

Auto dealers who purchase used cars at auto auctions will be interested in classifying the cars into good cars and bad cars before making their purchase decision. Our project involves the use of several Data Mining techniques to classify used cars in an auction. With a limited knowledge on the factors that determine the car quality, it can be very challenging for a human to make an accurate prediction about the car quality. This is where Data Mining comes into play. Using the Cross Industry Standard Process for Data Mining (CRISP-DM) and building powerful machine learning models will help in accurately predicting if a car will turn out to be a “bad buy” or not.

1. Introduction

Auto dealers are in the business of purchasing used cars at auto auctions and reselling it to their customers. It is very important for auto dealers to ensure that all the cars which they resell to their customers are of good quality. In this process, these auto dealers face the risk of buying a bad quality car with a lot of issues. The auto community refers to these unfortunate purchases as “kicks” and these bad cars are also known as “lemons”. Purchased cars are declared as “lemons” if there are tampered odometers, mechanical issues or other unforeseen problems with the car. The auto dealers will not be able to re sell such cars to their customers. Moreover, these bad cars can be a huge financial burden on the car dealers because of all the transportation, throw-away and repair costs that they must bear with. By predicting if a car is going to be a bad buy or not, we can help the dealers make wise and informed decisions on the cars that they need to purchase which can in turn help them minimize their incurred loss and maximize their profits. Hence the objective of this project is to help the car dealers accurately predict if a purchased car is going to be a “lemon”. We will also try to identify the most important attributes that will help us make this prediction.

In this paper, we examine several data mining/machine learning techniques that will help classify the cars at the auction by taking into account the attributes of the car. A challenge that we encountered in the process was the imbalance in the dataset, with the majority of the cars belonging to not a “lemon” class. This imbalance makes it challenging to train a model that is capable of accurately predicting the minority class and thus beating the null accuracy.

The approach begins with Data Wrangling followed by Data Preprocessing, Exploratory Data Analysis. EDA is followed by building machine learning models capable of performing binary classification followed by model fine tuning and performance evaluation. The process of EDA followed by model building, fine tuning and performance evaluation is an iterative process which can be carried out until the best and desirable results are achieved.

2. Data Description

The dataset we used was downloaded from kaggle provided by Caravana. We have a separate collection for train and test dataset. Based on the number of records, the train and test dataset are divided in the ratio 3:2. The training dataset has a total of 34 assessment parameters, one of which is the target variable. The group of independent variables we have are a mix of categorical and numerical type of data. There are 20 categorical, 11 continuous numerical and 3 discrete numerical variables. The target attribute is a binary variable with 64007 ‘good buys’ and 8976 ‘bad buys’. The large disparity between the counts of 2 classes in the target variable makes our dataset imbalanced. Most of the independent variables in our data describe various specifications of the auctioned car such as ‘model’, ‘color’, ‘Wheel Type’ etc. Remaining attributes inform us about the acquisition and current price of that car during auction and retail sales. This dataset has missing values which are defined as a string value named “NULL”. 18 of those 33 independent attributes contain “NULL” string values which must be treated.

3. Data Wrangling

We observed 2 main issues with our data which are missing values and outliers. To handle missing values, we made use of the measures of central tendency. Missing values in Categorical variables, were replaced with the Mode of the variable, where mode is the most frequently occurring value. Whereas, for numeric variables, the missing values were replaced with the mean value of that variable for variables whose distributions are not skewed, and with the median otherwise.

Imputing with mean/median/mode may not be the most accurate imputation and may induce bias in the data and also in the estimates produced by the model. But the main advantage of this method of NA imputation over other methods such as KNN[1] imputation or MICE[2] imputation is that this method is computationally very inexpensive and works very fast in comparison to KNN or MICE imputation, especially when dealing with large datasets. Because of this reason, we decided to go ahead with the mean/median/mode method of imputation. Next, we used the Z score approach to identify outliers in numeric variables and treat them, as algorithms such as K Means clustering and K Nearest Neighbors are sensitive to outliers. The values which are more than 3 standard deviations away from the mean were identified as outliers and such variables were winsorized, by limiting all values to fall within 3 standard deviations away from the mean.

