**Objective**

Our objective is to build a model to detect defects in cars that make them unsellable.
From analyzing the data, we should be able to find and create good features that can help us to make an accurate model.

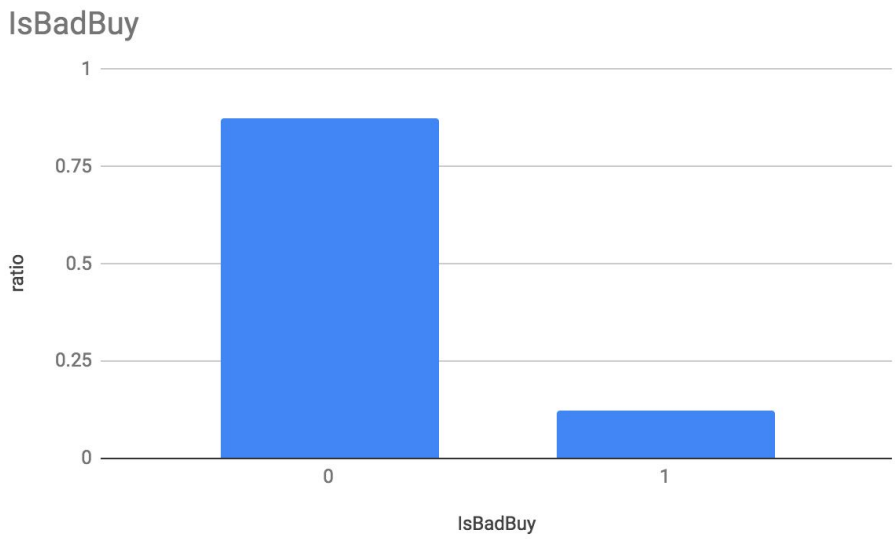
**Method**

I have decided to build a Random Forest model because it has the ability to handle a large data set with high dimensionality by providing high accuracy.

**Feature Data Analysis**

I analyzed each feature by converting and removing some features and imputing some missing values to make a better model.

- Target feature (IsBadBuy): When we look at the target variable, we can find they are greatly imbalanced ( 88% are 0, 12% are 1). So we may need an under/over-sample to build a better model




- Categorical Feature: You can find my all analysis for categorical features in the Categorical Feature analysis section in Jupyter Notebook. In this report, I will briefly explain how I handled categorical features.

    1. Removed features: I removed some unnecessary features such as RefID, PurchDate, and WheelTypeID because RefID is unique by each vehicle, I could not find any significant information from PurchDate and we also have the WheelType feature so we don't need WheelTypeID.

    2. Make, SubModel, VNST: Since these features have too many categories, instead of creating dummy variables, I counted the frequency against each category and

sorted them in descending order. Then I retained categories that went up to 95 percentile in the cumulative. Lastly, I listed all remaining categories as others.

3. Model, Trim: Since these features have too many categories and are unlike MAKE, SubModel, and VNST, there are still more than 100 categories in its top 95% categories in cumulative. So I converted each category to a probability of isBaybuy being 1 given that category.

4. PRIMEUNIT, AUCGUART: I found that most of the values in these features are missing. The missing value is treated as another category, like "No info" in the dummy variable.

5. Dummy variables: After I converted the first 3 steps above, I converted the remaining categorical features into dummy variables. You can find more details in the“Make dummy variables” section in Jupyter Notebook.

- Numerical Feature: You can find my all analysis for Numerical features in the Numerical Feature analysis section in Jupyter Notebook. In this report, I will briefly explain how I handled and updated numerical features.

    1. Missing value imputation: I could find missing values from MMRAcquisitionAuctionAveragePrice, MMRAcquisitionAuctionCleanPrice, MMRAcquisitionRetailAveragePrice, MMRAcquisitonRetailCleanPrice, MMRCurrentAuctionAveragePrice, MMRCurrentAuctionCleanPrice, MMRCurrentRetailAveragePrice and MMRCurrentRetailCleanPrice feaures. I imputed them with its mean values.

    2. Multicollinearity: I found really high correlations among acquisition-related features (MMR*, VehBCost). Multicollinearity can cause problems when we build a model. So to solve this, I used PCA to replace these correlated features with a smaller set of uncorrelated components. You can find more details on the “Remove multicollinearity” section in Jupyter Notebook.

**Model building**

After a pre-processing, the data had 88 features. I used a  Random Forest model with these features.  Since the target feature is imbalanced, I trained models by using both under-sampled and over-sampled data.

