Don't get kicked

AIM:

To predict if the car purchased at the Auction is a Kick (bad buy).

Data Description:

The training set had 72983 examples with 34 features with a good mix of numerical and categorical features. After going through the data dictionary and doing a cursory analysis of the dataframe, I realized that there are some redundant and irrelevant features. Here are some of my observations about the data.

1. RefId is just the serial number of the rows.
2. Purchase date and vehicle Year can be used together to infer Vehicle age.
3. WheelType and WheelTypeID are one and the same.
4. VNZIP1 is a more fine-grained feature than VNST.
5. PRIMEUNIT and AUCGUART have most of the values missing (approx. 98%) so imputation cannot be used on these features even though they seem very important.
6. After doing some EDA and correlation analysis, I found that all the MMR price features are highly correlated. But it seemed intuitive to see how they compare against each other.

Methodology:

1. Handling categorical features

   I dropped most of the redundant columns described in the previous section and did some feature engineering using MMR price differences to create new features. Handling categorical features was a big challenge in this project. Some fields such as Make, Model, Trim, Size had some important information about type of engine etc. I used Label encoder from sklearn package to encode them. But I had to apply one hot encoding on top of it because output of Label encoder can be misinterpreted by Machine Learning algorithms from sklearn package as they have ordinal values e.g. 1, 2, 3 for a column with 3 unique values. I also had to handle categorical feature values which were not present in the training set so that encoders don't spit errors.

2. Handling missing values:

   A lot of the columns had missing values. For numerical features I imputed them using the median values and for categorical features I used the most frequent value from that column.

3. Normalizing the feature space:

   I used Standard Scalar module to normalize the feature space to make it more suitable for my classifier.

4. Class Imbalance:

   The dataset was highly skewed. Just by predicting 0 for everything, one could get approx. 88% accuracy. I didn't handle the imbalance in the current solution.

5. Random forest classifier with a train test split 0f 0.33

Evaluation:

I got an accuracy of approx. 90% with Precision of 65%. I also tried to submit my solution on Kaggle and got a score of approx. 0.21

Future Work:

Because of the limited time, I couldn't try a lot of things. I would suggest following additional steps to train a better model.

1. Oversampling of the minority class using SMOTE or random sampling with replacement.
2. Grid Search over the model parameters.
3. Stratified Cross validation
4. Feature engineering on features like odometer reading and vehicle age, total acquisition cost vs MMR features.
5. Feature selection based on relative importance based on gini measures.
6. Trying more sophisticated algorithms such as extreme gradient boosting and other ensembling techniques.
7. Applying better evaluation criteria such as Precision, recall, F-1, AUC of ROC etc.

Please find attached the jupyter notebook with the code and plots.