4. Exploratory Data Analysis

Exploratory Data Analysis is performed in order to identify trends and patterns in the data and to identify potential effects that each variable can have on the target variable. The main reasons for performing Exploratory Data Analysis are: [1]

- detection of mistakes
- checking of assumptions
- preliminary selection of appropriate models
- determining relationships among the explanatory variables, and
- assessing the direction and rough size of relationships between explanatory and outcome variables.

One very insightful bi-variate analysis bar graph that we obtained is shown below. This bar graph shows the % of Good Buys and Bad buys for each of the vehicle age groups. Through this bar graph we were able to validate our assumption. It is clearly evident that the proportion of bad buys increases with vehicle age and out of all the purchases made with 9-year-old vehicles, 31% of those purchases were bad buys.

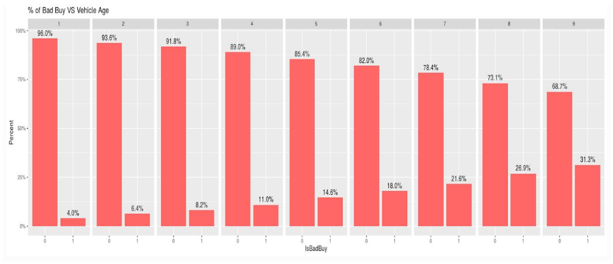

Fig. 1 Vehicle Age versus percent of bad buys

[1] K Nearest Neighbours imputation, finds K nearest neighbours for a missing datapoint from the entire dataset and fills the missing datapoint with the most frequently occuring value amongst the K nearest neighbours

[2] Multiple Imputation by Chained Equations

The remaining Bi Variate Analysis visualizations are shown on our application, which contains bi variate analysis on other vehicle attributes and auction related attributes. Only the visualizations that show some pattern or trend were chosen to be displayed on the web app.

5. Data Preprocessing

Exploring the data showed us that few of the categorical variables contained too many categories with low frequency count. Now, each one of these categories will create a new variable during one hot encoding and increase the data dimensionality. Therefore, for the variables 'Model', 'SubModel', 'Trim', VNST' and 'Make', we decided to group the categories having frequency count less than 500 into a single category called 'Other'.

To identify the redundant features in our data, we performed two feature selection tests. First, we carried out a chi-square test of independence for the categorical variables, to determine the importance of those variables with respect to the target variable [5]. We observed that the 'color' variable and 'TopThreeAmericanName' variable had a p-value of 0.29 and 0.336 respectively. Since, their p-value is more than 0.05, we fail to reject the null hypothesis[3]. All other variables had significant p-value.

The second technique that we used was to run a simple model of logistic regression over the car data set. From the model's summary, we concluded that all the dummy variables generated for the categories in "Model" and "SubModel" variables are not significant as they have a p-value greater than 0.05. So, we dropped the variables 'Color', 'Model', 'SubModel' and 'TopThreeAmericanName'.

Using one hot encoding, we created dummies for all the categorical variables in our data and performed a train-test split. As we might oversample the data for handling the imbalance, this split will help prevent leaking of information to the test data. The partition was done with respect to the target variable so that the distribution of the data remains constant.

The target variable which represents if a car is a good buy or a bad buy, is highly imbalanced. This imbalance might result in our models believing that all of the cars are good buys. Hence, we decided to perform hybrid sampling over the data[6]. It combines oversampling of minority class with undersampling of majority class using SMOTE (Synthetic Minority Oversampling Technique) algorithm[3]. SMOTE uses K-Nearest neighbor method to generate the synthetic observations when oversampling. After hybrid sampling, the dataset had 35000 observations with a balanced target variable. As expected, when we built the model, we got excellent results with accuracy more than 85% and recall more than 65%. Unfortunately, the results did not replicate on the test data which was a clear indication of overfitting. Our next approach was to just undersample the data for the majority class. A lot of information was lost in this process, but this technique gave us the smallest generalization error.