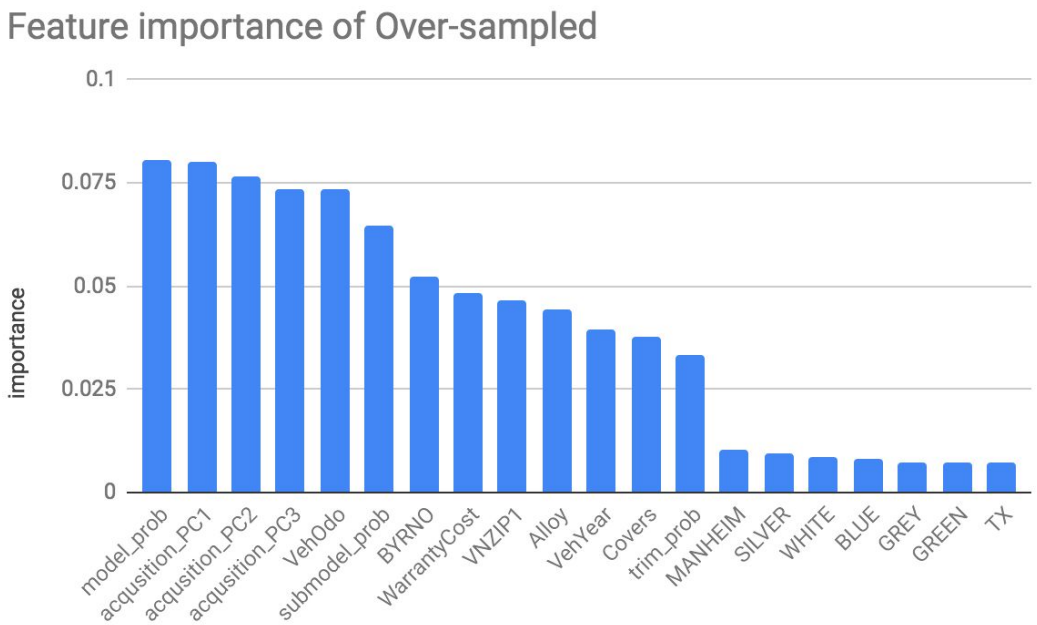
- Hyper-parameter tuning: I selected the hyper-parameters by using a grid search.

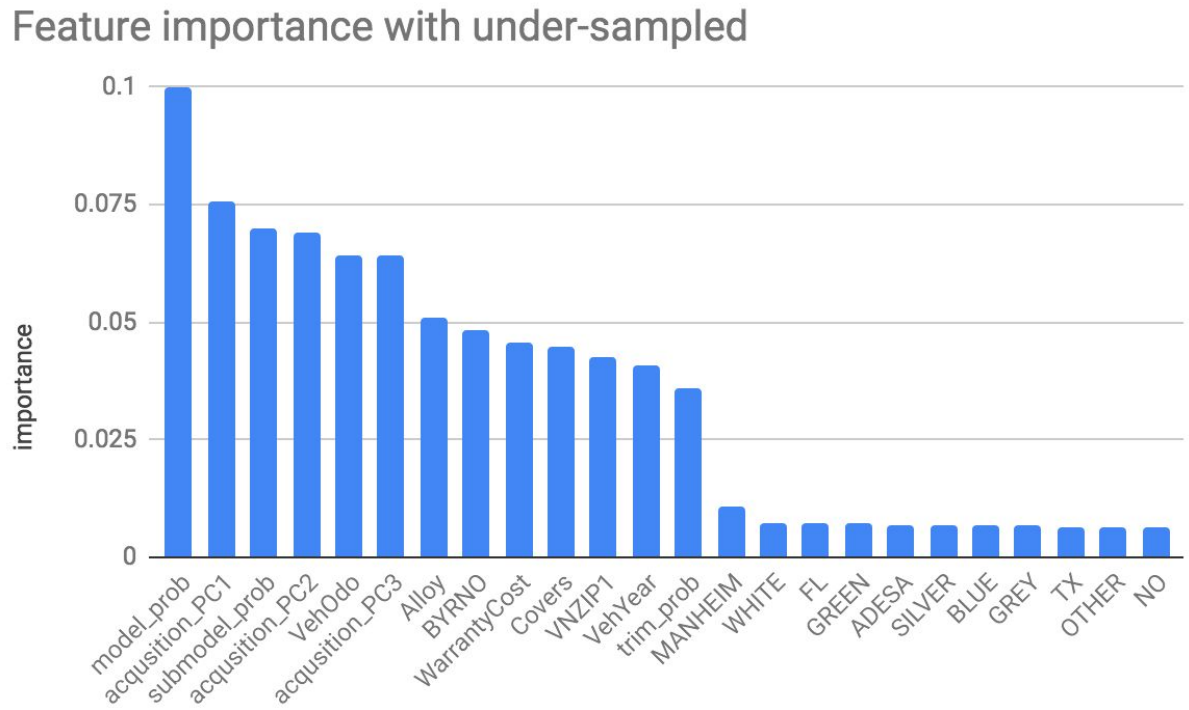
| Hyper-parameters | min_samples_split | n_estimators | max_depth |
|---|---|---|---|
| Original data | 10 | 70 | 100 |
| Under-sampled data | 10 | 70 | 100 |
| Over-sampled data | 10 | 70 | 50 |

- Model Evaluation:  I evaluated each model through 3-fold cross-validation.  And from the results, we can see that the model trained with over-sampled data shows the best performance.

| Metrics | Accuracy | Precision | Recall | F1 |
|---|---|---|---|---|
| Original data | 0.898 | 0.895 | 0.598 | 0.327 |
| Under-sampled data | 0.701 | 0.704 | 0.692 | 0.698 |
| Over-sampled data | 0.965 | 0.953 | 0.978 | 0.965 |

- Feature Importance: When we look at the feature importance of the models trained with under/over-sampled data, we can find that model, acquisition-related components and odometer are the most important features for predicting a defected car.



**Predictions for test set**

There could be overfitting problem in the model trained with over-sampled data, but I made predictions by using the mode trained with over-sampled data because it shows the best performance among the models and feature importances of the models trained with under/over-sampled data are pretty similar.

**Conclusion**

In conclusion, we can see that a random forest model trained with over-sampled data shows better performance compared to the other models. We also confirmed that model, PCA components and odometer are the most important features for predicting a defected car. In order to avoid defected cars, we could use this model to identify cars which are more likely to have problems.

**Future works**

I will investigate the possibilities of overfitting of the model trained with over-sampled data and will try a more advanced over-sampled approach, such as SMOTE. Additionally, I will also try to build more models and compare their performance.