As we intended to use distance-based algorithms such as K-Nearest Neighbors, we used Standard scaler to scale all the numerical variables to the same range so that all variables could contribute to the prediction. Pre-processing of data for classification algorithms finishes here, the remaining processing steps are for unsupervised learning and association rules mining.

After creating dummies, our data had too many variables. Hence, to reduce the number of dimensions without losing much of the information, we performed principal component analysis which can be pivotal when performing unsupervised learning. The original dataset (before splitting) in its entirety was used for PCA. All the principal components are perpendicular to each other,

[3]**Null Hypothesis:** No relationship exists between the categorical variables in the population; i.e. they are independent
Alternate Hypothesis: There exists a relationship between the categorical variable and they are dependent

so the issue of multicollinearity does not exist in them [7].

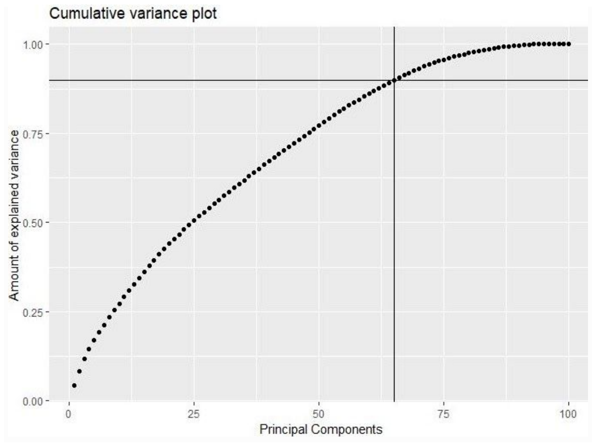

Fig. 2 Principal Components cumulative variance plot

This graph here tells us how much variance each principal component explains cumulatively. We can see that as the number of principal components increases on the X-axis the cumulative explained variance also goes on increasing but the rate of increase decreases with each principal component. Out of all the components, the first 65 principal components explain 90% of the variance in our data. Hence, we decide to select the first 65 principal components for clustering. We normalize all the principal components as K-means clustering is a distance based algorithm.

Finally, for association rules mining, again we used the original data before splitting and sampling. Association rules are mined only for categorical variables, hence we grouped all the numerical variables into categories. We observed in exploratory data analysis that all the numeric variables follow either a normal distribution or flat distribution. So, we made a function that will categorize a numerical variable by assigning every observation to their respective quantile, where each quantile will be a category. Lastly, we converted that data frame to a transaction matrix as apriori algorithm does not work with data frame.

6. Machine Learning Methods

The purpose of this research is to be able to determine whether a car in an auction turns out to be a good buy or a bad buy by evaluating the features of that car and the auction beforehand. We developed 4 supervised learning classification models which include Logistic Regression, K-Nearest Neighbor, Random Forest and Gradient Boosting to be able to identify a new example of the test as a good or poor buy. A K-means clustering unsupervised method of learning was developed to determine how many groups our data naturally contains. Finally, we mined the rules of association between independent variables and the two groups of the dependent variable using apriori algorithm.

6.1. Logistic Regression

Logistic regression is one of the most used algorithms when it comes to predicting the binomial outcomes. Logistic regression groups the output of the linear regression into categories. The sigmoid function at the end of linear regression modifies the continuous output to probability between 0 and 1. A threshold is set of any value between 0 and 1 (generally 0.5), to classify the points as either yes or no.

$$y = \left(1 + e^{-(\beta_0+\beta_1 X_1+\beta_2 X_2+\ldots+\beta_n X_n)}\right)^{-1} \qquad \ldots(1)$$

Where, y represents the output of sigmoid function, X is the dependent variable and β is the coefficient generated for that variable by regression.

For logistic regression base model, the training accuracy was calculated to be 68% and the testing data accuracy was calculated as 70% with a recall of 64%. We have to keep in mind that our training dataset is balanced while test data is not. The no information rate for test data is 88%.

We decided to regularize the model to shrink the coefficients of variables, which in turn reduces the variance of the model. These measures are taken to reduce overfitting. Regularization adds an extra penalty term to the cost function.

$$\sum_{i=1}^{n} (y_i - \widehat{y}_i)^2 + (1-\alpha)\lambda \sum_{j=1}^{m} \beta_j^2 + (\alpha)\ \lambda \sum_{j=1}^{m} \beta_j \quad ...(2)$$

The first term in the equation is cost function for regression, that is sum of squared errors, where y_i are the actual values and $\widehat{y}_i$ are the predicted regression values. Second term is ridge regularization that adds a penalty term equal to sum of squared value of coefficients times λ, where λ is the hyperparameter that controls the intensity of regularization and α is the elastic net regularization parameter. Third term is lasso regularization, that adds penalty term equal to sum of absolute value of coefficients times λ

While finding the best value for λ and α, we observed a trade off between accuracy and recall. For smaller values of α i.e. when ridge regularization was dominant, the accuracy for test data was 70%, and recall was 65%. For higher values of α and λ i.e. when lasso regularization was dominant, the accuracy was 88% but recall went down to 23%. This trade off was observed because of the imbalance in data. Different distributions of target variables for train and test data are affecting the predictions of the model.

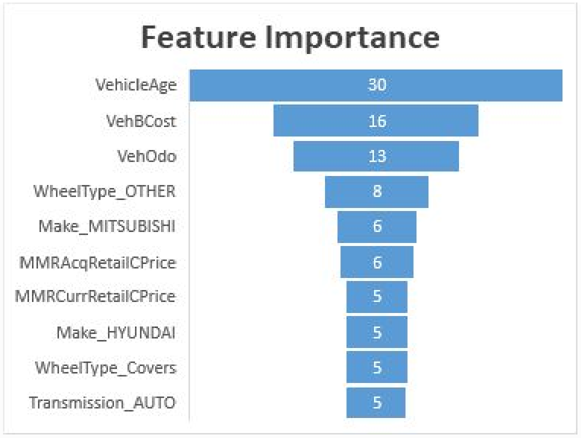

Fig. 3 Variable importance calculated using Logistic regression

6.2. K-Nearest Neighbors (KNN)

This is a distance based lazy evaluation algorithm that considers the class of the K nearest training data neighbors of a test data point to classify every test datapoint [4].

$$d_{minkowski} = \left(\sum_{i=1}^{n} \left| x_i - y_i \right|^p \right)^{1/p} \quad ...(3)$$

n : *total number of samples*

The Eq(3) is the generalized form of distance function[2]. If $p=1$,it is considered as Manhattan Distance and $p=2$ Euclidean distance. By default the KNN uses the Euclidean function to calculate the distance. The number of neighbors to be considered K is a hyperparameter that can be tuned. The k-nearest neighbors of a given example test sample refer to the k points that are closest to the test sample. Here we will find all the training samples that are exact or similar to the test sample. When K-neighbors is specified, it considers the K number of samples from training that are closest to the testing sample. Now the closest training samples majority class will be given to the testing sample. If the K is too small then variance will be very high and if the k value is too large then the farthest training point will have the same effect as the nearest point. When using the large number of neighbors we can use weights which nullify the effect of the farthest sample.

$$Weight = (1/dist(train\ point, test\ point))$$

For this issue not to occur it is better to use optimum K which is neither too large nor too small. The KNN model doesn't have loss function, it just memorises the whole training data. The car auction data is an imbalanced dataset with 80% as ‘good buy’(majority class). Due to this the *KNN* performance deteriorates for imbalance data as the model considers everything as a good buy failing to classify the ‘bad buy’ in the data. The accuracy of the model is 87% but the recall is quite low 43% for the best model with neighbors as 21. The recall is quite low because the majority class present in the data is the ‘good buy’.

6.3. Random Forest

Random Forest classifier which is an ensemble method consisting of multiple decision trees which combines the predictions made by multiple decision trees and each node split is generated using a selected number of input features [4]. The decision trees with bagging, forms a special case of random forest which gives the randomness to the model compared to bagging by randomly selecting the samples to build the individual tree with replacement. The primary unit of random forest is decision trees, it uses the splitting criteria as Gini Index or Entropy.

$$Entropy(t) = -\sum_{i=0}^{c-1} p(i|t)\, log_2 p(i|t) \qquad ...(4)$$

$$Gini\ Index(t) = 1 - \sum_{i=0}^{c-1} [p(i|t)]^2 \qquad ...(5)$$

$$\Delta = I(Parent) - \sum_{j=1}^{l} \frac{N(v_j)}{N} I(v_j) \qquad ...(6)$$

$$Loss\ function = -\frac{1}{N}\sum_{i=1}^{N} y_i.log(p(y_i)) + (1-y_i).log(1-p(y_i)) \qquad ...(7)$$

c in (4),(5) is the number of classes present in data [4]. $p(i|t)$ in (4),(5) denote the fraction of records belonging to class i at a given node t [4]. $I(.)$ in (6) represents impurity of a node, N in (6) represents total records in the node before splitting, j in (6) represents the total number of attributes, $N(v_j)$ represents all the records associated with node v_j. Δ(6) represent the gain, if the impurity used is entropy it can be called as Information Gain [4]. y_i in (7) is the class label. $p(y)$ in (7) represents the predicted probability of the point being class 1 for all N points. The random forest uses the (7) as the loss function. The decision tree works in a way that entropy is maximum at the root node and will eventually reduce to zero at the leaf node. The node splitting will be based on a feature which has the highest Information Gain(6). Information Gain can be defined as difference in Entropy (4) before splitting and the weighted average of entropy of child nodes.

To reduce correlation between trees, the random forest selects P input features randomly. As a result, instead of using all the available features, the decision tree will split a node based on these P features. In a random forest the tree can grow to the fullest without any pruning. This may help reduce the bias present in the resulting tree. Once the trees have been constructed, the predictions are combined using a majority voting scheme. Ideal number of features that can be considered while splitting a node for random forest can either be $log_2(attributes)$ or $(attributes)^{1/2}$. The number of features that can be considered to get the best node split for each decision tree, depth of the tree and the number of trees are the hyperparameters that can be tuned. The trees parameter specifies the number of trees in the random forest model. Lesser number of trees might result in overfitting due to high variance. So, a higher number of trees ensures the random forest classifier does not overfit the data. max_depth is the number of levels in the tree from the root node to leaf nodes. If the tree goes deep it might reduce bias but increases the variance leading to over-fitting. $Number\ of\ features$ is to inform each tree , the number of features to consider when looking for the best split to make. This will not allow the tree to fit the data closely.

To ensure the model is not over-fitting (i.e high variance and low bias) number of trees should be more and $number\ of\ features$ should be the optimal value as mentioned in the above discussion. There should be proper trade-off between variance and bias to build a low variance and low bias model. So we had to use 800 trees and 12 features to get the best results of accuracy 83% and recall 55%. A proper trade off had been maintained between recall and accuracy as $number\ of\ features$ are increased the recall would go up but the accuracy would go down.

6.4. Gradient Boosting Machine (GBM)

Gradient Boosting Machine is an ensemble method consisting of multiple decision trees where each decision tree is built sequentially one after the other. The final model is produced by taking the weighted average of predictions made by each base classifier [4]. The GBM also uses decision trees, it uses the splitting criteria as Gini Index(5) or Entropy(6). Each tree is grown using information from previously grown trees. Boosting does not involve bootstrap sampling, instead each tree is fit on a modified version of the original data set. GBM works iteratively as mentioned above to reduce the loss function(7).

Unlike fitting a single large decision tree to the data, which amounts to fitting the data hard and potentially overfitting, the boosting approach instead learns slowly. Given the current model, a decision. We then add this new decision tree into the fitted function in order to update the residuals. These trees can be small with only a few leaf nodes. After every iteration the residuals are updated for the new tree to build on. By updating the residuals after every iteration the log loss (7) will be reduced. Unlike bagging, construction of every tree depends on the previously built trees.

The hyperparameters tuned for GBM are the depth of each tree, the number of trees, the learning rate and the minimum number of data points needed in each node for splitting. The *number of trees* specifies how many trees must be included in the model. Lesser the *number of trees* high chances of overfitting due to high variance. So a higher *number of trees* makes sure that the gradient boosting classifier doesn't overfit the data. *Number of features* is to inform each tree, the number of features to consider when looking for the best split to make. *learning_rate* A technique to slow down the learning in the gradient boosting model is to apply a weighting factor for the corrections by new trees when added to the model. Higher values of *learning_rate* for the models tend to overfit (high variance).

To control overfitting (high variance and low bias) increase the *number of trees* in the model, increase the *number of features* required in a node and low *learning_rate* for the model. After fine tuning hyperparameters, we had to use 1400 *trees*, 0.01 *learning_rate*, 10 *min_sam_leaf* and 12 *number of features* to get the results for the best model of accuracy 82% and recall 51%. A proper trade off had been maintained between recall and accuracy as *number of features* are increased the recall would go up but the accuracy would go down.

6.5. K-means Clustering

K-means clustering is a partitional type of clustering algorithm i.e. the number of clusters are needed to be defined prior to running the model. To know the optimum number of clusters beforehand, the elbow curve needs to be plotted with sum of squared error on the Y-axis and number of clusters on X-axis. The sum of squared errors is calculated by taking the sum of the euclidean distance (3) between centroid and each data point in that cluster. This process is repeated for each cluster. A sharp bend in the curve signifies a drastic decrease in reduction of sum of squared error. That point is considered as the optimum number of clusters. For our data, the optimum number for clusters is 4. Centroids for every cluster are assigned randomly at the beginning and based on distance from all points, gets updated after every iteration. We specify maximum iterations for our model to be 100 which should be sufficient for the centroid points to attain their best position. The following graph shows the 2D representation of data points grouped into 4 clusters.

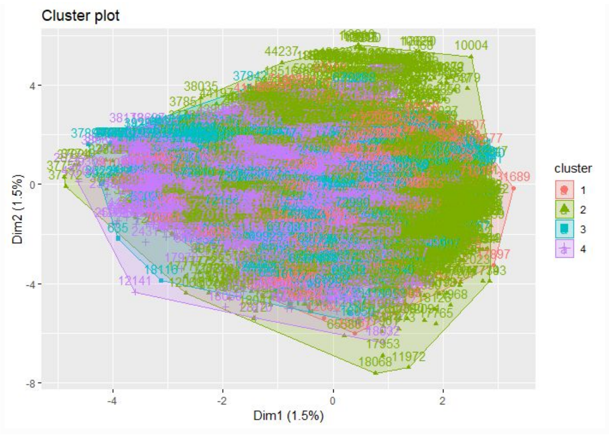

Fig. 4 2D representation of clusters

6.6. Association Rules Mining

For every target variable class (RHS), the apriori algorithm generates rules (LHS). The model is tuned mainly by two hyperparameters, support and confidence. Support is the ratio of number of times LHS and RHS occurred together, divided by the total number of observations. Confidence is the number of times LHS and RHS occurred together, divided by the number of times that only LHS occurred. Both of these values do not take into account the proportion of RHS over the total number of observations. Therefore, to evaluate the rules, we need a parameter that accounts for support, confidence, and proportion of RHS values, that parameter is 'lift'. Lift is characterized as confidence divided by support of RHS i.e. it considers RHS interaction with rules other than LHS as well. If the values for both lift and confidence are strong for a rule, then we find these rules to be the most important rules in a technical sense.

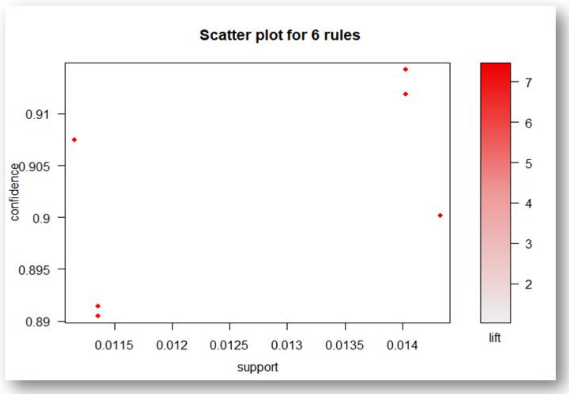

Fig. 5 Scatter plot of best rules extracted bad purchases

This scatter plot is plotted after fine tuning the model hence, it only shows rules with high values for support and confidence scales.

7. Evaluation

Once a predictive model has been developed using historical data, the next step is to evaluate the performance of the model by making predictions on a new validation dataset which the model has not seen before. This evaluation is to ensure that the model has low variance and bias. For classification problems, the confusion matrix is the main data source for performance evaluation. For a 2 class classification problem, the Confusion Matrix will have a 2x2 dimension. The 4 cells in the confusion matrix measure the True Positive, True Negative, False Positive and False Negative values which are the base values behind every model evaluation metric. True Positives are the number of occurrences where the actual value and predicted value are both 'yes' (car is lemon), whereas True Negatives are the number of occurrences where the actual value and predicted value are both 'no' (car is not a lemon). The True Positives (TP) and True Negatives (TN) have a positive impact on the model evaluation metrics. False Positives (FP) are the number of occurrences where the actual value is 'no' (car is not a lemon) but predicted value is 'yes' (car is a lemon), whereas False Negatives (FN) are the number of occurrences where the actual value is 'yes' (car is a lemon) and the predicted value is 'no' (car is not a lemon). False Positives and False Negatives have a negative impact on the model evaluation metrics. The 5 most important model evaluation metrics that are calculated from the confusion matrix are Accuracy (8), Precision (9), Recall (10) and F1 score (11).

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad ...(8)$$

$$Precision = \frac{TP}{TP+FP} \quad ...(9)$$

$$Recall = \frac{TP}{TP+FN} \quad ...(10)$$

$$F_1 = \frac{2*PRECISION*RECALL}{PRECISION+RECALL} \quad ...(11)$$

Since our dataset is imbalanced, Accuracy alone cannot be a reliable metric for us to evaluate our model performance and other metrics such as Precision, Recall and F1 score have to be taken into account to understand our model performance. Recall is certainly an important metric for evaluating our model performance because, by focusing on maximizing Recall, we will be reducing False Negatives (Type II error) which are highly undesirable when making predictions on car quality.

7.1. Results

Let us compare the performance of each of our supervised learning models. From Fig. 6, we can observe that the Random Forest model is the best performer with a Recall of

55% and with an Accuracy of 83%. This is followed by Gradient Boosting Machine, which has a Recall of 51% and Accuracy of 82%. The K Nearest Neighbours model has a very high Accuracy of 87%, but the model underperforms when we take Recall into account, because of which KNN cannot be considered as the best performing model.

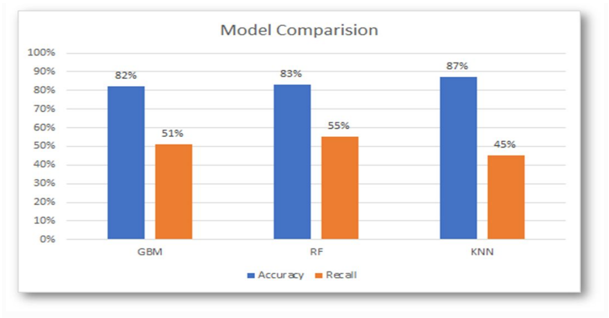

Fig. 6 Model Performance Comparison

The other evaluation metrics such as Precision, F1 score and AUC for each of these three models are shown in the below table.

Algorithm	**Accuracy (%)**	**Precision (%)**	**Recall (%)**	**F1 score (%)**	**AUC (%)**
RF	83	47	55	54	74
GBM	81	35	54	42	73
KNN	87	43	47	48	71
LR	70	23	64	34	75

Let us look at the result we obtained from our Association Rules Mining.

The Fig.7 is a paracord plot, which plots the rules generated for 'Bad Buy'. The rules were sorted on the basis of lift and then top 5 rules were plotted. From this graph, we can observe that, if a car is not purchased online and has wheel type as 'other,' transmission as 'auto' and is purchased from ADESA's auction, there is a high probability of that vehicle being a bad purchase.

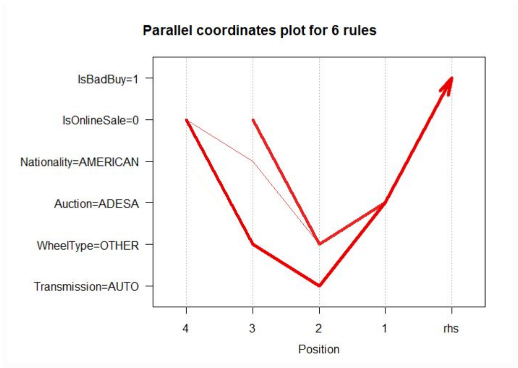

Fig. 7 Paracord plot Association Rules

8. Discussion

One of the biggest challenges that we faced during our model building and evaluation phase was to surpass the null accuracy of our imbalanced dataset without having to compromise on the Recall scores. Because of the imbalanced nature of our data, a null model would have produced an Accuracy of ~88% and our goal was to try and attain an accuracy of > 88%. But during our model building phase, we observed a trade off between Accuracy and Recall, where an increase in Accuracy was resulting in a lower Recall. Hence it was of paramount importance for us to maintain the right balance between Accuracy and Recall, such that neither of them are lower than the desirable value. Maintaining the right balance was also quite challenging and it involved running multiple iterations until we obtained satisfactory results.

Thus, we believe that an important next step for this project would be to explore more advanced algorithms or techniques which deal with this class imbalance problem and yield both a high accuracy and a high recall score. We look forward to handling this challenge in the future.

Acknowledgements

We would like to thank our professor Dr Ying Lin for being an excellent instructor who helped us gain all the knowledge that is required to complete this project. He also

guided us through the entire project by providing complete instructions on the ways to approach this project.

References

[1]. Howard J. Seltman *"Experimental Design andAnalysis"*. [Online].Available:https://www.stat.cmu.edu/~hseltman/309/Book/chapter4.pdf

[2]. Merigó JM, Gil-Lafuente AM (2008) "*Using the OWA operator in the Minkowski distance."* Int J Electr Comput Eng 3: 149–157.

[3].[Online]https://towardsdatascience.com/imbalanced-data-in-classification-general-solution-case-study-169f2e18b017

[4]. Tan PN, Steinbach M, Karpatne A, Kumar V (2013) Introduction to data mining, 2nd edition. Pearson, London.

[5]. S. Wu and P.A. Flach (2002). *Feature selection with labelled and unlabelled data*. In Proceedings of ECML/PKDD'02 Workshop on Integration and Collaboration Aspects of Data Mining, Decision Support and Meta-Learning, pages 156–167.

[6]. Blagus, R., Lusa, L. (2013). *SMOTE for high-dimensional class-imbalanced data*. BMC Bioinformatics 14, 106 (2013). [Online]. https://doi.org/10.1186/1471-2105-14-106

[7]. Sugiarto, Teguh (2017). *Application of Principal Component Analysis (PCA) to Reduce Multicollinearity Exchange Rate Currency of Some Countries in Asia Period 2004-2014*, Volume 3, Journal International Journal of Educational Methodology.